Visualizing High-Dimensional Structure with the Incremental Grid Growing Neural Network

Justine Blackmore

Report AI95-238 August 1995

Artificial Intelligence Laboratory
The University of Texas at Austin
Austin, TX 78712

Visualizing High-Dimensional Structure with the Incremental Grid Growing Neural Network

by

Justine Marie Blackmore, B.S., B.S., B.A.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Arts

The University of Texas at Austin

December 1995

Visualizing High-Dimensional Structure with the Incremental Grid Growing Neural Network

Approved by
Supervising Committee:

Risto Miikkulainen

Roy Jenevein

Visualizing High-Dimensional Structure with the Incremental Grid Growing Neural Network

Justine Marie Blackmore, M.A.
The University of Texas at Austin, 1995

Supervisor: Risto Miikkulainen

Understanding high-dimensional real-world data usually requires learning the structure of the data space. The structure may contain high-dimensional clusters that have important topological relationships. Methods such as merge clustering and self-organizing maps are designed to aid the visualization of such data. However, these methods often fail to capture critical structural properties of the input. Although self-organizing maps capture high-dimensional topology, they do not represent cluster boundaries or discontinuities. Merge clustering extracts clusters, but it does not capture local or global topology. This thesis presents an algorithm that combines the topology-preserving characteristics of self-organizing maps with a flexible, adaptive structure that learns cluster boundaries in the data. It also proposes a method for analyzing the quality of such visualizations, and outlines how it could be used for automatic parameter tuning.

Contents

Chapter 1

Introduction

1.1 The problem

Real world data is often very high-dimensional, and often has a structure that is difficult both to recognize and describe. When presented as a set of high-dimensional vectors in tabular form, the relationships between data items may be difficult to fathom. For instance, human blood can be tested for the presence or absence of hundreds of inherited traits such as blood types, HLA factors, proteins, and DNA markers (Cavalli-Sforza et al. 1994). Similarity between blood samples is an indication of genetic similarity between individuals. On a larger scale, the structure of a set of samples from populations around the world reflects the global organization of human genetics. Learning the structure of such a data set would yield knowledge of how human populations are related (figure 1.1). The domains of experimental psychology, marketing analysis and cognitive science also rely heavily on techniques for visualizing the statistical properties of high-dimensional data.

1.2 Standard tools

Because the complicated relationships in real world data are difficult to perceive, tools for visualizing high-dimensional data are crucial in discovering patterns in the data. Visualization involves mapping an unknown high-dimensional space (the data set) onto a drawable structure for inspection by the data analyst. Standard visualization tools generally fall into two categories: tree-based methods and dimension reduction methods. Popular tree-based methods include the merge clustering algorithm and the minimum spanning tree algorithm. The merge clustering algorithm represents cluster properties of the data in a 2-dimensional merge tree, such that the leaves and branches under each merge node are closer to each other than to any other cluster in the tree. The algorithm begins by making the data pair with the smallest distance the lowest split in the cluster tree. These two vectors are aver-

aged into a single vector representing the split, and this average replaces the two vectors in the data set. This process is repeated until only the last data item to be merged and an average vector representing all the other data items are left (see figure l.l).

The minimum spanning tree algorithm is another tree-based approach. This method constructs a lowest cost graph of the data, where the cost of a link is defined as the distance between two linked data vectors. This method produces a less constrained 2-D structure than the clustering methods just mentioned, and primarily exposes the closest pairwise relationships between items. However, the MST algorithm does not explicitly look for clustering patterns in the data.

Both of these tree-based methods, however, suffer from the same drawback: the high-dimensional topology of the data set is lost or unrecognizable (figure 1.2b). There may be important relationships between branches that are not represented because of the fixed structure or limited connectivity of the tree. In figure 1.1, for instance, one would expect the Lapp population to be genetically similar to the Finnish and Swedish populations, but such a relationship is not immediately apparent in the merge cluster tree. Similarly, in figure 1.2b the E1 data item is as closely related to C0 as it is to E0, but the merge tree does not reflect this.

While clusters in the data set are important, the overall topology of the data set is equally so. Dimension reduction techniques are designed to capture high-dimensional topology in lower-dimensional spaces (sub-manifolds). If the technique can be used to reduce the dimensionality to two, then it can be used for visualization. Such dimension reduction techniques include principal component analysis and multidimensional scaling, both popular methods for visualization. In principal component analysis, a new coordinate basis for the data is derived such that the coordinate axes are orthogonal and account for the data variance in decreasing order of magnitude. That is, when the data is projected onto the first principal component, this axis exhibits the maximum variance. The second principal component has maximum variance subject to being orthogonal to the first, and so on. In this way, the principal components account for the variance in the data set in descending order. Thus each principal component represents something about the data, and subsequent principal components recover some of the information lost in the previous reduced representations. For visualization purposes, only two or at most three such axes are useful. Generally the first and second components are used to account for as much of the data variance as possible. Note that when only two components are used, any information present in the other components is lost. Plotting the data along these two axes results in a scatter plot in which clustering of data items may be apparent. However, since there is no connectivity between individual data items, cluster boundaries are not explicitly represented.

Multidimensional scaling is another dimension-reduction technique that is often used

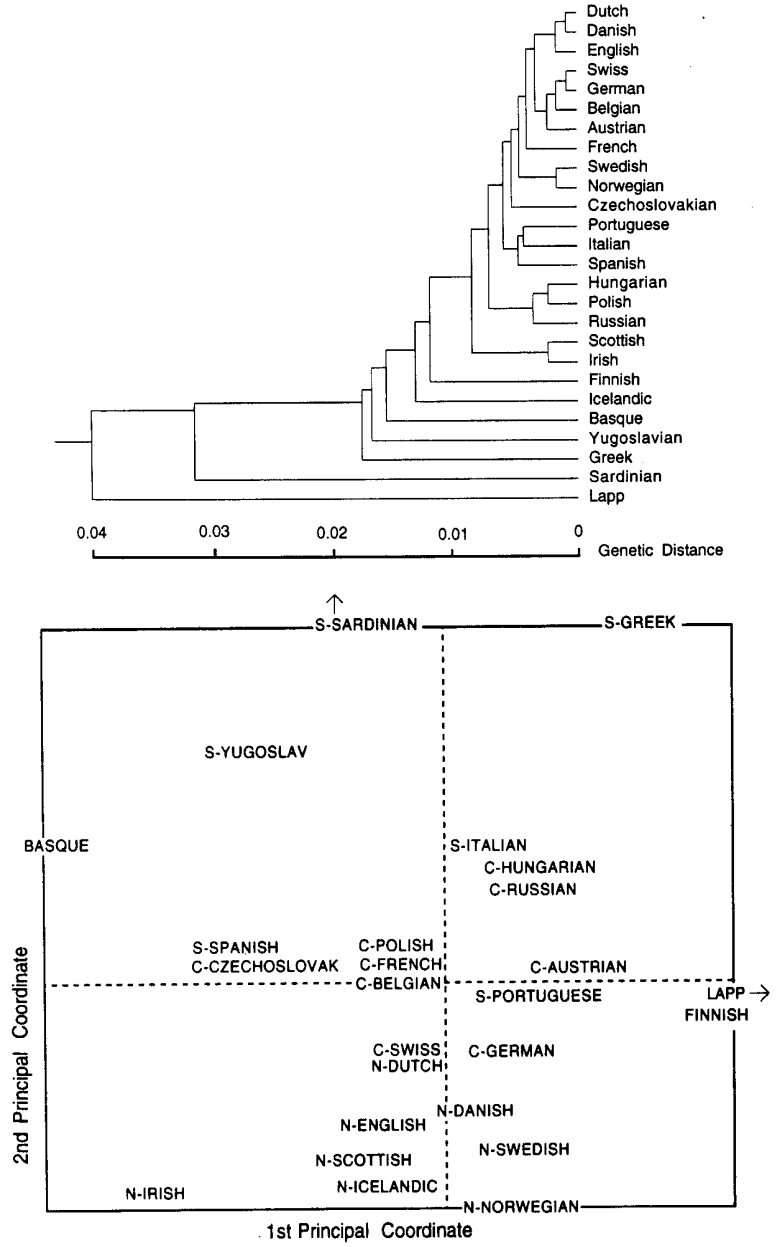


Figure 1.1: **Illustration of visualization task to be performed.** Genetics data tends to be extremely high-dimensional and complicated. No single visualization tool is sufficient to discover the structure of the data set. Hence such data sets must be examined by a combination of tools, each of which captures different properties of the data. (a) The merge clustering representation captures clusters. (b) The principal component representation shows topology. IGG is designed to capture both the clustering of merge tree and the topology of PCA in a single representation. No other visualization methods are capable of this. (From Cavalli-Sforza 1993)

for visualization. The goal here is to map the data onto a lower-dimensional space such that the N-dimensional distances between each pair of data items in the input are preserved to some user-defined level of acceptability in the lower dimension. In metric MDS, the goal is to approximate (to some scale factor) the actual distances between each data item. In non-metric MDS, the goal is to reproduce the distance rankings between each pair of data items in the lower-dimensional configuration. All MDS algorithms have two basic components: a) some criterion for deciding when a lower-dimensional configuration is good enough, and b) a procedure for moving each point in the low-dimensional space so that the configuration improves according to the criterion in (a). For visualization, the result must be a 2-D or 3-D drawing in which the N-dimensional distances or distance rankings between each pair of data items in the input are explicitly represented. However, no MDS algorithm can be guaranteed to find a stable 2-D or 3-D configuration. Also, as in PCA, there is no connectivity between data items in the 2-D or 3-D map. Without connectivity, the within-cluster structure of the data is lost.

More recently, self-organizing neural networks have been added to the list of dimension-reduction techniques. Kohonen's (1989, 1990) self-organizing map algorithm maps high-dimensional data onto a fixed network of nodes. Vectors nearby in the high-dimensional input space are mapped onto nearby nodes of the network. The network structure preserves the topology of the data set as much as possible. However, the network cannot learn or represent discontinuities in the data due to its fixed grid connectivity (figure 1.2b).

Thus it seems each of the above methods for visualization is limited in some way. Either they focus on extracting information about clusters in the data without considering topology, or on capturing topology without any explicit clustering. The shortcomings of both tree-based techniques and self-organizing networks originate from the fixed representation strategy: the drawable structure is not flexible, and therefore cannot adapt itself to the input space. On the other hand, techniques that have no connectivity (PCA, MDS), fail to encode information about local cluster topology in their 2-D representations. As a result, each of the above methods often fails to discover important patterns in the data.

1.3 Self-organizing map tools

Self-organizing map algorithms have recently been developed that can potentially overcome the above limitations. These methods incrementally grow and prune a network (Fritzke 1991a, l991b, 1992; Jockusch 1990; Kangas et al. 1990; Martinetz and Schulten 1991; Ritter 1991; Rodriques and Almeida 1990; Xu and Oja 1990). The algorithms employ heuristics to ensure that nodes are added only where the network needs them to represent the input space, and that nodes are deleted only when they do not represent any part of the input space. Most of the resulting networks, however, have arbitrary dimensionality and connectivity.

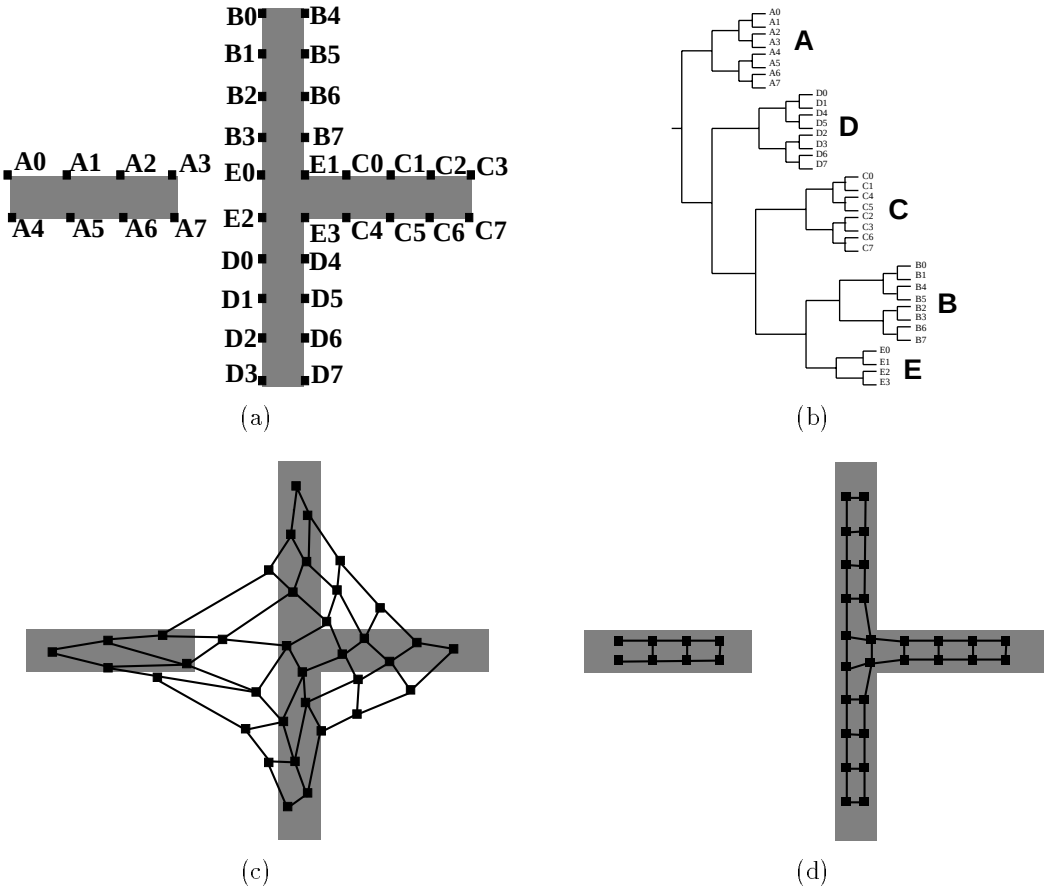


Figure 1.2: **Capturing cluster boundaries in a 2-D input space.** (a) The input space consists of 36 two-dimensional vectors uniformly distributed in the shaded regions. The vectors were labeled A through E according to the region and numbered. A represents the separated cluster, B, C, and D the connected arms and E the area connecting B, C, and D. (b) The merge clustering representation. The tree structure has captured the clusters in the data, but the overall topology of the data set is not apparent. (c) The self-organizing map representation. The black dots indicate the locations of the network's weight vectors after the map has been organized. Lines indicate network connectivity. Although the map has captured the overall 2-D topology, its connections span the cluster boundaries. Note also that nodes have been allocated to areas where there is no input, representing the structure of the data set inaccurately. (d) A more desirable 2-D network visualization that might be obtained with incremental grid growing. This network has 4 clusters: 3 are connected through a small number of nodes, and the fourth is separate from the others. Note that neither PCA nor MDS is represented here: for 2-D input, these algorithms are not useful.

There is no guarantee the final network structure can be easily drawn in two dimensions. Therefore these new methods are not ideally suited for visualization of high-dimensional data.

1.4 Incremental Grid Growing

This thesis presents an incremental self-organizing algorithm called Incremental Grid Growing (IGG) that captures both complicated topology and high-dimensional cluster boundaries. The network is strictly 2-dimensional, but incrementally adapts its shape and connectivity to the structure of the data set (figure 1.2d). The local and global structure of the input is automatically embedded in the 2-D drawing.

1.5 Outline of thesis

The organization of the thesis is as follows. Chapter 2 discusses Kohonen's self-organizing map (SOM) algorithm in detail. In chapter 3, the IGG algorithm, which is built around Kohonen's self-organizing map algorithm, is described in detail. Chapter 4 demonstrates how IGG represents the topology of various high-dimensional spaces. For illustration, the results of applying IGG to 2-D and 4-D spaces are presented. Chapter 4 also demonstrates how IGG captures the clustering and topology of a 5-D data set with graph-like, hierarchical structure (the minimum spanning tree task). Finally, IGG is demonstrated on a 240-D semantic data set derived from Webster's online thesaurus. Chapter 5 discusses a real-world application of IGG in the domain of population genetics, where visualization is essential to understanding the data. The application comes from the research presented in L. Cavalli-Sforza et al. *The History and Geography of Human Genes*. After describing the domain context, chapter 5 concludes with a comparison of merge cluster, PCA and IGG representations of high-dimensional genetics data taken directly from the book. Chapter 6 discusses the practical issues in implementing and using the IGG algorithm, such as parameter tuning and map quality assessment. Chapter 6 also proposes a measure for determining how well high-dimensional topology is captured in a two-dimensional map. Quantifying the degree of topology preservation is an important research topic, and future work with IGG will concentrate on this area. Chapter 7 concludes the thesis with a summary of the main goals and results of the research.

Chapter 2

The self-organizing map and related neural network techniques

Kohonen's self-organizing map algorithm (SOM) maps items from an arbitrary input distribution onto a planar network of neurons. The algorithm causes a 2-D neural network to develop topological patterns that are similar to those present in the input space (figure 2.1). If the input space is of dimension greater than 2, the self-organizing map thus performs a dimensionality reduction from the higher-dimensional input space to the 2-dimensional network (figure 2.3). As mentioned in the introduction, such a dimensionality reduction can be a great aid to visualizing and understanding the data.

2.1 SOM algorithm

The SOM planar network can be of any geometry (usually rectangular). Each node is connected only to its direct neighbors (figure 2.1). Each item in the input is represented by an N-dimensional vector, and each node in the planar network is associated with an N-dimensional weight vector. During organization, each item in the input is presented to the network a number of times in random order. When an item in the input is presented to the network, the node whose weight vector most closely matches the input vector is chosen as the winner. The weight vector of the winning node is then modified so that the difference between it and the input decreases by a fraction α_{winner} (the learning rate). In addition, the weight vectors of any neurons within a certain 2-D neighborhood N_c of the winner are tuned toward the input by a fraction α_{ngb} (generally less than α_{winner}). Each "epoch" of training consists of presenting, in random order, each input vector to the network, determining the winning unit, and modifying the winner's weight vectors and those of its neighbors. The neighborhood function defines how far from the best matching unit nodes are to be tuned. Usually it is constant or Gaussian. Thus the core learning rule of the self-organizing map

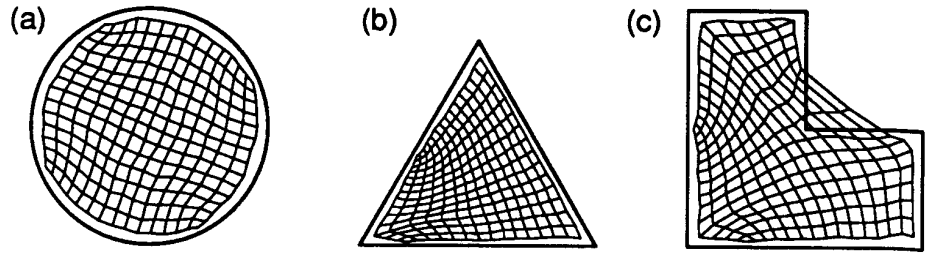


Figure 2.1: **Kohonen's 2-dimensional self-organizing map.** The uniform 2-D input space is defined by the outline. The position in 2-D space of the weight vectors in the map is indicated by intersecting lines—the lines represent the connections of the network. The weight vectors organize themselves in a way that closely resembles in the input space. Note that in the case of convex input (c), the mapping is less accurate. (From Hertz, Krogh and Palmer 1991)

is as follows:

Given the current input vector x, let i be the winning node chosen such that

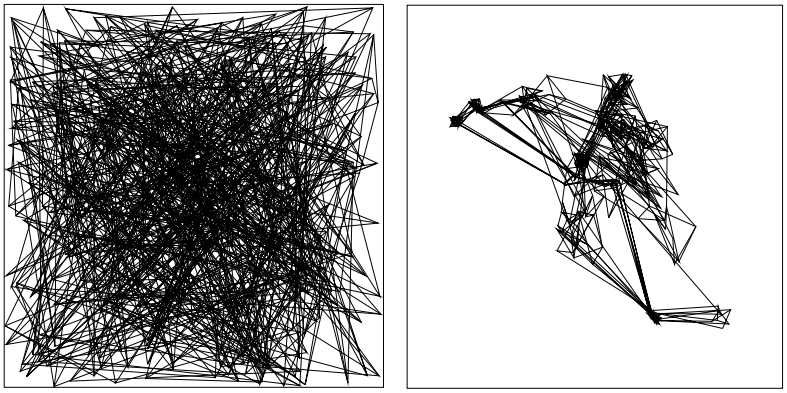
$$|w_i - x| \leq |w_j - x|, \quad \forall j \neq i \tag{2.1}$$

where w_k denotes the weight vector of node k. Then for each input vector x, the network is updated such that:

$$\begin{aligned} w_i &= w_i + \alpha_{\text{winner}}(x - w_i) \quad \text{and} \\ w_n &= w_n + \alpha_{\text{ngb}}(x - w_n) \quad \forall n \in N_c(i). \end{aligned} \tag{2.2}$$

The parameters to be selected by the user are the learning rates (α_{winner} and α_{ngb}), the neighborhood function (N_c) and the number of epochs to train for. In practice, the neighborhood function must allow the neighborhood to be large initially (on the order of half of the largest dimension in the map), and decrease to 0 over the number of epochs the map is trained for. As a result of applying this learning rule over many epochs, the weight vectors in the network become ordered such that similar items in the input are mapped onto near neighbors in the 2-D network.

The algorithm performs a topology-preserving mapping in the sense that vectors close in the input space are mapped onto neighboring nodes in the network. A classic SOM example is the so-called robot arm example. In this example, a robot arm's position is described by the combination of angles defined by its many joints. The problem to be solved is this: given a position in 2-D space, what are the joint angles that must be used to place the robot's hand there? If one trains a 2-D SOM on example combinations of the robot arm's joints, the map will capture the topology of the joint space in its weight vectors. When the map has been fully trained, one can choose a position in the 2-D map



(a) 0 samples (b) 30 samples

(c) 100 samples (d) 10,000 samples

Figure 2.2: **Time evolution of the SOM weight vectors for 2-D input.** The uniform 2-D input space is indicated by the outline. Initially the nodes are given random weight vectors. As organization proceeds, the topology of the weight vectors begins to resemble the topology of the input. By 100 epochs, a gross, global order has emerged. At 10000 epochs, the weight vectors have settled into an excellent representation of the input space. (From Miikkulainen 1993)

and obtain the desired angle combinations by examining the node's weight vector. The topology preserving property of the SOM algorithm makes this possible.

2.2 SOM theory

Most of the theoretical work on Kohonen's algorithm has focused on the ordering and convergence of 1-dimensional input mapped onto a 1-dimensional network (Kohonen 1989, Cottrell and Fort 1987). In addition, the stability and convergence properties of 2-dimensional maps given a specific input distribution (uniform) has also been analyzed (Ritter and Schulten 1988). Generally, stochastic methods are used to analyze the self-organizing algorithm as a

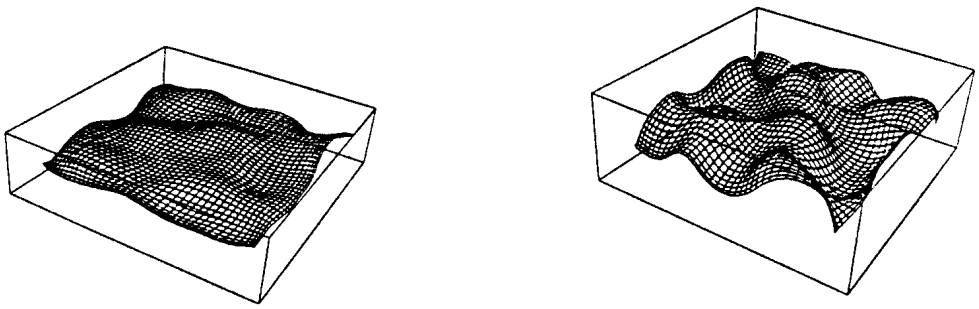


Figure 2.3: **Dimension reduction with Kohonen's SOM.** The uniform 3-D input space is indicated by the bounding box. The network weights organize themselves to cover the entire input space while preserving the topology of the input space. The fact that none of the network connections must cross another indicates that local topology is preserved in the map—nearby points in the input space are mapped onto neighbors of the map. (From Ritter and Schulten 1988)

Markov process. The weight vectors of the map constitute the states of the Markov process, and the state transitions are defined by the self-organizing learning rule above. Transitions are triggered by random selection of input vectors. Modeling the 1-D self-organizing map (figure 2.4) as a Markov chain, Kohonen (1989) proved that his algorithm converges to an ordered, stable state in the case of 1-D input and 1-D map. Defining an ordered state of the 1-D map to be a state such that the ordering of the weight vectors mirrors exactly the ordering of the input values, and using a measure of disorder that describes the total amount of node misplacements, Kohonen ex mined the possible transitions for the 1-D map case-by-case. He showed that if the input is randomly selected, this measure of disorder is more likely to decrease than increase, and can be expected to converge to 0 and stay there. With this result, Kohonen showed for the 1-D case that: a) once the weight vectors become ordered, they never become disordered, and b) the point density of the nodes will finally approximate that probability density of the input. Ritter (1989) later quantified the relationship more exactly. He showed that given an input probability density P, the map's point density approaches $P^{(2/3)}$.

Unfortunately, the theoretical work on the 1-D map case is not immediately extensible to the 2-D case because in 2-D the vectors are not as clearly ordered as in the case of the linear array. Several researchers have applied Markov methods to the analysis of the 2-D map. Most notably Ritter and Schulten showed that given uniform input, there are 2 equilibrium states possible corresponding to two absorbing states for the Markov process (1988). They then showed that convergence to one of these states is guaranteed if one imposes certain conditions on the neighborhood function. Ritter and Schulten also derived a Fokker-Planck equation describing the time evolution of the map's point density function during the final convergence phase. Careful analysis of this function reveals that statistical

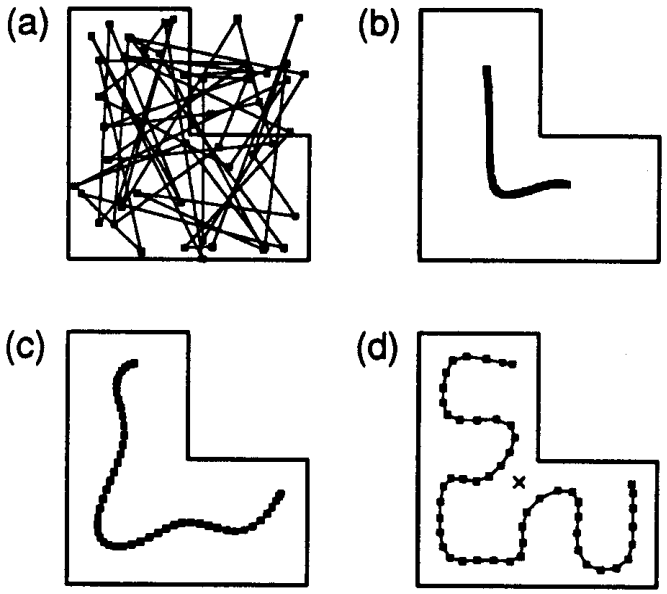


Figure 2.4: **The 1-D Kohonen SOM.** The 2-D input is mapped onto a 1-D linear array of nodes. Again the nodes are randomly initialized, and proceed quickly to gross topological order. The array is lower dimensional than the input, so its nodes must curve around in order for the weight vectors to cover the whole input space. As a result, the overall topology of the array is somewhat disturbed. (From Hertz, Krogh and Palmer 1991)

fluctuations can occur in the weight vectors during convergence, and that their character can depend on the time dependence of the learning rate. So far this type of analysis has only been carried out for the case of 3-D uniform input (Ritter and Schulten 1988). Extension to more general input spaces should provide important clues to predicting the final state of the map for arbitrary input.

In practice, it is possible for twists and other distortions to appear in the network topology (figure 2.5). Predicting the final topological state of the 2-D map is one of the primary yet most elusive goals of the theoretical analysis of SOM. Another goal is determining how the final state depends on the time evolution and shape of the neighborhood function and learning rates. Several researchers have proposed order definitions to quantify how well the 2-D map preserves the topology of the input space (Demartines 1992, Zrehen and Blayo 1992). These definitions of topology preservation are then used to study the effect of neighborhood size and shape on convergence of the 2-D map. For example, Lo, Fujita and Bavarian (1991) use a definition of order localized to the neighborhood of winner units. For this definition they are able to show that convergence to a state where neighborhoods

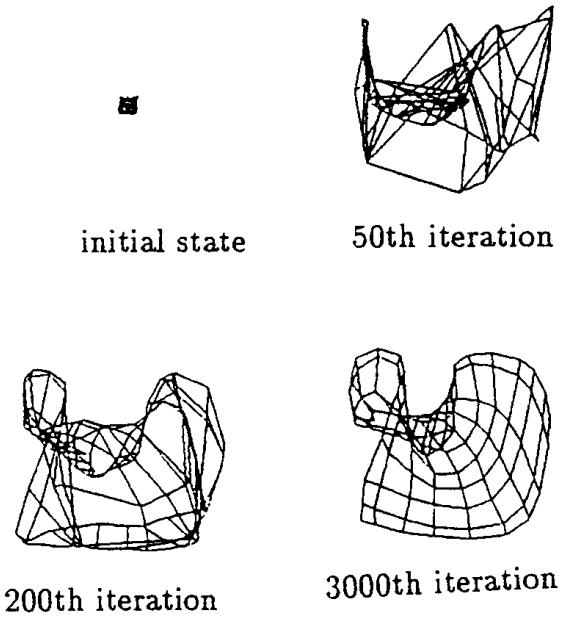


Figure 2.5: **2-D Kohonen map with twist.** The SOM algorithm is not guaranteed to achieve correct global order. Here the weight vectors have organized themselves into a distorted shape. The topology is disturbed and the weight vectors do not fully cover the input space. Because the input here is 2-D, simply plotting the weight vectors of the network reveals the poor topology. However, if the weight vectors were high dimensional, they could not be plotted. In that case, the twisted topology might go undetected. (From Lo, Fujita and Bavarian 1991)

never become disordered proceeds r faster when the neighborhood function more closely resembles the "Mexican hat" of lateral interaction profiles.

The inability to provide a clear mathematical formalism which describes the function computed by the self-organizing process has frustrated many researchers. Without such a formalism, it is difficult to predict how changes in algorithm parameters affect the outcome, or to determine how parameters interact to produce a final level of order preservation in the mapping. The result is that the exact mapping that is produced depends to some extent on how the experimenter tunes the parameters. Therefore the map may reflect some subjective quality that the experimenter wants to see in the 2-D representation of the data. Because the researcher is the only judge of the final quality of the map, he/she must know a priori what structure the map should develop. If the map quality cannot be defined quantitatively, then the algorithm cannot be judged objectively. In chapter 6, a measure for quantifying the map's topological order is proposed. Preliminary results suggest it will be useful for analyzing map quality.

Judging the quality of the map is further complicated by the fact that the fixed

geometry and connectivity of the network can misrepresent the high-dimensional structure of the input. The input may have high-dimensional clusters, but cluster boundaries are not represented in an ordinary self-organizing map. In a network where each node is connected to all of its direct neighbors, discontinuities will appear bridged, and nodes may acquire weight vectors situated within a discontinuity where the input probability is 0 (figure 2.1c). An SOM 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. For instance, in the robot arm example, there may be obstacles in the 2-D space or illegal joint combinations. The SOM needs to capture this property explicitly in order to guarantee the obstacles are avoided and the illegal joint combinations never attempted

2.3 Incremental SOM approaches

Self-organizing map algorithms have recently been developed that attempt to overcome the representation difficulties associated with the standard SOM (Fritzke l991a, l991b, 1992; Jockusch 1990; Kangas et al. 1990; Martinetz and Schulten 1991; Ritter 1991; Rodriques and Almeida 1990; Xu and Oja 1990). These algorithms are generally incremental in nature, adapting their network structure to the input space dynamically. These methods employ heuristics to ensure that nodes are added only where the network needs them to represent the input space, and that nodes are deleted only when they do not represent any part of the input space. The idea is that in order to develop an accurate representation of the topology, an 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. Thus an algorithm must be equipped with 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.

For example, Fritzke's growing cell structure algorithm incorporates heuristics for both adding to and removing from the network nodes and connections. These heuristics allow the algorithm to build a network that captures the high dimensional structure of the input and to correct inaccurate network features. In Fritzke's algorithm, the basic 2-D grid of SOM has been abandoned and replaced with nodes whose connectivity at all times define a system of triangles. 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. In the case of 2-D input, it is easy to verify that the network structure accurately represents the input space

by plotting the weight vectors in 2-D. When the input is high dimensional however, such an arbitrary structure may not be easily drawable in 2-D. Fritzke presents a drawing method that works reasonably well when the input space is of comparatively low dimension (e.g. 3-D). However, when the input is truly high-dimensional, the resulting drawing cannot be guaranteed to be planar. Thus Fritzke's self-organizing algorithm is not well-suited to the problem of visualization.

Most of the algorithms that grow and prune network structures result in networks of arbitrary dimensionality and connectivity. There is no guarantee the final structures can be easily visualized in two dimensions. The incremental self-organizing algorithm described in this thesis avoids the difficulties of an arbitrarily connected network by retaining a regular, 2-D grid structure at all times. At any point during the organization, the map has a simple 2-dimensional description, and topological relations are easily visualized. The IGG algorithm incorporates the adaptive, structure-enhancing heuristics of the above approaches into an incremental version of the standard SOM algorithm. This results in a 2-D map that captures both complicated topology and high-dimensional cluster boundaries. The network is strictly 2-dimensional, but incrementally adapts its shape and connectivity to the structure of the data set. The local and global structure of the input is automatically embedded in the 2-D drawing.

Chapter 3

Incremental Grid Growing

The incremental grid growing algorithm is designed to overcome the limitation of the fixed grid in self-organizing maps. IGG embeds the cluster boundaries directly in its 2-D network structure (figure 3.1). IGG builds the network incrementally, dynamically adapting its structure and connectivity according to the input data.

3.1 The IGG algorithm

Initially, the grid consists of four connected nodes with weight vectors chosen at random from the input space (figure 3.2). The following three steps are then iterated:

1. Adapting the current grid to the input distribution through the self-organizing map process,

2. Adding new nodes to the perimeter of the grid where the network is exhibiting a large errors in representation (figure 3.2a-d);

3. Examining the vectors of neighboring nodes to determine whether a connection between the nodes should be deleted from the map, or a new connection added (figures 3.2e-h).

The core of the learning of IGG, therefore, consists of the self-organizing map algorithm. The additional techniques of growing new nodes and adding and deleting connections allow the network to evolve a 2-D structure that reflects the relationships in the data. These techniques are described in detail below.

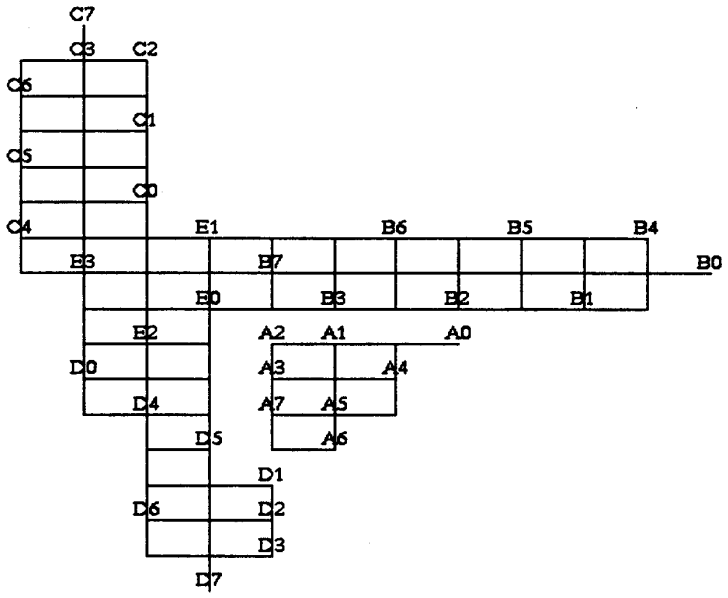


Figure 3.1: **IGG network for the 2-D cross-shaped input from figure 1.2a.** The input space structure is embedded in the 2-D network. Thus the structure of the input space is apparent in the network, without the necessity of plotting any weight vectors.

3.2 Growing new nodes

A boundary node is defined as any node in the grid that has at least one directly neighboring position in the 2-D grid space not yet occupied by a node figure 3.2). Each boundary node in the current network maintains an error value E over each organizational pass. Whenever an input vector is mapped onto a boundary node, the square of the distance between the input vector and the node's weight vector is added to the error value:

$$E(t) = E(t-1) + \sum_k (x_k - w_k)^2, \tag{3.1}$$

where E is the cumulative error, $\mathbf{w}$ the weight vector of the winning unit, and $\mathbf{x}$ is the input vector.

Large cumulative error values occur at nodes that have too many input vectors mapped onto them. Their weight vectors fail to adequately represent all of the input vectors in that area. Therefore, new nodes are added to the grid in the areas that have high cumulative error. New nodes are only added on the boundary, so that the structure remains 2-D and drawable at all times. During self-organization, these new nodes on the perimeter develop weight vectors for those areas of the input space that were previously

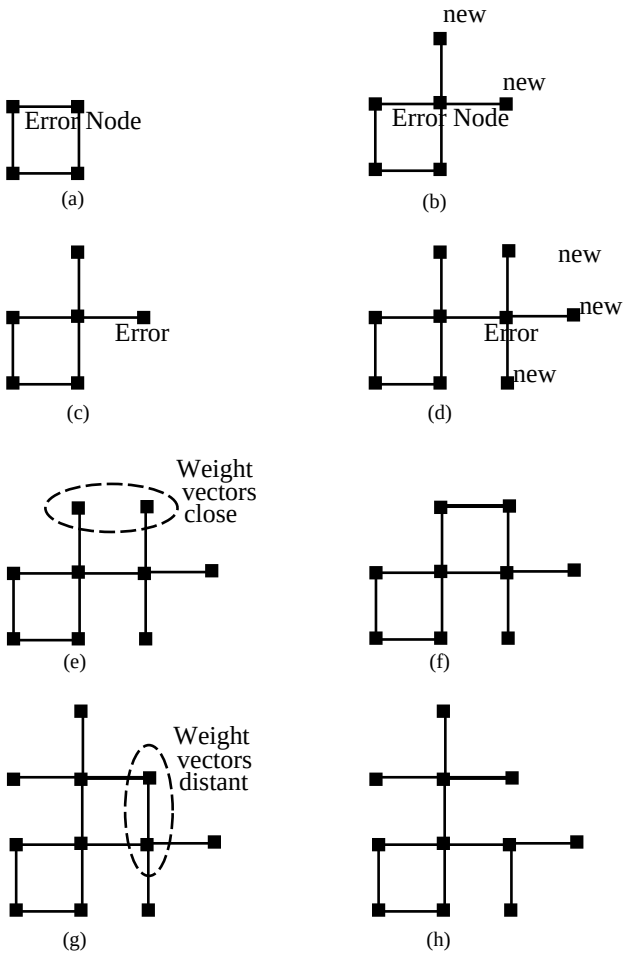


Figure 3.2: **Growing the map grid.** Figure (a) shows the initial structure after the first organization stage; the boundary node with the highest error value is marked. (b) New nodes are "grow" into any open grid location that is an immediate neighbor of the error node. (c) After organizing the new structure with the standard self-organization process, 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.

inadequately represented. The new nodes are directly connected to the error node. If any other directly neighboring grid spots are occupied (as in figure 3.2d), the new node's weight vector is initialized to be the average value of all the neighboring weight vectors:

$$w_{\mathrm{NEW},k} = 1/n \sum_{i \in N} w_{i,k}, \tag{3.2}$$

where $w_{\mathrm{NEW},k}$ 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 3.2b), the new node's weight vector is 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:

$$w_{\mathrm{ERR},k} = 1/(m+1) \left(w_{\mathrm{NEW},k} + \sum_{i \in M} w_{i,k} \right), \tag{3.3}$$

where $w_{\mathrm{ERR},k}$ 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.

Because new nodes are added only to areas that need them, each node in the structure always represents some region of the input space that lies inside a cluster. Therefore, node deletion is not necessary in incremental grid growing.

3.3 Adding and deleting connections

Initially, the new nodes are connected to the structure only through the high error node. During organization, the new weight vectors may become similar to the weight vectors of neighbors to which they are not connected. In this case, a new connection is added joining these nodes. An adjustable threshold parameter is used to decide if such a new connection should be grown. After each organizational pass, the similarity metric (e.g. Euclidean distance) between unconnected neighboring nodes is examined. If the value is below the *connect* threshold, a connection is added (see figures 3.2e-f). Similarly, a *disconnect* threshold is used to determine if there are two nodes in the map that are connected although they have evolved into separated areas of the input space. Exceeding such a threshold may indicate that a connection crosses a cluster boundary, and should be deleted from the grid (figures 3.2g-h).

Adding nodes only at the perimeter ensures that the map remains drawable at all times. The dynamic addition and deletion of connections allows the grid to learn high-dimensional cluster boundaries in the input, avoiding the limitation of standard self-organizing maps. As a result, the grid-growing algorithm is a flexible, dynamic tool for discovering the structure of complicated high-dimensional data.

3.4 Complexity

Because the central organizing step is based on the standard self-organizing algorithm, which spends most of its time computing the Euclidean distance between every network node and every input vector for tens of epochs, IGG's time complexity can be combinatorial in the number of inputs, number of dimensions, and network size. Little can be done about the properties of the input, but the running time can be reduced to linear in the network size by making the search for the closest weight vector for each input vector more local. By searching only a small neighborhood around the node that was closest to the current input in the previous epoch, the search time becomes constant for all epochs, regardless of the current network size. This optimization is possible because the network fully learns the input space during each organizational phase, making a local search sufficient. As an example, in our implementation we saw a speedup factor of 7 when the search was localized to a 1×1 neighborhood of the map. This is a promising result, and suggests that building large maps for very large data sets should be tractable.

Chapter 4

Testing IGG

This chapter presents the results of applying IGG to different examples of input spaces. First, a simple 2-D input example is presented for illustrative purposes, then expanded to a 4-D example. Then the minimum spanning tree of Kohonen (1990) is presented for purposes of comparing IGG to other visualization methods. Finally, the results of applying IGG to high-dimensional real world semantic data derived from Webster's thesaurus is presented as a demonstration of the algorithm's capabilities. Chapter 5 will present a real world population genetics application. Both this chapter and chapter 5 are devoted to demonstrating the algorithm's capabilities, without any discussion of IGG parameter settings. In the chapter 6, the problem of experimentally determining the best parameter settings will be discussed.

4.1 Illustration of IGG: 2-D and 4-D uniform random input

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. To illustrate IGG and demonstrate that it captures input space topology more accurately than SOM, IGG has been applied to a simple 2-D cross-shaped input space (figure 4.1). The 2-dimensional vectors were chosen with uniform probability from the cross-shaped shaded area. The grid developed four arms connected through an area that represents the central portion of the cross (figure 4.1). 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 continuity of the resulting grid. Note that the grid structure itself, even without any labeling 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.

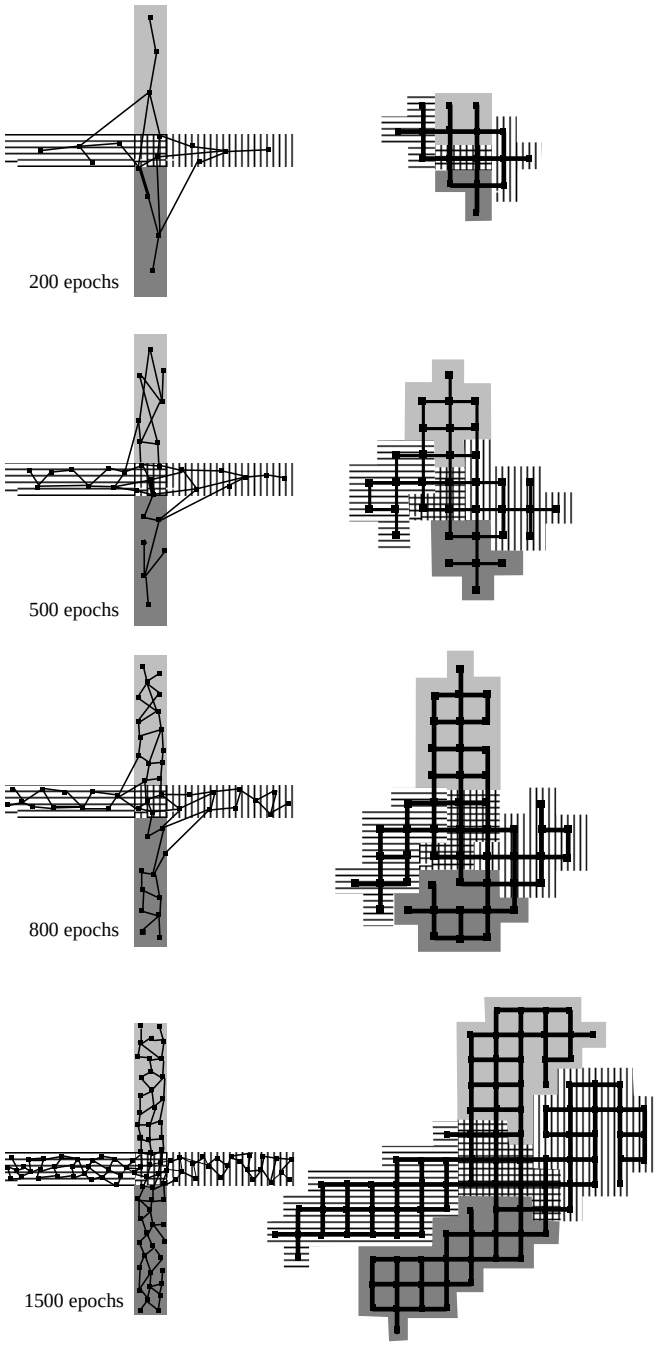


Figure 4.1: **Snapshots of the grid evolution for a 2 dimensional cross input.** On the left, the weight vectors are plotted on the 2-D input space. On the right, the corresponding grid structure is shown. Shading of the area around a node indicates the arm of the cross where that node's weight vector is located. The four arms are separated in the grid, with a common center.

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:

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

The final map representing this structure is shown in figure 4.2. 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. Note that the SOM cannot accurately reflect the convex, discontinuous nature of the input. Because each node is connected to each of its direct neighbors, the convex boundaries and discontinuities are smoothed out in the SOM weight vectors. The IGG representation, on the other hand, directly embeds the convexities and discontinuities in its 2-D structure. Thus the structure of the input is evident by inspection, without the necessity of examining weight vectors.

4.2 Comparison using minimum spanning tree data

To illustrate how the grid-growing algorithm differs from the standard visualization methods such as merge clustering and self-organizing maps, consider the minimum spanning tree example of Kohonen (1990). In this example, the input consists of the 5-dimensional vectors listed in figure 4.3. A superficial look at the input indicates that there are clusters of similar vectors in the data, as well as relationships between the clusters. The high-dimensional topology of this data set is difficult to describe, but a minimum spanning tree is one 2-D structure that in this particular case captures the hierarchical nature of the data quite well.

Conventional 2-D visualizations fail to represent the essential properties of the spanning tree. When merge clustering is applied to this data set (figure 4.4a), clustering is apparent in the resulting tree. However, the merge tree does not make the global and local relationships between elements clear. For example, the cluster *1-2-3-4-5-6* is split in a way that implies *1-2-3-4* is a single cluster, and *5-6* is a separate one. On the other hand, the hexagonally connected self-organizing map learned both the global and local topology of

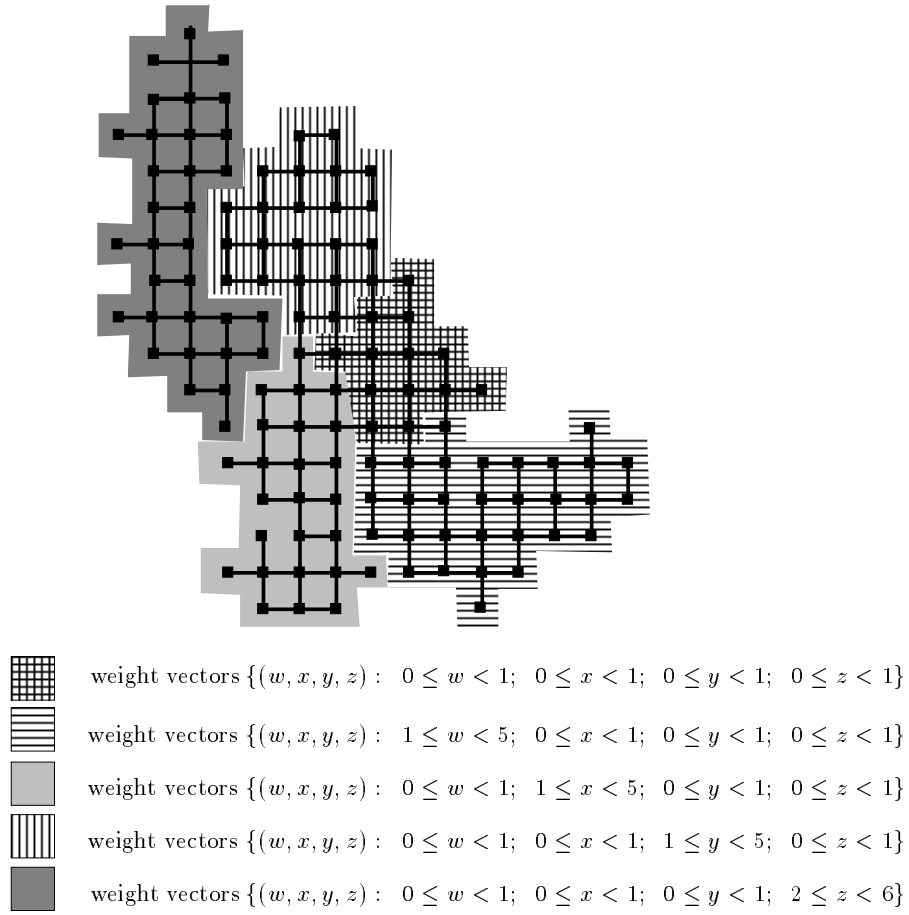


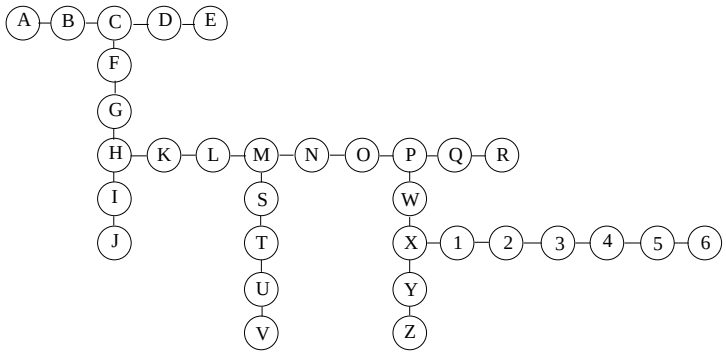
Figure 4.2: **Final grid structure for 4-dimensional, disjoint input space.** The common center (the first area listed) and the first three arms are connected, while the fourth arm is fully separated.

the data; however, it has no way of representing the cluster boundaries (figure 4.4b). For instance, the *S-T-U-V* cluster appears to be equally related to most of the other clusters. Both representations are therefore incomplete.

The incremental grid growing algorithm applied to the same data set results in a structure that captures both the cluster boundaries and the topology of the data (figure 4.5). 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 connectivity between clusters. For example, the *S-T-U-V* cluster is clearly related to the *M-N* portion of the *K-L-M-N-O-P-Q-R* backbone, and it is clearly not very well related to the other clusters. Similarly, the *F-G-H-I-J* cluster is related to the *A-B-C-D-E* cluster near the *B-*

ABCDEFGHIJKLMNOPQRSTUVWXYZ123456
12345333333333333333333333333333
00000123453333333333333333333333
00000000001234567833336666666666
00000000000000000012341234222222
00000000000000000000000000123456

(a)



(b)

Figure 4.3: **(a) The input vectors of Kohonen (1990).** (b) The minimal spanning tree of the data. The example was designed to illustrate the self-organizing map's capacity to represent the general topology of hierarchical data. In this case, the minimal spanning tree is one relational description that happens to capture the structure quite well.

C region, and to the *K-L-M-N-O-P-Q-R* backbone nearby *K*. Clearly, the algorithm has evolved a network that visualizes the underlying structure in the data.

4.3 Demonstration using real world semantic data

The spanning tree example is a good illustration of the properties of visualization techniques, but it is still a toy problem. Real world data is usually more challenging to visualize in 2-D. Each vector may contain hundreds of features, and the data set may contain thousands of such vectors. Very complex relationships between data elements are possible. In addition, real world data sets often contain a significant amount of noise. Minimum spanning trees (and other such simple structures) are usually insufficient to represent the complicated relationships in unknown, high-dimensional data.

Consider for example word semantics. The variety and complexity of relationships between word meanings overwhelms simple representation schemes. New learning and representation methods are often demonstrated using such semantic data (Miikkulainen 1993;

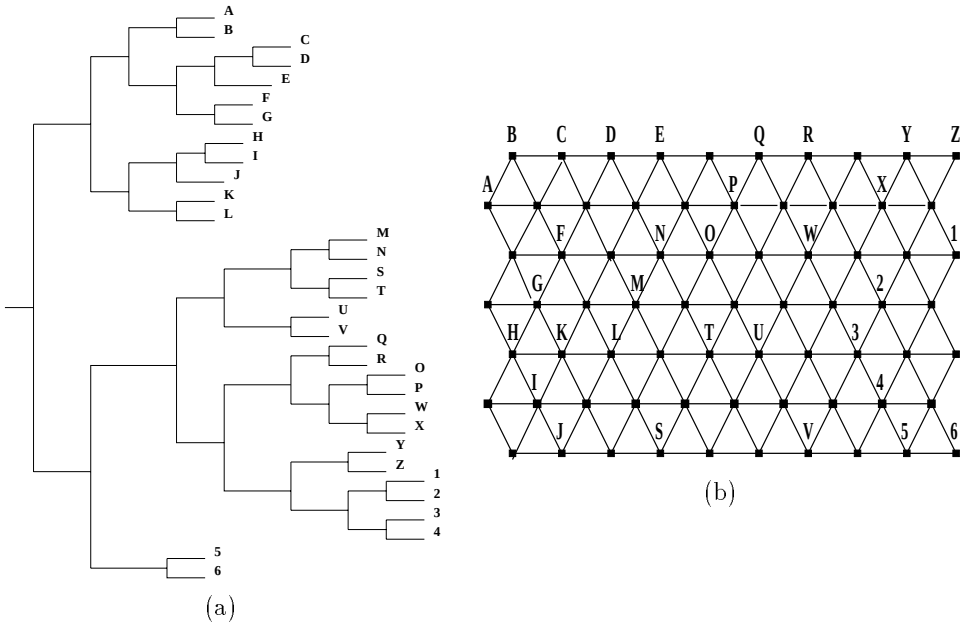


Figure 4.4: **2-D representations of the spanning tree data.** (a) The merge tree derived by the merge clustering algorithm. Cluster boundaries are apparent in the structure, but the global and local topology is not. (b) Map derived by the standard self-organizing algorithm (Kohonen 1990). The map is 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.

Ritter and Kohonen 1989; Schutze 1993; Scholtes 1993).

To demonstrate visualization with IGG, a high-dimensional semantic data set was compiled from the Webster online thesaurus. In this thesaurus, relationships among words are represented explicitly as lists, where each list indicates a different type of relationship. However, only the most direct relations between words are immediately apparent to the user. Less obvious relationships require a search through all the intervening lists. In an effort to visualize the global structure of such semantic data, IGG was applied to a subset of the thesaurus words.

In the online thesaurus, words appear in alphabetical order. Each word is accompanied by its part of speech, definition, and any idiomatic expressions it is used in, along with its cross reference lists. These lists include synonyms, related words, contrasted words, and antonyms. In order to apply IGG, feature vector representations for the words were created based on the cross-reference lists. The features are words, and each main entry in the thesaurus is represented by a feature vector indicating the presence or absence of each of the words in its lists. A value of l.O for a feature indicates that word (feature) is present in the related list; a value of -l.O indicates that the feature appears in the contrasted or

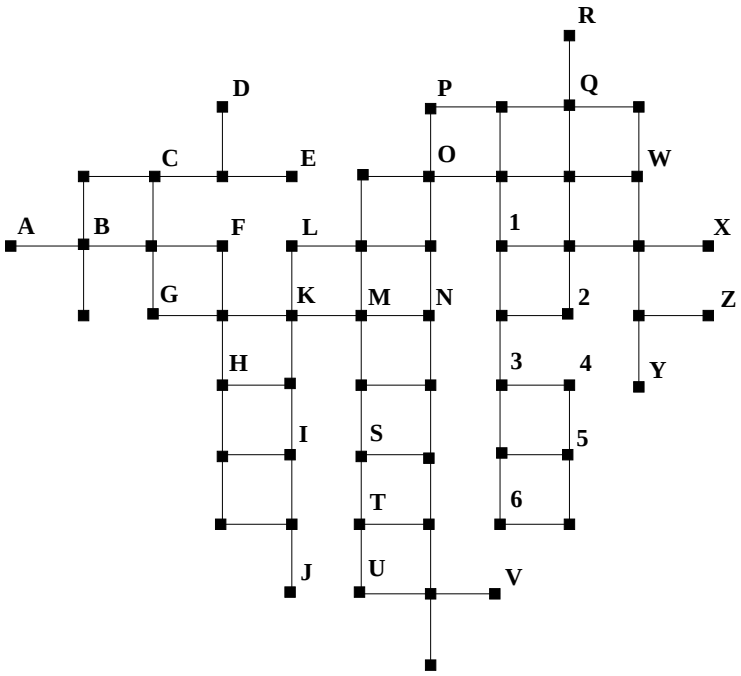


Figure 4.5: **IGG representation of the spanning tree data.** The limited connectivity between clusters in the map closely resembles the structure of the spanning tree. This map was produced using 4 organization phases for each incremental structure. In phase 1, the best matching unit was moved toward the input with an α of 0.18 (α_{bmu}); the neighbors were moved by 0.13 (α_{ngb}); the structure was trained for 25 epochs (nepochs). In phase 2, $\alpha_{\text{bmu}} = 0.14$, $\alpha_{\text{ngb}} = 0.1$, nepochs = 15. In phase 3, $\alpha_{\text{bmu}} = 0.09$, $\alpha_{\text{ngb}} = 0.06$, nepochs = 15. In phase 4, $\alpha_{\text{bmu}} = 0.05$, $\alpha_{\text{ngb}} = 0.0$, and nepochs = 15. The neighborhood size was a function of the number of nodes in the current structure. Where there were as many or fewer than 36 nodes in the map, the neighborhood size began at I and decreased to 0 by phase 4. If there were greater than 36 nodes, the neighborhood began at 2 and decreased to 0. Connections were added or deleted after phase 4: a connection was added between 2 nodes if their distance was less than 2.6 times the average distance in the map (tconnect). A connection was deleted if the distance between the 2 nodes was more than 2.9 times the average distance in the map (tdelete). The parameter settings were determined experimentally, but serve as good starting points for other IGG experiments.

antonym lists. A value of O.O indicates that the two words are not directly related by their lists.

For brevity, the example described here uses only verb entries from the thesaurus. Synonyms have been collapsed into single entries and single features. Also, only those verbs that appear most often are represented in the input set, and only those verbs that are used most often in the cross-reference lists are used as features. The resulting data set contains 197 verbs, each represented by 240 features. Because the data source is obtained from the real world, and because a very simple method for reducing its size was employed, the data

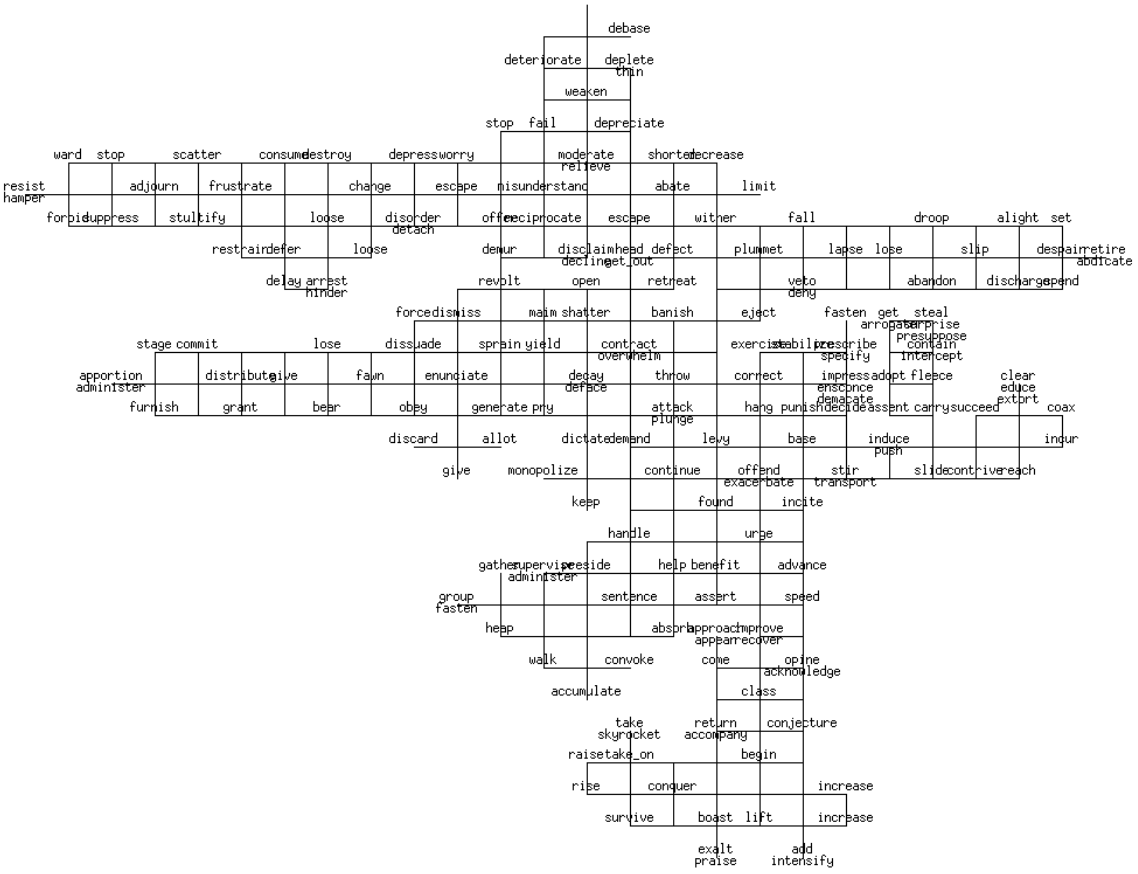


Figure 4.6: **The reduced Webster's thesaurus verb set.** The full verb set contains 1477 1477-dimensional vectors; here it has been reduced to 197 240-dimensional vectors for clarity. (a) A sample of the online thesaurus text. Only direct relations between words are explicitly represented, and looser relationships must be deduced by chasing word entries. (b) The IGG grid encodes semantic similarities as well as semantic boundaries. The simulation parameters were the same as in the minimum spanning tree example (figure 4.5), except nepochs for the 4 phases were 10, 6, 6, and 4, tconnect was 2.8 and tdelete was 33.

set is very noisy, and therefore rather difficult to describe and visualize.

The final grid derived by IGG is shown in figure 4.6. Semantic similarities have been captured in the 2-D topology of the grid to the extent possible with this data. Semantic dissimilarities are implicit in the map boundaries. Note that although the vector representation scheme is very simple, the data is not synthetic or contrived. The vectors were generated from the straight text of the thesaurus, and the full data set reduced only for brevity. The grid has captured the relationships to the extent they exist in the reduced set, and has placed the noise words in areas where there are similar vectors.

In the top left arm of the map, verbs that connote a retarding function are clustered

(e.g. *hamper*, *ward*, *suppress*, *stultify*). This arm merges into the top central arm, which contains verbs suggesting deterioration (*deteriorate*, *weaken*, *depreciate*). This arm joins an area that contains words implying lessening (*decrease*, *wither*). This area in turn blends into the top right arm, which contains words connoting a decreasing or removal function (*fall*, *droop*, *slip*, *retire*). The bottom two arms contain words that have more positive connotations. The bottom right contains words suggesting increase (*skyrocket*, *raise*, *increase*). This area merges into an area connoting improvement and promotion (*improve*, *speed*, *help*). To the left, this area blends with an arm connoting accumulation or gathering (*absorb*, *group*, *heap*, *accumulate*). In the central area connecting the more positive arms with the more negative arms, there is a gradation of meaning from bottom to top: *urge*, *incite*, *offend*, *attack*, *overwhelm*, *shatter*, *maim*, *revolt*, *demur*. The gathering region also merges into the negative region: *handle*, *keep*, *monopolize*, *dictate*, *demand*, *overwhelm*.

Two other arms are present in the grid, containing words that do not support the general positive-negative gradation. On the left, words that imply giving (*distribute*, *grant*) blend with words that suggest giving in (*lose*, *fawn*). These in turn merge into the more negative areas of the grid (*sprain*, *yield*, *decay*). On the right, words that connote success and encouragement (*coax*, *succeed*, *reach*, *push*) blend with words connoting support (*base*, *correct*, *stabilize*). This area in turn gradually blends with the positive region (*incite*, *urge*, *assert*).

The thesaurus data demonstrates the ability of IGG to represent complicated high-dimensional data, which is often both incomplete and noisy. Note that thesauruses exist for exactly this reason: the cognitive and intuitive similarities and dissimilarities between words are very hard to define or describe, and are often context dependent. To the extent possible, the network has learned the structure present in the input. The results support the idea that IGG can be used as a visualization tool for very complicated, unknown data. Chapter 5 explores the application of IGG to a real-world task in which visualization is an essential tool for data analysis and understanding.

Chapter 5

Application: human genetics data

The study of human evolution requires the analysis of the complicated relationships between human populations. To understand evolution, one must study the migration patterns of populations. To understand migration, one must examine how the similarities between populations diverge geographically. Similarities between human populations can be judged genetically, linguistically, and culturally. This chapter looks at the problem of discovering the how populations are related genetically. Visualization of high-dimensional genetic data is essential to the analysis of such relationships.

5.1 Studying human genetics

Over the last 30 years, L. Cavalli-Sforza and his colleagues have collected genetic information from native populations in virtually every inhabited area of the planet. The genetic information is derived from blood samples taken from individuals of a population inhabiting a geographic area. Individual blood samples can be tested for the presence or absence of various genes. Thus each individual can be described by the set of genes present in his/her blood.

If a large enough number of individuals is thus described, one can obtain an average description of the whole population. For example, if 50% of the individuals sampled in a population test positive for a certain gene, then the population has a gene frequency of O.5 for that gene. A whole population can be described by a set of gene frequencies in the same way that an individual can be described by a set of genes. If one describes a set of M populations with a set of N gene frequencies, then one obtains an N-dimensional data set of M populations. In order to discover the genetic relationships between these populations, one must visualize this data set.

5.2 The data set

Cavalli-Sforza et al. (Cavalli-Sforza, Menozzi and Piazza 1994) tested thousands of populations for hundreds of genes over 30 years. The sheer volume of the data is clearly an obstacle to analysis. Therefore, in order to discover the most general global trends, they first reduced the amount of data to a manageable level. They first averaged the gene frequency vectors of many populations chosen for linguistic, cultural and geographical similarity. In this way, they reduced the data set to 42 populations "aggregates". In addition, they reduced the dimensionality of the data by considering only those genes for which a majority of the populations had appreciable frequencies. The resulting data set consists of 42 population aggregates, each described by 120 gene frequencies. In figure 5.1, the populations and their general geographic locations are indicated.

Cavalli-Sforza et al. analyzed the 42 population aggregates and their relationships using standard visualization techniques, including merge clustering (figure 5.2) and principal component analysis (figure 5.3). Note that visualization, like all data analysis, depends on the method for calculating the distance between vectors. The Euclidean distance is one such measure. The distance metric they used is similar to Euclidean distance, but it is not the same. The F_{st} measure is designed specifically to measure genetic distances between populations. For instance, the calculation involves a form of normalization to account for the fact that gene frequencies are not normally distributed. Also, special terms are added to correct for sampling error when the sample sizes are small. In the cases where populations are missing information about certain genes, these genes are left out of the distance calculation. Thus this measure is a much more accurate description of the genetic distance between two populations than is the Euclidean distance. This is the measure that Cavalli-Sforza et al. used to obtain the merge tree in figure 5.2 and the PCA map in figure 5.3.

5.3 Visualizing the 42 populations

Figure 5.2 shows the merge cluster tree for the 42 populations using the F_{st} distance. Examining the tree top to bottom, one first finds a cluster for African populations (*San*, *Mbuti*, *Bantu*, *Nilotic*, *W. African*, *Ethiopian*). This is followed by a cluster than contains mostly European groups (*Lapp*, *Sardinian*, *Greek*, *Basque*, *Italian*, *Danish*, *English*), plus two Indian (*S.E. Indian*, *Indian*), one Persian (*Iranian*) and one African group (*Berber*). This is followed by a cluster consisting mostly of Asian and North Asian groups (*Samoyed*, *Mongol*, *Tibetan*, *Korean*, *Japanese*, *Ainu*, *N. Turkic*, *Chukchi*) with one American group (*Eskimo*). This is followed by a cluster of American groups (*S. Amerind*, *C. Amerind*, *N. Amerind*, *N.W. American*). The next cluster contains mostly Pacific island populations (*Indonesian*, *Philippine*, *Malaysian*, *Polynesian*, *Micronesian*, *Melanesian*) along with a

Population name	Geographic location
Bantu	South and Central Africa
EastAfrican	East Africa
NiloSaharan	Central Africa
WestAfrican	West Africa
San	Northeast Africa (Bushmen)
Berber	North Africa
Mbuti	Central Africa (Pygmies)
Indian	India
Iranian	Persia
NearEast	East Europe, West Russia
Uralic	North Asia, Northeast Europe
Ainu	East Asia
Japanese	Japan
Korean	Korea
MonKhmer	Vietnam
Thai	Thailand
Dravidian	Southeastern India
MongolTungus	Mongolia
Tibetan	Tibet
Indonesian	Indonesia
Filipino	Philippines
NorthTurkic	North Turkey
SouthChinese	South China
Basque	France and Spain
Lapp	Northern Scandinavia
Sardinian	Sardinia
Dane	North Central Europe
English	United Kingdom
Greek	Greek Islands
Italian	Italy
CentralAmerind	Central America
Eskimo	Alaska, Aleutian Islands
NaDene	Western North America
NorthAmerican	Central and Eastern North America
SouthAmerican	South America
Chukchi	Extreme Northeastern Asia
Melanesian	Islands northeast of Australia
Micronesian	Islands west of the Philippines
Polynesian	Islands northeast of New Zealand
Malay	New Guinea
NewGuinean	Australia
Australian	Malaysia

Figure 5.1: **The 42 population aggregates and general geographic regions.**

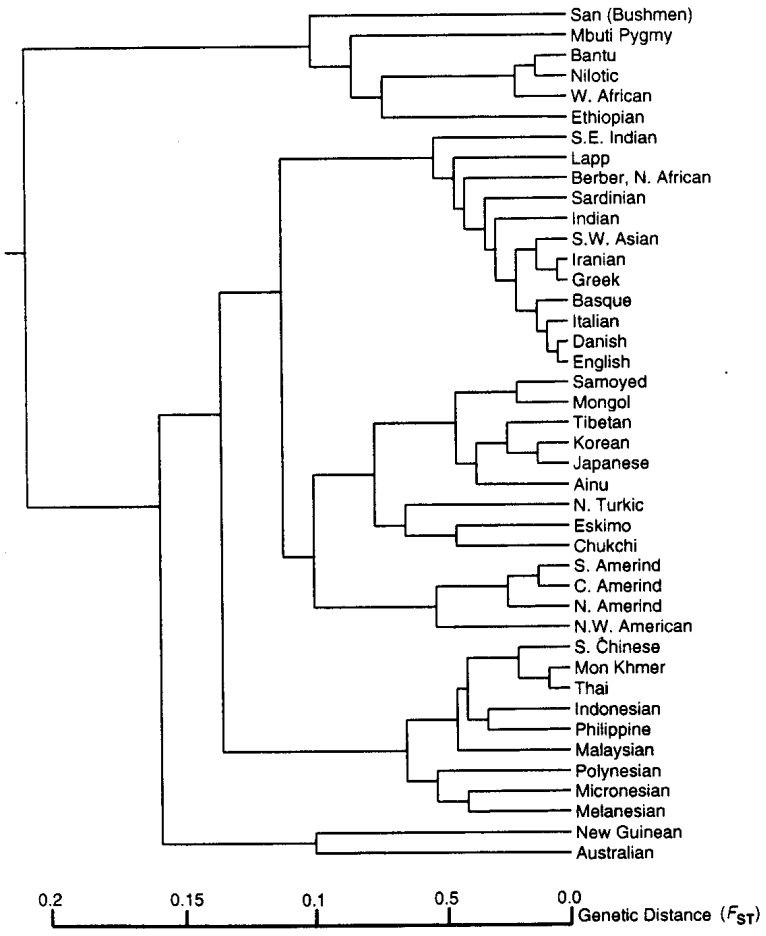


Figure 5.2: **Merge tree for the 42 populations using F_{st} distance.** The clusters are mainly geographic in nature. From the top to bottom, one sees clusters for African, European, Non h Asian and Asian, American, and Pacific island populations. Note that some populations end up mixed into unlikely clusters. For example, the Indian, Iranian and Berber populations are mixed into the European cluster. (From Cavalli-Sforza et al. 1994)

sub-cluster of Asian groups (*S. Chinese*, *Mon Khmer*, *Thai*). The final cluster consists of more Pacific island populations (*New Guinean*, *Australian*).

Figure 5.3 displays the PCA plot for the same data. For the most part, the PCA plot mirrors the organization of the merge tree. Note that in the upper left region, North Asian populations (*Mongol Tungus*, *Tibetan*, *Uralic*, *N. Turkic*) are mixed with American populations (*N. American*, *NaDene*, *C. Amerind*, *S. American*) and some Asian populations (*Japanese*, *Korean*, *Ainu*). These populations are not individually clustered, as they tend to be in the merge tree. Note also that the PCA plot seems to indicate a closer relationship

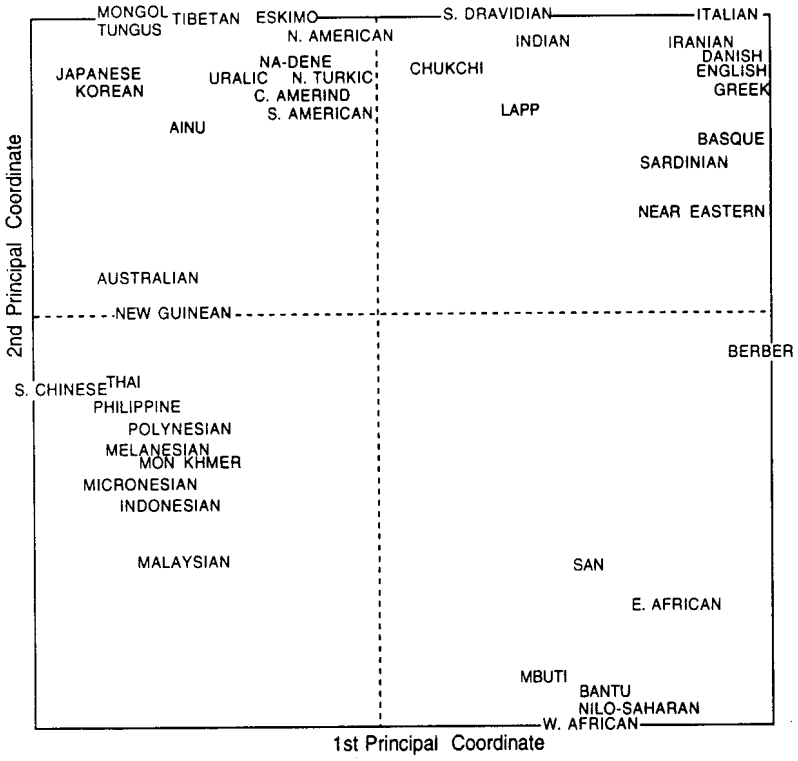


Figure 5.3: **PCA map of the 42 populations using F_{st} distance.** Here again the clustering follows mainly geographic boundaries. Note that the Indian populations (*Indian, S. Dravidian*) are not mixed into the European population as they are in the merge tree. Also note that there is little structure to the populations in the upper right. (From Cavalli-Sforza et al. 1994)

between the two Indian groups (*Indian, S. Dravidian*).

Using the merge tree and the PCA plots, along with other non-visual data analysis tools, they divide the 42 populations into nine clusters. After averaging the members of each cluster, they derive a summary merge tree that describes the overall structure of the 42 population data set (figure 5.4). The summary tree makes it clear that the genetic data is organized along mainly geographic lines.

In this application, it is clear that both merge clustering and PCA capture aspects of the data that the other does not. Because IGG is designed to combine the best properties of both techniques, the application is ideally suited to demonstrating IGG's potential. Also, Cavalli-Sforza notes that it would be instructive to have a method that allows links between branches of the cluster trees. IGG is such a method.

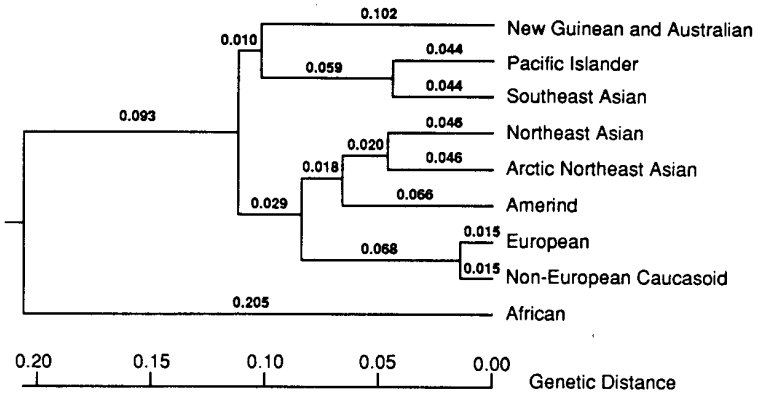


Figure 5.4: **Summary tree.** This tree obtained by averaging the 42 populations into nine cluster summarizes the structure of the data set. (From Cavalli-Sforza 1994)

5.4 Applying IGG to the task

Unfortunately, our best understanding of the F_{st} calculation has failed to reproduce the distance values published in the book, and we have been unsuccessful at gaining more insight through private communication with Cavalli-Sforza and his colleagues. For the demonstration and comparison for this thesis, the standard Euclidean distance was used for all algorithms. The data was normalized such that the variance for all frequencies is unity, which prevents any single gene frequency dominating the distance. Note that because a different metric has been used, the results presented here cannot be compared directly with the results presented in the book.

Using the Euclidean distances, both merge cluster and principal component analysis were applied to the data (figures 5.5 and 5.6). The IGG map of the data is shown in figure 5.7. It is apparent that the IGG representation incorporates the dominant relations present in the merge cluster and PCA representations in the same way that figure 5.4, defined manually, summarizes information in the corresponding F_{st} visualizations.

In the merge cluster representation (figure 5.5), several main trends are evident. Starting at the bottom right, one sees clustering for European populations (*English*, *Dane*, *Italian*, *Greek*, *Basque*), Asian populations (*Thai*, *SouthChinese*, *MonKhmer*, *Korean*, *Tibetan*, *Filipino*, *Japanese*), and Pacific island populations (*NewGuinean*, *Melanesian*, *Micronesian*, *Malay*). In addition, the upper part of the tree has a general trend toward the African populations (*San*, *Mbuti*, *Berber*, *East African*), mixed with North Asian populations (*Lapp*, *Chukchi*, *Ainu*, *MongolTungus*, *Uralic*, *NorthTurkic*). However, the upper part of the map does not have a very clear cluster structure, and has several misrepresentations. For instance, *Sardinian* and *Lapp* are both European populations, but they are mixed in

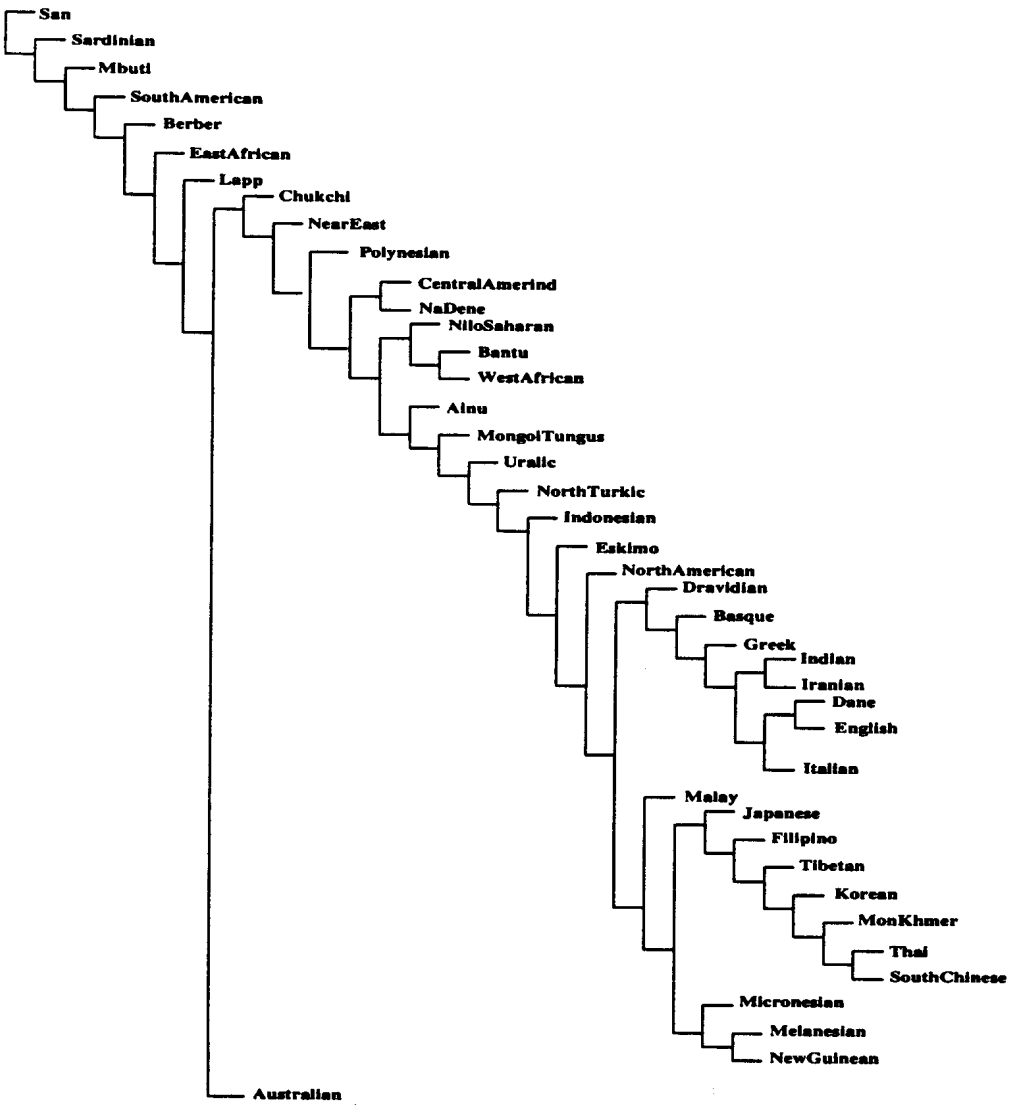


Figure 5.5: **The merge tree for the 42 populations using Euclidean distance.** How closely the populations are related is indicated by how close their merge points are. Several recognizable groups are present. At the lower right, some European populations are merged together (*Dane*, *English*, *Italian*, *Creek*, *Basque*). Below this group, Asian populations are clustered (*Japanese*, *Tibetan*, *Korean*, *MonKhmer*, *Thai*, *SouthChinese*). Below that, some Pacific islands are represented (*Micronesian*, *Melanesian*, *NewGuinean*). All of these groups are missing some members, which appear merged nearby unrelated populations. Note also, the African, North Asian and American populations are not clearly recognizable.

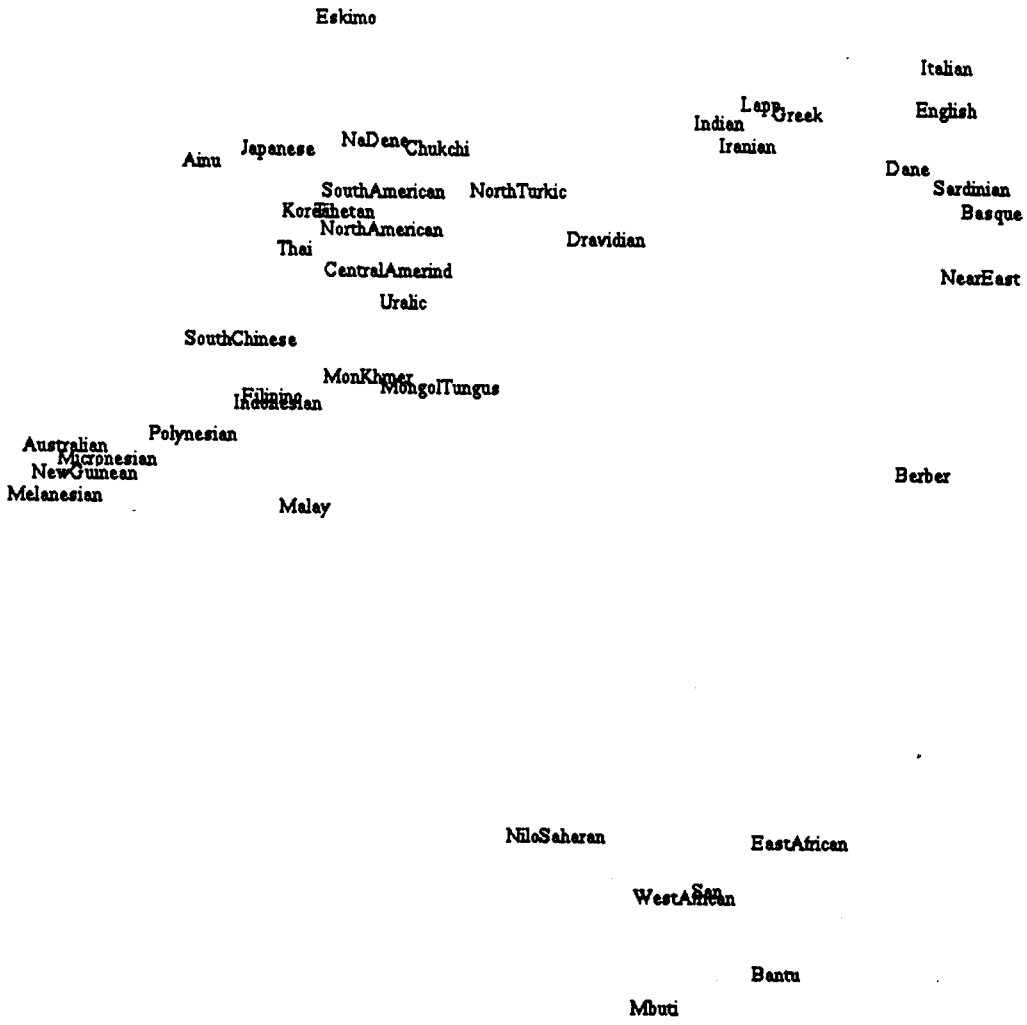


Figure 5.6: **The PCA map for the 42 populations using Euclidean distance.** The clustering structure is a little more apparent than in the merge cluster tree. The European populations are clustered together in the upper right (*Italian*, *English*, *Dane*, *Sardinian*, *Basque*). The African populations are clustered in the lower region (*NiloSaharan*, *EastAfrican*, *San*, *WestAfrican*, *Bantu*, *Mbuti*. The Pacific populations are at left (*Australian*, *Micronesian*, *NewGuinean*, *Melanesian*, *Polynesian*, *Malay*, *Filipino*, *Indonesian*). Unfortunately, the Asian, North Asian and American groups are mixed together in the top left central region.

with the first few African populations. In addition, an African cluster (*Bantu*, *WestAfrican*, *NiloSaharan*) merges into the tree near the North Asian area (*Ainu*, *MongolTungus*, *Uralic*, *NorthTurkic*), but far from the first African area. Polynesian and Indonesian are both Pacific island populations, but both appear in the map separated from each other and from the other Pacific groups. Australian is also clearly a Pacific group, although it joins the merge tree far from any geographic relatives.

In the PCA representation (figure 5.6), the clustering is a little better. The European populations are close in the upper right (*Italian*, *English*, *Dane*, *Sardinian*, *Basque*, *NearEast*, *Lapp*, *Greek*). The African populations are clustered in the lower right (*NiloSaharan*, *EastAfrican*, *San*, *WestAfrican*, *Bantu*, *Mbuti*). The Pacific groups are cluster on the left (*Australian*, *Micronesian*, *NewGuinean*, *Melanesian*, *Polynesian*, *Malay*, *Filipino*, *Indonesian*). The North Asian and Asian groups appear mixed in the top region, just left of center (*Chukchi*, *NorthTurkic*, *Japanese*, *Korean*, *Tibetan*, *Thai*, *Uralic*, *SouthChinese*, *MonKhmer*, *MongolTungus*). As in the merge cluster tree, there are some misfit populations mixed into the general clusters. The Indian, Iranian and Dravidian populations occur near the European populations. This also occurs in the merge cluster tree, so it is likely that this is simply the best place for the populations. Also, the American populations (*SouthAmerican*, *NorthAmerican*, *CentralAmerind*, *NaDene*, *Eskimo*) populations are mixed into the middle of the Asian-North Asian group. This is different from the merge cluster representation, so it is unclear where these groups really belong in the general scheme of things. The PCA representation seems more clear than the cluster tree; however, one should be careful and note that only the first two principal components are included, and therefore the graph does not incorporate all of the information in the data into the 2-D representation.

The IGG map (figure 5.7) combines the best of both worlds. Clusters are clearly present, as well as the global topology of the data set. The European populations occupy the top arm, and blend, through the Iranian, Indian and Dravidian populations to the North Asian populations in the center of the map (*MongolTungus*, *NorthTurkic*, *Chukchi*, *Uralic*, *Tibetan*). This suggests that these "misfits" are genetically somewhere between European and North Asian populations. The North Asian populations, in turn, blend into the Asian populations in the lower center portion of the map (*MonKhmer*, *Thai*, *Korean*, *Ainu*, *Japanese*, *SouthChinese*). The lower Asian groups are nearby the Pacific groups in the lower arm (*Filipino*, *Malay*, *Indonesian*, *Micronesian*, *Polynesian*, *Melanesian*, *NewGuinean*, *Australian*). The North Asian groups also blend into the right arm, which contains the American groups (*NorthAmerican*, *Eskimo*, *SouthAmerican*, *CentralAmerind*, *NaDene*). All the major geographic clusters (*European*, *North Asian*, *Asian*, *African*, *Pacific*, *American*) are present in the IGG map, and the misfit populations now seem to fit into the overall topology. It therefore improves upon both the cluster tree and the PCA plot.

The human genetics application is ideally suited for studying visualization algo-

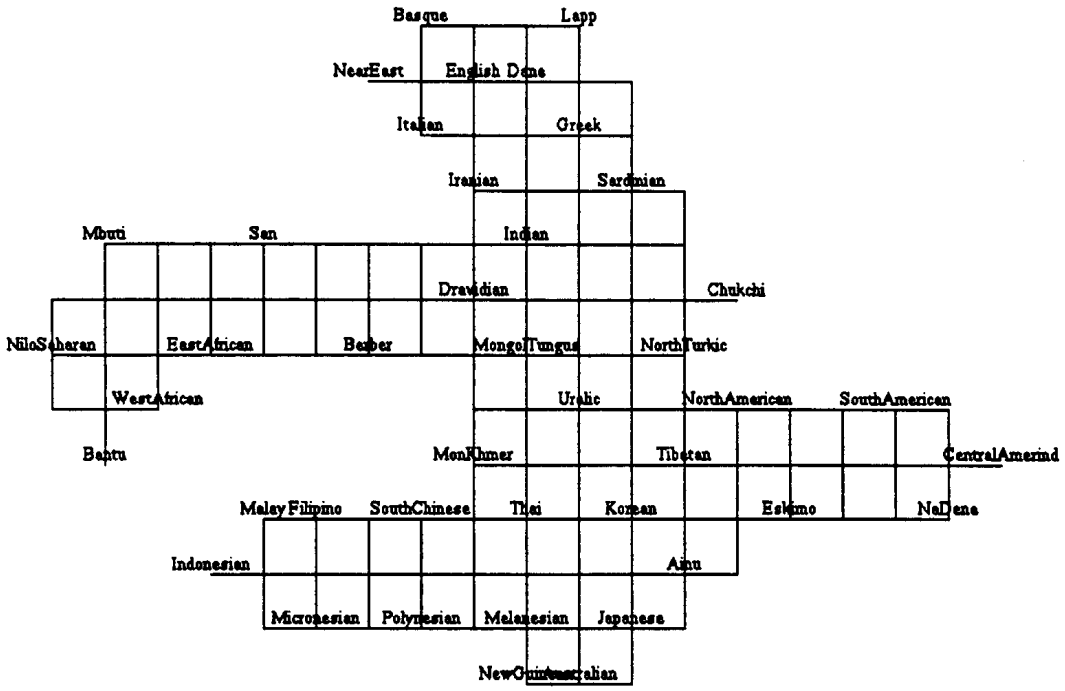


Figure 5.7: **The IGG network for the 42 populations using Euclidean distance.** The structure of the network reflects the structure of the data. The main geographic clusters of the data are present in the arms and central region of the map. The top arm contain the European populations (*Basque*, *Lapp*, *NearEast*, *English*, *Dane*, *Italian*, *Greek*, *Sardinian*). The left arm has the African groups (*Mbuti*, *San*, *NiloSaharan*, *Bantu*, *WestAfrican*, *EastAfrican*, *Berber*), The lower arm contains the Pacific groups (*Australian*, *NewGuinean*, *Melanesian*, *Polynesian*, *Micronesian*, *Indonesian*, *Malay*, *Filipino*). The American populations are in the left arm (*NorthAmerican*, *SouthAmerican*, *CentralAmerind*, *NaDene*, *Eskimo*). The central region contains the North Asian populations in the upper-right central area (*Chukchi*, *MongolTungus*, *NorthTurkic*, *Uralic*, *Tibetan*), and the Asian populations in the lower central area (*MonKhmer*, *Thai*, *Korean*, *SouthChinese*, *Ainu*, *Japanese*). Each of the arms blends into the Asian regions in a reasonable way: the northern African (*Berber*) blends into more southern Asia areas and northern American (*NorthAmerican*, *Eskimo*) populations blend into the North Asian area. The Pacific populations blend into the Asian populations. As in both the merge cluster and PCA representations, the misfit Indian/Persian populations (*Iranian*, *Indian*, *Dravidian*) seem to be genetically between the North Asian groups and the European groups. The fact that this occurs for all the representations suggests that it is a proper feature. Thus IGG is able to capture all the most important information present in both the merge tree and PCA.

rithms. The data is very high-dimensional and noisy, and the individual data items are interrelated in complex ways. Also, human populations can be related linguistically and culturally as well as genetically. Visualizing data from each of these domains can yield important insight into how genetic interchange differs from linguistic and cultural interchange. Preliminary results seem to indicate that IGG performs well on this data. Further work with the full population genetics data set, along with linguistic data for the same populations, should not only yield insight into this domain, but also help to expose the strengths and weaknesses of IGG.

Chapter 6

Discussion and future work

The quality of the final organization of the map can be defined as the degree to which the map captures the input topology. Maps are considered qualitatively similar when they capture the same topological properties of the input. Given this definition of quality, there are three important questions to be asked:

1. How do the algorithm's parameters effect the final quality of the map?
2. Is there a way to quantify this definition of map quality?
3. Is there a way to automatically adjust the parameters based on monitoring the map quality?

6.1 IGG parameters

The original SOM algorithm forms the core of the IGG algorithm. However, the IGG organizational strategy can be varied along a number of lines. IGG includes, of course, all the basic parameters associated with SOM—the number of epochs to train for, learning rates, neighborhood size and shape and a schedule for changing them. In addition, IGG must include parameters for the total number of nodes to be included, as well as the disconnect and reconnect thresholds. Also, the incremental nature of IGG supports restricting for the total number of nodes to be included, as well as the disconnect and reconnect thresholds. Also, the incremental nature of IGG supports restricting the search for the best matching weight vector to a small neighborhood around the winner during the last epoch to speed up computation. Thus the neighborhood search size is another parameter to be set. Finally, because the algorithm is incremental and its structure changes every time it undergoes growth, every one of the organization parameters might be changed as a function of the number of nodes in the current structure. It is vital to understand how changing these variables, both individually and in combination, effects the final organization of the map.

6.2 Parameter sensitivity

Without a formal mathematical function defining how the algorithm parameters and variables affect the final state of the map, one can only judge the algorithm's sensitivity to parameters experimentally. The algorithm's sensitivity to a given parameter is defined as the degree to which the quality of the map depends on a certain setting of that parameter. For instance, the standard SOM algorithm will almost always converge to qualitatively the same map so long as the neighborhood function starts large and tends to O, and the learning rates are initialized between O.1 and 0.2 and tend to O.O over the course of organization. Thus SOM is relatively insensitive to its parameter settings, within those constraints. An obvious question to ask is, does IGG exhibit the same sort of insensitivity to parameter settings?

Because IGG is incremental, it does not converge to an asymptotic state as does SOM. For each intermediate structure, IGG's neighborhood and learning rates tend to zero. However, every time the IGG map grows, both global ordering and fine-tuning must occur in order for the new nodes to be incorporated into the map properly. Thus these parameters must be reset to their initial values whenever new nodes are added. As a result, IGG tends to be more sensitive to these parameter settings than is SOM. In addition, the connectivity of the map is constantly changing as connections are added and deleted. Thus IGG tends also to be sensitive to the connection threshold settings.

In addition to the connection thresholds, the IGG algorithm appears to be sensitive to the number of epochs organized for a given neighborhood size. At larger neighborhoods, distortions in the local and global topology are more likely to be introduced, and IGG must constantly return to larger neighborhoods in order to fully incorporate new nodes. If a distortion is introduced during one random organization epoch, it may take tens of epochs to overcome it. But there is no guarantee it will ever be overcome. With no way to tell whether the map seems to be correcting the disorder or making it worse, there is no way to tell when more epochs is too many.

In summary, because the structure and connectivity of the IGG map is always changing, the quality of the map does not converge to a final state, as with SOM. To obtain good quality in the final map, IGG must aim at a constantly moving target. It thus tends to be much more sensitive to the parameters that define how that target moves—i.e. the connection thresholds and the amount of training. Getting good quality maps thus depends on setting these parameters such that no distortions are introduced into the map that can adversely effect the quality of the result.

6.3 IGG distortions

The primary qualitative distortion that must be avoided during IGG evolution is the cluster split. A cluster split occurs when nearby nodes in the map that have become tuned to data items in the same high-dimensional cluster become separated by a node or nodes that have become tuned to data items in a different cluster. When this situation occurs, the cluster may remain split, and the "strain" on the inter-cluster connections will increase, causing these connections to be cut. Once the connections are cut, the map has no chance of recovery. Such splits are difficult to detect if one does not a priori know the cluster structure of the input, and therefore may go uncorrected. If a split is not corrected during organization, the error will be reinforced, connections will be cut, and the map will be pulled apart.

6.4 Avoiding distortions

Is it possible to develop heuristics for setting parameters that will guarantee cluster splits are avoided during organization? The preliminary experimental answer appears to be "no". The incremental nature of IGG seems to exaggerate the effects of any disorderings that occur. In general, getting good quality IGG maps depends on balancing two opposing considerations. On the one hand, the map must have sufficient connectivity so that proper order is developed in the map and splits will have maximum opportunity to be corrected. On the other hand, the algorithm must cut enough connections so that specific cluster structure is built into the map incrementally. Experimentally, it has proven somewhat difficult to balance these considerations. In practice, when the structure of the data set is unknown, it is best to set the disconnection and reconnection thresholds to the lowest values that still ensure that no clusters break free from the map during evolution. (This is a reasonable constraint, since it is reasonable to assume that all items in the data set are related to each other in some way, and therefore must be allowed to influence the global cluster structure of the 2-D representation). Once the map has grown to its full size, connections may be examined again to determine which clusters are truly separable.

In summary, the behavior of the IGG algorithm is controlled primarily through the connection thresholds and the number of epochs parameters, and these must generally be tuned for each separate data set. By tuning these parameters appropriately, one can achieve a faithful 2-D representation of complex high-dimensional data. Selecting appropriate values would be simplified if the quality of the map could be quantified for every structure after every epoch. If there were a measure for how well the current structure captures the high-dimensional order in the data, it would be possible to determine when distortions have been introduced, and whether each epoch was worsening or improving the situation. Such an order measure would not only make it possible for the algorithm to tune its parameters

automatically, but would also allow quantitative comparisons between maps. In the next section, a method for measuring how closely the topology of the map mirrors the topology of the input is proposed. Although this research is still in the experimental phase, the proposed measure of map quality appears promising.

6.5 Measuring quality

Self-organizing maps are topology preserving to the extent that the high dimensional relations in the data are preserved in the 2-D maps "as much as possible". The goal is for the 2-D map to capture the most important relations in the data, such that the closest items in the input space will exhibit small distances on the map. Obviously it may not be possible to preserve all close high-dimensional relationships when the dimensionality of the data set is reduced to two. To what extent then does the 2-D representation capture the topological order of the original data set? How can the order preservation be quantified? How can the quality of the mapping be judged objectively?

Kohonen (1987) successfully tackled this issue for the 1-D map case. However, that work has not yet been successfully extended to 2 dimensions and arbitrary input. Several authors have discussed the necessity for developing some kind of "goodness" measure for 2-D maps (Blayo and Demartines l99O, Zrehen 1992). There has been some research into the evolution of easily calculable map quantities (e.g. the input mapping error and standard deviation, and the similarity between map's point density and input's probability distribution). However, the amount of error between each input and the weight vector to which it is mapped does not provide any information about the topological order of the map. A few researchers have proposed quality measures for SOM designed to quantify how well the map captures the input topology. The research in this area is still preliminary. So far, there has been no investigation of how parameters effect map quality as judged by these measures. Also, the measures are not generally applicable to all 2-D representations, so they are not well-suited for comparisons between different representation methods.

An order measure that permits in-depth investigation of the functional relationship between map quality and parameter settings would significantly increase the overall state of knowledge about SOM algorithms. The next section proposes such a measure. The proposed quantity measures the extent to which the input topology is present in the 2-D map, and is similar to the fitness measures used in non-metric multi-dimensional scaling. The proposed measure has the potential to:

1. indicate when a distorted relation exists in the map,

2. provide a flexible constraint for tuning organization parameters, and

3. allow direct quantitative comparisons between different 2-D representations of high-dimensional data.

6.6 Ranking relationships

Each data item in the input space stands in some relationship to every other input item. The similarity between any two items is measured by calculating some distance (usually Euclidean) between the input vectors representing the items. Items whose distance is smaller are judged to be more similar to each other than are items whose distance is larger. For each individual data item, one can calculate how close it is to every other item. Obviously, it will be closer to some items and further from others. If one orders the distances from smallest to largest, one can see immediately which vectors the data item is closest to, and which vectors are further away. Using this partially ordered set of distances one can rank the item's relation to every other item. Vectors with a ranking of 1, for instance, are the data item's closest relatives. Vectors with a ranking of 2 are very close relatives of the data item indeed, but not quite so close as rank 1 relatives, and so on. The goal of the self-organizing map is to represent the relations in the data whenever possible. The degree to which it captures the rankings between every data pair is a good indication of the quality of the map.

To determine the degree to which the rankings are present in the map, one needs only to calculate the pairwise distances in the 2-D map space. That is, for each node whose weight vector represents an input item, calculate the distance on the map to every other node whose weight vector represents an input item. Given the set of distances in 2-D space, then for each data item one can order its distances to every other item. Again, one can use this partially ordered set to determine how each data item is related to every other data item. One can rank the 2-D relationships for each data item and compare the result with the high-dimensional rankings. The degree to which the 2-D map has captured the pairwise rankings is a measure of the topological order of the map.

6.7 Rankings as a measure of topological order

In general, one cannot expect a perfectly order preserving mapping from the high-dimensional input space to two dimensions. Therefore, one must quantify the error induced in the mapping. One way the global error might be indicated is by counting the misrankings that occur in the 2-D map. Let a rank i inclusion error be an instance when for some input item, the 2-D map gives an i ranking to an element which does not have such a ranking in the input space. Similarly, let a rank i exclusion error be an instance when for some input item, the 2-D map does not give an i ranking to an element which has such a ranking in the

input. By counting the number of inclusion and exclusion errors at each rank, one obtains an estimate of how well the map is representing each relationship rank. Because the lowest order-ranks represent the nearest-neighbor relationships in the data, one would expect the lowest order ranks to contain the fewest errors if the map is capturing the most important relationships in the data.

Such a method for quantifying order preservation has a number of advantages. It can be calculated for any lower-dimensional mapping, including SOM, minimum spanning tree, principal component analysis and multi-dimensional scaling. Thus it can be used to compare the quality of these representations for a given data set. Also, it can be calculated at any point during organization, yielding an estimate on-the-fly of how organization is progressing. It can also be used to examine disorder in the map. When close relatives from the input have a large ranking discrepancy, one can generally assume a distortion that needs investigation has been introduced into the map. In addition, the measure is flexible. One can relax the strictness of the rank relationships, thus allowing the map to have a certain degree of error. For example, one might choose to make no distinction between rank 1 and rank 2 neighbors. In this case, the map is allowed to consider rank 1 and rank 2 neighbors as equals, and does not need to encode their distinction in order to be considered correct.

By examining how the order measure changes during organization, one gets a much clearer picture of how the algorithm behaves. This is essential to determining how parameter settings affect the evolution and final result of the mapping. An in-depth study of how order develops as a function of parameter settings could lead to heuristics that would allow the algorithm to tune its parameters dynamically. Already, a preliminary study of order evolution in the IGG algorithm has led to a heuristic for determining a good point to stop organizing at each phase for each structure. Since the IGG algorithm is particularly sensitive to the number of epochs parameter, this result is quite promising. Automatic parameter tuning is the Holy Grail of all self-organizing methods. The fact that the IGG algorithm can use the order measure to dynamically determine even one of its parameters is significant. It suggests that the proposed measure can be used to enhance all SOM techniques.

6.8 Future work: Automatic parameter tuning

It seems clear that IGG has the potential to exceed the topology-preserving characteristics of other visualization techniques. Quantifying this property with a measure that might help guide the algorithm's organization will enhance IGG's usefulness as a knowledge discovery tool. Consequently, future work on the algorithm will focus on studying how the order measure is effected by different organization strategies and parameter choices. Like any neural network algorithm, IGG would benefit from a method for setting all parameters automatically. An in-depth study of how order evolves in the map will yield information

essential for pursuing automatic parameter tuning. In addition, it will be studied how to use the order measure to improve the final quality of the mapping. For instance, because IGG is incremental, it may be possible to restrict organization to only those areas of the map showing poor order. Similarly, it may be possible to localize parameter settings based on local quality of order in the map. Specifically, future work will investigate whether the measure can be used to:

1. guide dynamic tuning of connection parameters,
2. detect and correct local distortions in the map when they occur,
3. localize organization only to those areas of the map that need better order, and
4. localize the tuning of connection and learning rate parameters based on the local order in the map.

This thesis demonstrates that IGG can extract and represent important relationships in high-dimensional data given the right set of organization parameters. These organization parameters were determined experimentally for all of the examples presented here. Having established the algorithm's potential as a visualization tool, the next step is to automate parameter selection. The above research is designed to achieve this goal.

Chapter 7

Conclusion

The research reported in this thesis has concentrated on developing an algorithm based on Kohonen's self-organizing map specifically to be used for the task of visualization. The Incremental Grid Growing algorithm (IGG) automatically embeds cluster boundaries into a regular, 2-D topology-preserving network using an incremental, self-organizing approach. The result is a 2-D map that represents both the high-dimensional topology of the input space and the cluster boundaries that define the high-dimensional structure. The algorithm thus addresses the shortcomings of popular visualization tools. No single visualization tool is able to capture both the clustering and topology together in a single representation. Merge clustering only extracts clusters, whereas principal component analysis and the self-organizing map only represent topology. Since discovering patterns in the data requires both types of knowledge, analysts must use some combination of these techniques. IGG is able to combine both types of information into a single 2-D representation. Consequently, it is a promising new tool for discovering relationships in unknown, complex real-world data sets.

The results on the example data sets demonstrate how well IGG extracts information about both topology and clustering in high-dimensional input spaces. The minimum spanning tree data set illustrates that the algorithm can capture graph-like structures. The thesaurus example demonstrates how IGG can extract both clusters and topology from an extremely noisy, real-world semantic data set. The human genetics application shows that IGG is able to combine the best properties of two standard visualization techniques in a single 2-D representation. IGG captures the cluster structure of merge trees in the final structure of the network. In addition, the topology of the IGG map generally contains all of the topological properties captured in a PCA map. All of these results suggest that IGG is well-suited to the task of visualizing complex, high-dimensional data.

This thesis has also presented a promising new method for measuring how well visualization techniques represent the high-dimensional structure of the data. The degree

of topology-preservation in the representation is defined as the degree to which the map captures the ranked pairwise relationships between all of the input items. The search for the best possible mapping can clearly benefit from quantifying order-preservation. Further, this order measure may prove useful in the task of automatic parameter tuning for self-organizing algorithms. For instance, using the proposed measure, the time evolution of order in IGG maps was examined. Analysis of the behavior of the order measure suggested a heuristic rule for automatically determining one of IGG's parameters. Preliminary results indicate that this heuristic is indeed a useful guide to organization. This is a significant result, and future work will focus on extending the experiments to other parameters of the algorithm.

This research will continue to concentrate on discovering the unknown structure of large high-dimensional data sets. Specifically, the full genetics data set from Cavalli-Sforza consisting of 491 populations described by 128 gene frequencies, will be investigated. IGG will also be applied to high-dimensional linguistics data for the same populations (Nichols 1992) for comparison. Clearly, the relationships and interactions between human populations are complex. Visualizing the relationships within and between genetic and linguistic clusters may aid in understanding the evolution and migration of populations. Incremental Grid Growing was designed with just this kind of application in mind.

Bibliography

Bauer, H., Pawelzik, K. and Geisel, T. (1992). A Topographic Product for the Optimization of Self-Organizing Feature Maps. In Moody, J. E., Hanson, J. and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems 4*, 1141–1147. San Mateo, CA: Morgan Kaufmann.

Blayo, F. and Demartines, P. (1991). Data analysis: How to compare Kohonen neural networks to other techniques? In Brieto, A. editor, *Lecture Notes in Computer Sciences 540: Artificial Neural Networks (IWANN 1991)*. Berlin; Heidelberg; New York: Springer.

Bouton, C. and Pages, G. (1993). Self-organization of the one-dimensional Kohonen algorithm with non-uniformly distributed stimuli. *Stochastic Processes and their Applications* 47:249–274.

Cavalli-Sforza, L. L, Piazza, A. (1993). Human Genomic Diversity in Europe: A Summary of Recent Research and Prospects for the Future. *European Journal of Human Genetics*, 1:3–18.

Cavalli-Sforza, L. L, Menozzi, P., and Piazza, A. (1994). *The History and Geography of Human Genes*. Princeton, NJ: Princeton University Press.

Cottrell, M., and Fort, J. C. (1987). Etude d'un algorithme d'auto-organisation. *Ann. Inst. Henri Poincare*, 23:1-20.

Cottrell, M., Fort, J. C., and Pages, G. (1992). Two or three things that we know about the Kohonen Algorithm.

Demartines, P. (1992). Organization measures and representations of the Kohonen maps. In Herault, J., editor, *Proceedings of the First IFIP Working Group* (Grenoble, France).

Fritzke, B. (199la). 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. (199lb). 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.

Fritzke, B. (1992). *Wachsende Zellstrukturen—ein selbstorganisierendes neuronales Netzwerkmodell.* PhD thesis, Technischen Fakultat, Universitat Erlangen-Nurnberg, Erlangen, Germany.

Hertz, J., Krogh, A. and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation.* Redwood City, CA: Addison-Wesley.

Jockusch, S. (1990). A neural network which adapts its structure to a given set of patterns. In Eckmiller, R., Hartmann, G. and Hauske, G., editors, *Parallel Processing in Neural Systems and Computers*, 169–172. Amsterdam; New York: North-Holland.

Jockusch, S. and Ritter, H. (1994). Self-organizing Maps: Local Competition and Evolutionary Optimization. *Neural Networks* 7:1229–1240.

Kangas, J., Kohonen, T. and Laaksonen, J. (1990). Variants of self-organizing maps. *IEEE Transactions on Neural Networks*, 1:93–99.

Kohonen, T. (1982a). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69.

Kohonen, T. (1982b). Analysis of a simple self-organizing process. *Biological Cybernetics*, 44:135–140.

Kohonen, T. (1984). *Self-organization and Associative Memory.* Berlin; Heidelberg; New York: Springer.

Kohonen, T. (1989). *Self-organization and Associative Memory.* Berlin; Heidelberg; New York: Springer. Third edition.

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

Lo, Z., and Bavarian, B. (1991). On the rate of convergence in topology preserving neural networks. *Biological Cybernetics*, 65:55–63.

Lo, Z., Fujita, M. and Bavarian, B. (1991). Analysis of Neighborhood Interaction in Kohonen Neural Networks. In *Proceedings of the Fifth International Parallel Processing Symposium* (Los Alamitos, CA), 246–249.

Lo, Z. Yu, Y. and Bavarian, B. (1993). Analysis of the convergence properties of topology preserving neural networks. *IEEE Transactions on Neural Networks*, 4207–220.

Martinetz, T. and Schulten, K. J. (1991). A "neural gas" network learns topologies. In *Proceedings of the International Conference on Artificial Neural Networks* (Espoo, Finland), 197402. Amsterdam; New York: North-Holland.

Martinetz, T. and Schulten, K. J. (1994). Topology Representing Networks. *Neural Networks* 7:507–522.

Miikkulainen, R. (1993). *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon and Memory.* Cambridge, MA: MIT Press.

Nichols, J. (1992). *Linguistic Diversity in Space and Time.* Chicago, IL: University of Chicago Press.

Ritter, H. J. (1991). Learning with the self-organizing map. In *Proceedings of the International Conference on Artificial Neural Networks* (Espoo, Finland), 379–384. Amsterdam; New York:North-Holland.

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

Ritter, H. J., and Schulten, K.J. (1988). Convergence properties of Kohonen's topology conserving maps: Fluctuations, Stability and Dimension Selection. *Biological Cybernetics*, 60:59–71.

Rodriques, J.S., and Almeida, L.B. (1990). Improving the learning speed in topological maps of patterns. In *Proceedings of the International Neural Networks Conference* (Paris, France), 813–816. Dordrecht; Boston: Kluwer.

Scholtes, J.C (1993). *Neural Networks in Natural Language Processing and Information Retrieval.* PhD thesis, Universiteit van Amsterdam, Amsterdam, the Netherlands.

Schutze, H. (1993). Word Space. In Giles, C. L., Hanson, S. J., and Cowan, J. D., editors, *Advances in Neural Information Processing Systems 5.* San Mateo, CA: Morgan Kaufmann.

Xu, L., and Oja, E. (1990). Adding top-down expectation into the learning procedure of self-organizing maps. In *Proceedings of the International Joint Conference on Neural Networks* (Washington, DC), vol. II, 531–534. Hillsdale, NJ: Erlbaum.

Zrehen, S. and Blayo, F. (1992). A geometric organization measure for Kohonen maps. In *Proceedings of Neuro-Nimes*, 603–610.

Vita

Justine Marie Blackmore was born on January 22, 1964 in Glen Cove, New York, the child of Jacqueline Graham Blackmore and George Glover Blackmore. She obtained a Bachelor of Science degree in Physics, a Bachelor of Science degree in Mathematics, and a Bachelor of Arts degree in Astronomy from the University of Texas at Austin in 1987. After working in scientific academia for several years, she began studying computer science, and entered the CS graduate program at the University of Texas in 1993.

Permanent Address: 4903 Shadyglade
Austin, Texas 78756
USA

This techreport version of the thesis was typeset with LaTeX 2$_\varepsilon$[1] by the supervisor.

[1] LaTeX 2$_\varepsilon$ is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this thesis were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin.