

1

Introduction: What is a Self-Organizing Map?

André Skupin¹ and Pragya Agarwal²

¹ *Department of Geography, San Diego State University, San Diego, CA 92182-4493, USA*

² *Department of Geomatic Engineering, University College London, London WC1E6BT, UK*

1.1 INTRODUCTION

In the quest to understand and address important issues of the modern era, from environmental degradation to economic development, enormous amounts of geographic data are being generated. With the increasing adoption of such technologies as hyper-spectral remote sensing or wireless sensor networks, the growth rate of data volumes continues to rise. Granularity of geographic data is increasing both in geometric space (i.e. more features and finer cell sizes), and in attribute space (i.e. more attributes and finer measurements of attribute values), leaving us with truly n -dimensional data. We are thus increasingly faced with a data-rich environment, in which traditional inference methods are either failing or have become obstacles in the search for geographic structures, relationships, and meaning. With respect to statistical analysis, some problems of traditional approaches, especially regarding spatial autocorrelation, are increasingly being addressed (Fotheringham *et al.*, 2000, 2002; Rogerson, 2001). However, many see the need for a paradigmatic shift in how geographic data are analysed and this push for a new direction is gaining strength, as indicated by the emergence of such disciplinary labels as *geocomputation* (Fischer and Leung, 2001; Longley, 1998; Openshaw and Abrahart, 2000) or *geographic data mining* (Miller and Han, 2001).

It is this direction, characterized by intense computation applied to large data sets, which is explored in this book. Specifically, it addresses a method known as the Kohonen map or self-organizing map (SOM). It may appear odd to devote a complete volume to a single technique. Indeed, most books on GIS are either textbooks giving an introduction to the overall field or are devoted to a particular application domain, like hydrological modelling. However, those books that explicitly address geo-computation or geographic data mining tend to cover a multitude of very heterogeneous methods and are thus not able to explore each approach in great detail. Very few have limited themselves to a more narrowly defined group of related techniques (Openshaw and Openshaw, 1997). Furthermore, the SOM method was not developed by GIScientists and an excellent monograph already exists that is regularly updated (Kohonen, 2001).

This edited volume aims to demonstrate that there is indeed something special about this method, something that makes it curiously attractive to diverse and sometimes conflicting interests and approaches in GIScience. Those interested in clustering and classification will recognize in it elements of k -means clustering, but with an explicit representation of topological relationships between clusters. Anyone accustomed to dealing with n -dimensional data through a transformation and reduction of variables, as in principal components analysis (PCA) or multidimensional scaling, will tend to interpret the SOM method in that light. The predominantly two-dimensional form of most SOMs means that cartographers and others involved in geographic visualization can readily envision its integration within interactive visualization environments. Those struggling to communicate the results of complex computational procedures to policy-makers and the broader public may find SOMs to be uniquely accommodating in many circumstances. This volume intends to provide a common platform for all those facets of current work in GIScience that pertain to use of SOMs. This is what we hope will separate this volume from others that only allow an abbreviated discussion of the SOM method as one example of artificial neural networks (ANNs) due to broader scope and limited space.

This chapter is aimed at answering basic questions about what a SOM is, how it is created and used, and how it relates to other techniques that readers may be familiar with. All this is done primarily through plain language explanation and visual illustration, as opposed to formulas and the language of mathematics. Kohonen's monograph cannot be beat in the latter regard and is highly recommended to anyone wanting to delve deeper into the inner workings of a SOM (Kohonen, 2001). This chapter also discusses important questions about the relationship between GIScience and the SOM method and finally provides an overview of the other chapters in this book.

1.2 RELATED METHODS

The SOM is part of a large group of techniques known as *artificial neural networks* (ANNs). These have a reputation for performing surprisingly well, while providing little explanation for how results are exactly arrived at. In fact, ANNs are often seen as black-box operations. However, at least in the case of the SOM method, the actual algorithms can be surprisingly simple and the process of self-organization is not beyond comprehension. One quickly realizes that, apart from seeing the SOM only in the context of other ANN methods, depending on its purpose and training parameters one could also interpret it primarily as a *clustering* or *dimensionality reduction* technique. In fact,

the SOM is an ANN method that always performs both clustering *and* dimensionality reduction. The separation invoked in this section is designed to more clearly convey the position of the SOM method in relation to standard statistical and geocomputational approaches.

1.2.1 Artificial Neural Networks

First, it is important to note that ANNs, also known as computational neural networks (CNNs) (Fischer, 2001), are by no means simulations of biological neural networks. At best, one could say that the original idea behind neural computing drew inspiration from biological counterparts, and that most actual implementations are far removed from that inspirational source. What artificial and biological neural networks have in common is that information is not stored in any single location, but rather in a parallel, distributed form, and that certain mechanisms exist in which new information can be ‘learned’ through changes that potentially affect large portions of the network (i.e. learning rules).

The general structure of an ANN consists of a set of *input nodes* and a set of *output nodes*. Alternatively, these nodes are also known as neurons, processing elements, or computational units (Fischer, 2001). Multivariate data presented to input nodes gets processed such that output nodes are activated according to weights that are associated with each incoming link. Neural network training is largely concerned with setting these weights. To do this, many neural networks contain one or more layers of hidden nodes (Figure 1.1). During training, the weights of incoming connections to these nodes are

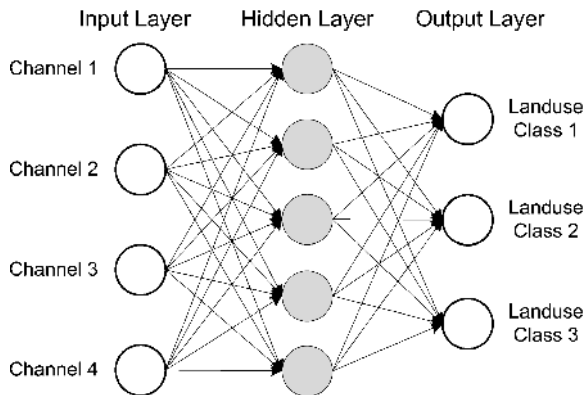


Figure 1.1 Supervised, feed-forward, neural network trained with multispectral remote sensing data and known landuse classes

summed up according to a predefined function. Depending on whether its result satisfies a certain threshold function, an outgoing connection can then be activated. The number of hidden layers and type of connections are an important basis for categorizing different ANN types. In addition to the fixed number of layers and fixed network topology found in many neural networks, there are also neuro-evolutionary models, which use genetic algorithms to help shape neural networks during training.

A fundamental distinction can be made between supervised and unsupervised neural networks. In the *supervised* case, input data presented to the network during training

consist of multivariate data with known outcomes or classifications, i.e. input–output pairs. For example, one could train a neural network with land use classes and corresponding multispectral signatures (Figure 1.1). Multispectral data will be presented, in this case, to the input nodes and a land use class is associated with each output node. During training, weights of hidden layers are iteratively adjusted to establish a good fit between multi-spectral values and correct land use classes. After training is complete, new multi-spectral observations can be presented to the input nodes and land use classes predicted. Most awareness of the power of neural networks within GIScience stems from the use of supervised models. When training data are both multivariate and multi-temporal, one can even predict change patterns (Pijanowski *et al.*, 2002). Supervised neural networks have also been used for purposes other than classification. For example, regression models could be constructed, when continuous outputs are available.

In *unsupervised* learning, the input vectors do not correspond to classes known a priori. Output nodes compete for the input vectors on the basis of certain similarity functions and the weights of winning nodes are adjusted according to the weights of respective input nodes. Due to this competitive learning procedure, input nodes that are quite similar are driving adjustments of similar output nodes. At the same time, dissimilarities in the input data become accentuated. All this supports unsupervised learning’s primary role of finding major structures, clusters, and relationships in multivariate data.

In addition to the fundamental distinction discussed above, one can also distinguish neural networks in terms of whether, during training, adjustments made to neuron weights are only fed forward to the following layers or are also having an effect on preceding layers. Accordingly, *feed-forward* and *recurrent* networks are distinguished. Finally, an important concept is that of *back propagation* (Rumelhart and McClelland, 1986), which is used in feed-forward networks and refers to how errors (i.e. differences between known outputs and neural network outputs) are minimized by making adjustments to neuron weights.

Where does the SOM method fall within the overall system of ANNs? The standard SOM algorithm – the most widely known form and used in many popular software packages (e.g. SOM_PAK) – involves an unsupervised neural network with competitive learning and no hidden layers (Figure 1.2). In that traditional form, SOMs have been especially popular for purposes of clustering and visualization. However, there are also

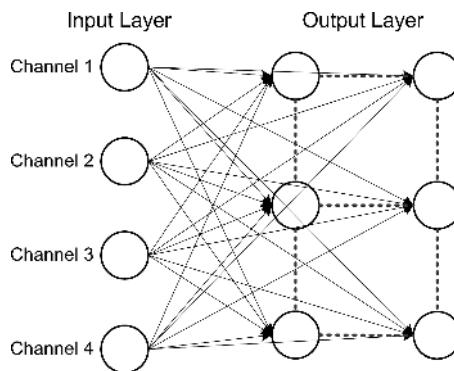


Figure 1.2 Small SOM trained with multispectral remote sensing data

supervised variants useful for classification, including Kohonen’s own *learning vector quantization* (LVQ) (Kohonen, 2001). In this chapter, and for most of this edited volume, the standard SOM algorithm is the focus of discussion. For detailed coverage of other neural networks readers are encouraged to refer to various surveys of this subject (Gurney, 1997; Hertz *et al.*, 1990) as well the growing number of geographically oriented literature (Fischer, 2001; Openshaw and Openshaw, 1997).

1.2.2 Clustering Methods

Discussion of the SOM method in the geographic literature tends to focus on its clustering qualities. The basic idea behind clustering is the attempt to organize objects into groupings based on certain shared characteristics. In spatial clustering, this is typically done in two-dimensional space and thus understood in terms of geometric *proximity*. When applied to feature attributes, clustering may often involve the same Euclidean distance measure, but the results are interpreted as *[dis]similarity*. Clustering involves the search for structures and grouping, and should not be confused with classification, which sorts unknown items into previously defined categories. Since clustering is the most frequent interpretation and implementation of the SOM method, it is useful to compare it to some of the more popular approaches, including hierarchical and *k*-means clustering. In this chapter, the three methods are juxtaposed after being applied to a data set of 32 attributes (mostly derived from population census data) for 50 US States and the District of Columbia (Figure 1.3).

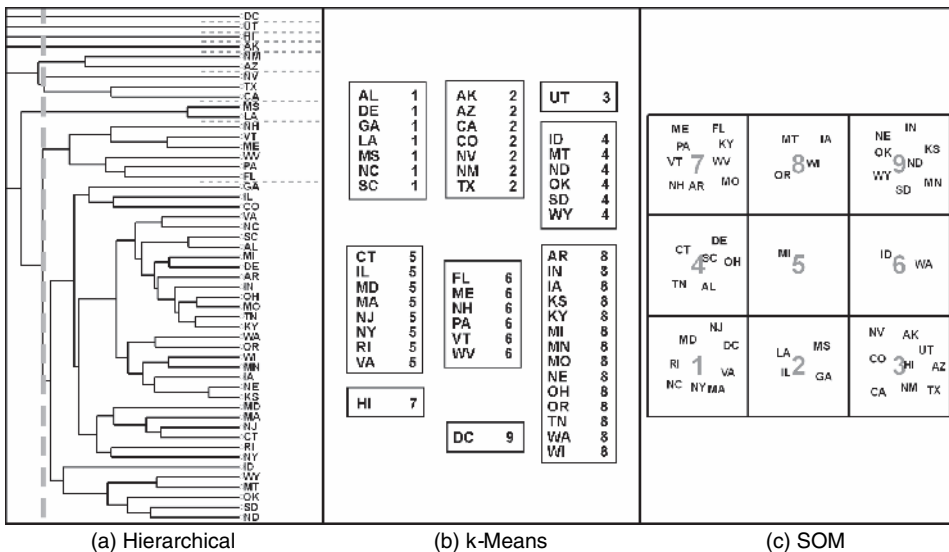


Figure 1.3 Comparing three different clustering techniques applied to demographic data for US states

Hierarchical clustering is the most widely known technique. It models distance and similarity relationships between input objects by turning each into a leaf node within

a binary tree structure. This tree is formed either by subdividing the full data set into smaller branch nodes until arriving at individual leaf nodes (divisive clustering) or by merging leaves into larger branch nodes (agglomerative clustering). The exact shape of the clustering tree is affected by the distance criterion used to evaluate candidate branch nodes before each merge (e.g. single-linkage) and by the distance measure used (e.g. Euclidean). The resulting tree structure can be visualized as a dendrogram, a portion of which is shown in Figure 1.3(a), where the average-linkage criterion and Euclidean distance measure are used. It can be seen that the hierarchical clustering tree contains multiple clustering solutions. To allow comparison with the other two methods, one solution (with nine clusters) is emphasized by applying a cut through the tree at the appropriate distance level. Horizontal, dashed lines indicated cluster separations. For example, California, Texas, and Nevada form one cluster, with Nevada joining the other two only just before the nine-cluster split and not long before New Mexico and Arizona are merged at a slightly coarser cluster level.

One downside of hierarchical clustering is that feature space partitions can be far from optimal, since it attempts to compute all possible granularities at once. Compare this with *k-means clustering*, which looks for a partition based on a given number of clusters (k). As in the hierarchical solution, Utah, Hawaii, and Washington, DC are placed into their own ‘clusters’, but other states are more evenly distributed across the other six clusters [Figure 1.3(b)]. Like *k-means*, the standard SOM algorithm also assumes a fixed number of units and uses the same objective function as *k-means clustering*. However, it creates a topologically ordered partition. For example, the nine-cluster solution derives from a topologically ordered 3×3 grid of neurons [Figure 1.3(c)]. Cluster 1 is an immediate neighbour of cluster 2, while cluster nine is far away from either. Contrast this with the *k-means* solution, in which no indication of relationships between the nine clusters is given. For in-depth coverage of various clustering techniques, readers are referred to the numerous dedicated volumes on the subject (e.g. Sneath and Sokal, 1973).

1.2.3 Dimensionality Reduction Methods

Creating a topologically ordered partition of n -dimensional data in a form supportive of low-dimensional presentation implies that the SOM method performs some type of dimensionality reduction. This is already apparent in the case of the nine-cluster solution [Figure 1.3(c)], but becomes even more relevant as we move towards larger SOMs consisting of hundreds and even thousands of neurons. In such cases, a SOM will allow mapping out of individual, n -dimensional, data vectors in a low-dimensional display space. It is thus worthwhile to compare the SOM with other dimensionality reduction techniques. PCA is the most frequently used of these methods. The first two principal components often express enough of the multivariate structure of a data set that simple two-dimensional scatter plots are commonly found, illustrated here for the same demographic data used earlier [Figure 1.4(a)]. *Multidimensional scaling* (MDS) is the technique most appropriately fitting into the dimensionality reduction category, as it attempts to preserve high-dimensional distance orderings in low-dimensional space (Kruskal and Wish, 1978). While one can choose the output dimensionality as an input parameter, the two-dimensional form is by far the most common, since it supports

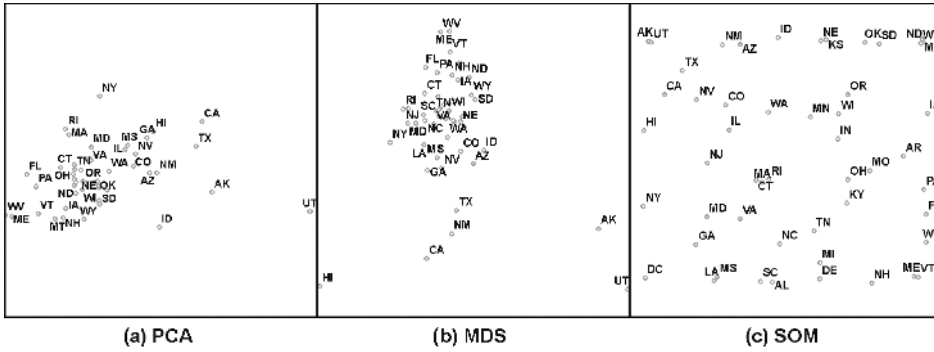


Figure 1.4 Comparing three different dimensionality reduction techniques applied to demographic data for US states

many different visualization forms, from traditional print media to interactive exploration [Figure 1.4(b)].

PCA and MDS employ an object-based conceptualization where the original input vectors are interpreted as discrete objects and are the sole basis of computation and visualization, which accordingly almost always consists of labelled point features [Figure 1.4(a) and (b)]. However, the SOM method conceptualizes input vectors not as discrete objects but as representative samples from an n -dimensional information continuum (Skupin, 2002b). During SOM training those samples drive a topologically ordered tessellated representation of that continuum. It is then not surprising that most SOM-based visualizations are constructed from a uniform cell structure resembling raster geometry. The field-like conceptualization implemented in SOM makes it easy to map various other n -dimensional vectors onto a trained SOM, even, and especially, if they were not part of the training data set. In order to allow for comparison with the PCA and MDS solutions, the trained SOM is here applied to the same vectors used during training [Figure 1.4(c)]. Notice especially the differences in the placement of outliers, like Utah or Alaska. The SOM's topology-preserving mapping makes more efficient use of the available space, at the cost of higher distortion of relative feature space distances as compared with PCA and MDS.

1.3 SOM ALGORITHM

Applications of SOM in geographic information science tend to employ the standard algorithm first described by Kohonen (1990). Therefore, it makes sense to spend some time in this chapter on introducing that algorithm. However, there already exist many good, formal descriptions of the algorithm, most notably in Kohonen's own monograph *Self-Organizing Maps* (Kohonen, 2001), and in various journal and conference proceedings articles. Readers are well advised to refer to those sources for the mathematical foundation and physiological justification of the algorithm. This introductory chapter instead presents the SOM method using mostly plain language and graphic illustrations, from pre-processing of training data to using the finished neural network.

1.3.1 Source data

The SOM method can be and has been applied to hugely diverse data sets, as will be evident from the collection of chapters in this book. Broadly speaking, one needs data containing individual items with n -dimensional, quantitative attributes. Raw source data will often already exist in that form. Well-structured data produced through a census of human population or multi-channel remote sensing are prime examples. On the other end of the spectrum, one can even use unstructured data, once they are suitably transformed. For example, a corpus of scientific articles can be turned into a SOM input data set after indexing and construction of a vector space model (Skupin, 2002a). From the analysis of mutual fund performance to classification of human voices, performing these transformations correctly is a crucial part of every analytical procedure. However, since the specific steps are highly dependent on the given subject domain, readers are advised to refer to domain-specific literature.

Pre-processing of n -dimensional vectors resembles typical procedures for other neural network methods. Two main concerns are the existence of skewed distributions and the range of values for each attribute. Neural networks tend to be fairly robust, but being aware of and, if feasible, counteracting the according effects will help to create a more useful model. When encountering highly skewed variables, logarithmic transformation is the first and most obvious choice. It is also a good idea to normalize all values of a given variable to fit into a predefined range, typically between 0 and 1.

1.3.2 Training the neural network

The SOM performs a ‘non-linear, ordered, smooth, mapping of high-dimensional input data manifolds onto the elements of a regular, low-dimensional array’ of neurons (Kohonen, 2001, p. 106). Each neuron has associated with it an n -dimensional vector of the same dimensionality as the input data. For example, if 32 census attributes for each of the states of the US are used as input, then a 32-dimensional vector is created for each neuron.

The first step in the creation of a SOM is to determine its size and topology type. The SOM’s size k is given as the number of neurons to be used in the x and y direction. Thus, a size of two neurons in x and three neurons in y would yield six neurons, while a 100×100 neuron SOM would consist of 10 000 neurons. Two topology types are frequently used. The first is the square topology, where each neuron is connected to four neighbouring neurons. When used for visualization, a 10×10 neuron SOM would thus have a square shape overall [Figure 1.5(a)]. The second, and more frequently encountered, possibility is the use of a hexagonal topology, with six neighbours to every neuron. Given an equal number of neurons in x and y one would observe a rectangular shape, with the longer side along the x -axis [Figure 1.5(b)].

Before training can begin, n weights for each neuron are initialized. In order to later observe true self-organization, one could assign random values. Alternatively, it is possible to help the training algorithm along (and shorten training times) by assigning weights according to a linear estimate, such as the first two principal components derived from the training data. In some software solutions, this is one of the built-in options for SOM initialization (Section 1.4).

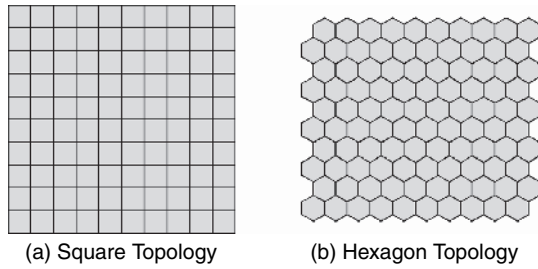


Figure 1.5 Network size and topology type of a SOM are chosen before training begins. Notice overall shape difference for SOMs with identical size, but different topology type

Training consists of an iterative process, during which individual input vectors are presented to the neuron grid, the best-matching (i.e. most similar) neuron vector is found and weights of that and other neuron vectors are modified. Understanding the nature of these modifications goes to the heart of understanding self-organization. Once the best-matching neuron is found, its n weights are modified towards an even better match with the input vector. In addition, neighbouring neurons up to a certain distance from the best-matching unit (BMU) are also modified to better fit that input vector. These focal modifications are over the course of many iterations causing similar input data to be associated with closely positioned neurons. In the literature, iterations are alternatively referred to as training steps, runs or cycles.

It is important to understand, especially in comparison with such methods as MDS, that relationships among input data are at no time directly assessed. Instead, topology preservation in a SOM is achieved as a quasi by-product of how weights of neuron vectors are adjusted during training. That is why *self-organization* is an appropriate title. A schematic example should serve to illustrate how this works (Figure 1.6). Let us assume that we were training a 3×3 neuron SOM with only four input vectors (1, 2, 3, 4). In feature space, these four vectors form two distinct clusters (1 and 4; 2 and 3). Starting with the initialized SOM, the first input vector finds the neuron at location ($x = 1; y = 2$) to be its BMU. Accordingly, weights of that node are adjusted towards the input vector. In addition, neurons within a single-neuron neighbourhood are slightly adjusted [Figure 1.6(b)]. For the next cycle, the second vector is presented to the neuron lattice, finds the neuron at ($x = 3; y = 3$) as its BMU, and adjusts weights for that neuron and its neighbours [Figure 1.6(c)]. Those adjustments cause the third

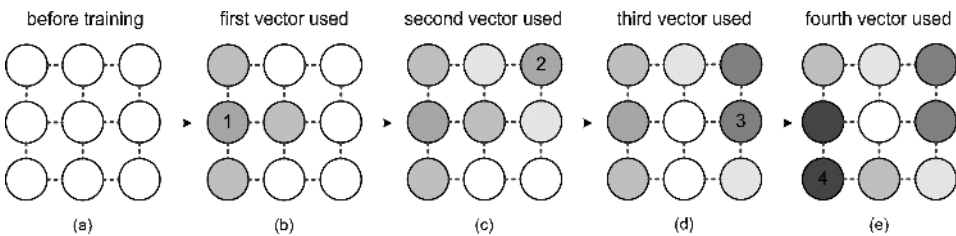


Figure 1.6 Process of self-organization during SOM training. A 3×3 neuron SOM is trained with four observations representing two distinct groups in attribute space (See Colour Plate 1)

vector to be drawn to the vicinity of the previous vector at $(x = 3; y = 2)$. Weights of its BMU and neighbouring neurons are modified. However, the single-neuron neighbourhood includes neurons that previously underwent modification on account of the first and second vector. Those neurons now undergo further modifications. Notice how the neuron at $(x = 2; y = 2)$ becomes a separator between cluster regions, since members of the two clusters have attempted to pull it in either direction [Figure 1.6(d)]. Finally, the fourth vector finds its BMU at $(x = 1; y = 1)$. The ensuing weight adjustments finish the self-organization of the SOM into two distinct clusters [Figure 1.6(e)]. This whole process would however be repeated many times over when dealing with real data. As training progresses, an input vector may then be reused and find a BMU that is different from the previous cycle it was involved in. For example, the first input vector may now find the neuron at $(x = 1; y = 1)$ to be a better fit. As a rule though, major global relationships will be established early, followed by a distinction of finer structures late during training.

To look at the training process more formally, let us consider the input data as consisting of n -dimensional vectors \mathbf{x} :

$$\mathbf{x} = [\xi_1, \dots, \xi_n]^T \in \mathfrak{R}^n \quad (1.1)$$

Meanwhile, each of k neurons has an associated reference vector \mathbf{m}_i :

$$\mathbf{m}_i = [\mu_{i1}, \dots, \mu_{in}]^T \in \mathfrak{R}^n \quad (1.2)$$

During training, one x at a time is compared with all \mathbf{m}_i to find the reference vector \mathbf{m}_c that satisfies a minimum distance or maximum similarity criterion. Though a number of measures are possible, the Euclidean distance is by far the most common:

$$\|\mathbf{x} - \mathbf{m}_c\| = \min_i \{\|\mathbf{x} - \mathbf{m}_c\|\} \quad (1.3)$$

The best-matching unit and neurons within its neighbourhood are then activated and modified:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t)[\mathbf{x}(t) - \mathbf{m}_i(t)] \quad (1.4)$$

One of the main parameters influencing the training process is the *neighbourhood function* (h_{ci}), which defines a distance-weighted model for adjusting neuron vectors. Two functions are most popular, the linear and the Gaussian model (shown here):

$$h_{ci}(t) = \alpha(t) \cdot e^{-d_{ci}^2/2\sigma_i^2(t)} \quad (1.5)$$

One can see that the neighbourhood function is dependent on both the distance between the BMU and the respective neuron (d_{ci}) and on the time step reached in the overall training process (t). The maximum of d_{ci} corresponds to the *neighbourhood radius*, which is a training parameter determining the set of reference vectors to be modified around each BMU at a particular time step [$N_c(t)$]. In the Gaussian model, that neighbourhood's size appears as kernel width (σ) and is not a fixed parameter. The neighbourhood radius is used to set the kernel width with which training will start. One typically starts with a neighbourhood spanning most of the SOM, in order to achieve a rough global ordering, but kernel width then decreases during later training cycles. Similarly, the initial

learning rate (α_0) is an input parameter, which is then gradually decreased as t progresses [$\alpha(t)$]. SOM training stops when a predetermined number of training cycles (t_{\max}) are completed.

As self-organization progresses during training and neighbourhood radius and training rate slowly decrease, the SOM gradually settles into a stable configuration (Figures 1.7 and 1.8). One way of visualizing this is to show the weights of a particular variable for each neuron and observe changes over multiple training cycles. In Figure 1.7 ‘vacant housing’ as one of 32 census variables is shown on a 20×20 neuron SOM, with snapshots at four time periods. Training begins with randomized weights. Early cycles establish major global relationships, visible here as an almost linear relationship between vacant housing and the x -axis after 40 000 cycles. After 80 000 cycles, more detailed structure emerges, with low values of vacant housing in the centre-left portion of the SOM. Finer, local structures emerge during the remaining cycles, with training ending at 100 000 cycles. Alternatively, the training progress could be visualized by plotting individual training vectors onto the trained SOM at chosen cycling intervals according to the location of the best-matching unit. These temporal vertices are then linked to form trajectories (Figure 1.8). With snapshots taken every 10 000 cycles, one can see how the SOM settles towards the end of training, as indicated by a lack of major movement after about 80 000–90 000 cycles.

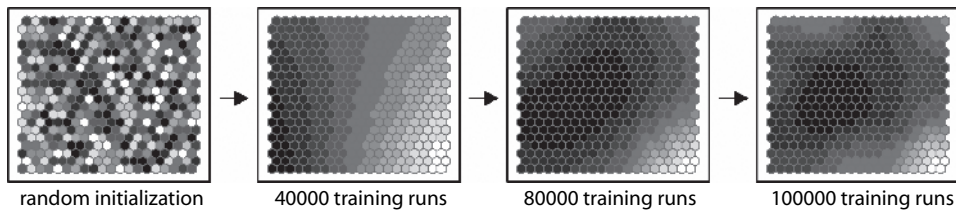


Figure 1.7 Changes to the component plane for the variable ‘vacant housing’ during training of a SOM with demographic data for US states. Higher values indicated by lighter shading

Most of the parameters mentioned here can be specified before beginning the training process, including the network size, topology type, distance function, neighbourhood function, neighbourhood radius, total number of training cycles, and training rate. The ability of directly influencing these parameters constitutes much of the difference between the various SOM implementations, including those found in commercially available software.

1.3.3 Using the trained neural network

Once training is finished, the neural network is ready for use. First, it is advisable to visualize the SOM itself, and sometimes this alone already justifies use of the SOM method. Another main mode of using a trained SOM is to map individual n -dimensional features into low-dimensional space by finding the best-matching neuron. For the purpose of interactive, exploratory analysis, SOM can also be linked to other visualizations, including geographic maps.

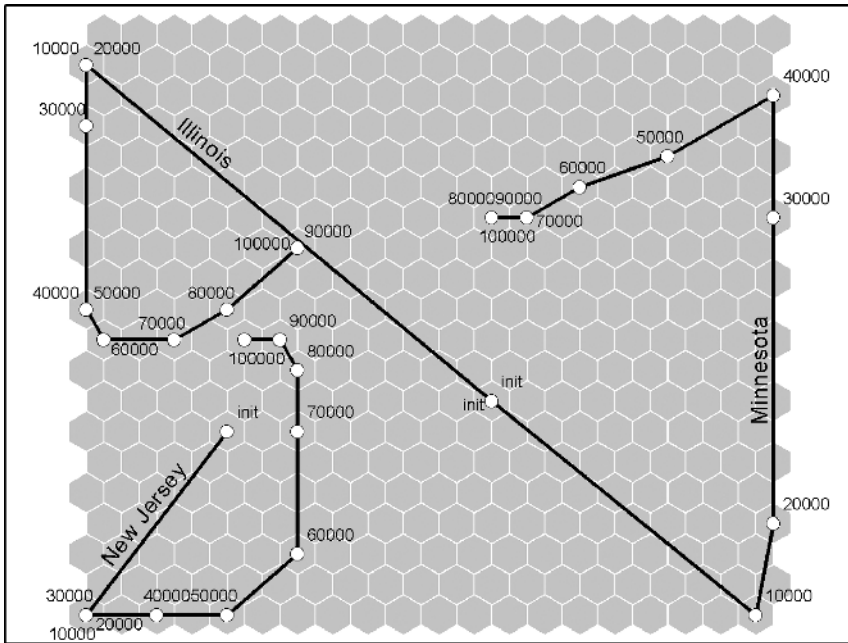


Figure 1.8 Position changes of three states while training a SOM for a total of 100 000 runs. Starting with a randomly initialized SOM, each state is recorded once every 10 000 runs

1.3.3.1 Visualizing the SOM

The main methods for visualizing a SOM involve *component planes*, *distortion patterns*, and *clustering*. *Component plane* visualization symbolizes neuron weights for individual variables. For example, with census data one could create separate visualizations for each input variable, such as population density, percentage of Hispanic population, and so forth. One of the possible applications is to look for relationships between variables, based on visual similarity of component planes. The most common approach to component plane visualization is to apply colour or grey shading, as seen in the visualization of vacant housing (Figure 1.7). Other possibilities include the use of graduated symbols for individual variables or the placement of bar charts to show the weights for multiple variables at each neuron.

While SOM training has the effect of preserving major topological relationships, geometric proximities can be drastically distorted. This refers particularly to contraction effects observed for sparsely occupied or empty feature space regions and expanded representation of high-density regions (Lin *et al.*, 2000). In more general terms, one can state that low-density and high-density regions in feature space are associated with marked distortion when they are modelled in the low-dimensional space of topologically ordered neurons. One common approach is to visualize the degree of distortion, i.e. the change in relative distance between n -dimensional locations and their low-dimensional representation, and treat zones of high contraction as a type of cluster boundary. Identifying these ‘clusters’ can be rather subjective though, especially when different magnitudes of distortion are encountered in different regions of a SOM, since it

is up to the human observer to decide when visual structure constitutes a cluster boundary. The most frequently used method to visualize distortion patterns is the U-matrix method (Utsch, 1993), which explicitly symbolizes the n -dimensional distance of neighbouring neurons.

A third approach to visualizing the SOM itself is to treat each neuron as a distinct feature possessing an n -dimensional vector, to which traditional cluster techniques, like hierarchical or k -means, can be applied. Since the neurons are already topologically ordered, one will find that such n -dimensional clusters tend to form regions in two-dimensional SOM space. This can be especially useful with high-resolution SOMs, for example to enable multi-scale, interactive exploration (Skupin, 2002a).

1.3.3.2 Mapping data onto the SOM

Visualizing a SOM means exploring the model itself that one has created through neural network training. However, *applying* the model will often involve the mapping of n -dimensional vectors *onto* the trained SOM. The bulk of SOM applications are focused on this aspect of Kohonen's method. For example, in industrial applications, one could track a machine part based on various measurable attributes. In an analysis of voting behaviour of different politicians, one could map individual persons in two dimensions, as an alternative to MDS, which has traditionally been used for this purpose. Geographic objects can also be mapped onto a SOM, as shown in Figures 1.3(c) and 1.4(c). Those figures also illustrate the difference between using a SOM for classification into a limited number of classes [Figure 1.3(c)] and spatial layout with differentiated locations for many features [Figure 1.4(c)]. Speaking of clustering, please note that for supervised classification one should not use the SOM method itself but a related method called *learning vector quantization* (Kohonen, 2001).

When input features can be arranged into meaningful sequences, output locations derived from the locations of best-matching units can be strung together to form trajectories. This has been demonstrated for multi-temporal data, where the same features and attributes are observed for multiple time periods (Deboeck and Kohonen, 1998; Skupin and Hagelman, 2005). Other possibilities include the mapping of space-time paths onto a SOM trained with the attributes of geographic features (see Chapter 6). Even the training process itself can be visualized via trajectories (Figure 1.8).

1.3.3.3 Linking SOM to other visualizations

In most circumstances, SOMs will not become the sole analytical tool for investigating an n -dimensional data set. Instead, it constitutes an additional method that will be used in conjunction with other computational and visual tools. Integration of a SOM with other forms of representation is becoming increasingly important, especially when dealing with geographic data, for which a dominant visual form already exists in the form of geographic maps. Integration of most SOMs with more traditional geographic visualizations is straightforward since a two-dimensional SOM can be readily represented using standard GIS data structures. There are obvious advantages to doing this in an interactive setting, but it can even be useful for static cartographic output (Figure 1.9). Notice how

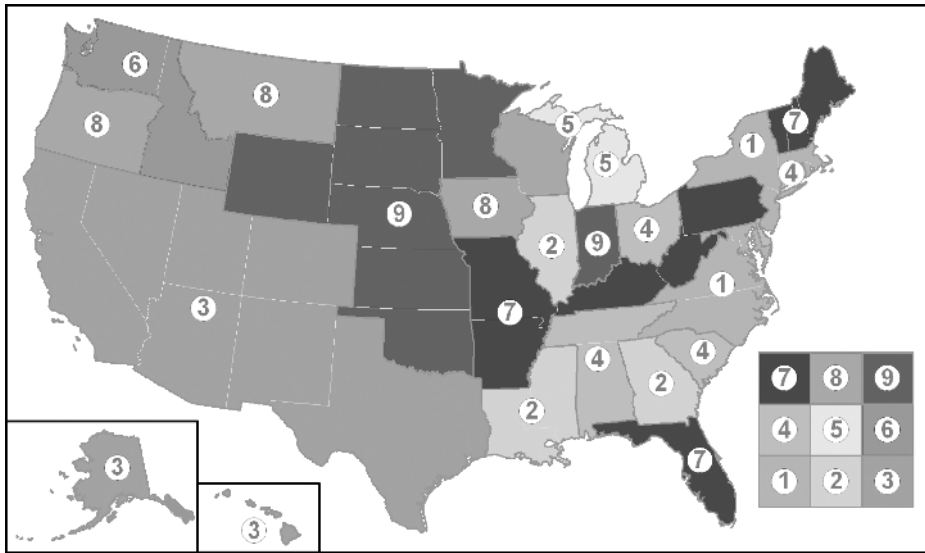


Figure 1.9 SOM-based clustering of census data combined with colour design informed by network topology. Relationships among clusters are indicated by displaying legend constructed from two-dimensional SOM geometry (See Colour Plate 2)

the low-dimensional, topologically ordered set of neurons helps with the design and presentation of a colour legend, in addition to its clustering function.

Side-by-side display of SOM and geographic map in a commercial off-the-shelf GIS was demonstrated as early as 1998 (Li, 1998), but little progress has been made since. In fact, most geographic SOM applications still suffer from a lack of imagination regarding integration, as indicated by the preponderance of run-off-the-mill graphic output from dedicated SOM software (especially the SOM Toolbox for Matlab) in the GIScience literature. One example for a different direction is the integration of SOM training and visualization in the GeoVISTA Studio system (Gahegan *et al.*, 2002). An interactive approach mostly unexplored is the linking of different SOM solutions, for example linking a high-resolution SOM to one consisting of a small number of neurons.

1.3.4 Extensions of the SOM method

Beyond the widely used approach presented in the previous sections, numerous extensions and modifications of the SOM method have been proposed. Most of these have not yet found their way into end-user SOM implementations, including those described in this book. Nonetheless, many of these modifications do indeed improve upon Kohonen's original method in a number of ways.

One example is the notion of a *growing SOM* described by Fritzke (1999). Instead of a fixed network size, this approach provides for the insertion of new nodes into the neural network in response to specific nodes not satisfying a certain objective function. For example, one could use the *quantization error* as such an objective function, i.e.

how well a given node actually fits those data items for which it has become the best-matching unit. New nodes would be inserted into the neighbourhood of existing nodes that have high quantization errors. One could envisage numerous variants of this approach, for example using an entropy maximization goal instead of the error minimization described above (Fritzke, 1999). Conversely to the adding of nodes, growing SOMs may also involve the removal of individual nodes, if some objective criterion is thereby improved.

A frequently encountered issue with the traditional SOM algorithm is the appearance of border or edge effects. The most obvious of these is that neurons near the edge of the SOM come to represent larger portions of the n -dimensional input space. In other words, there is a large degree of compression or contraction at the edge as compared with inner neurons. This becomes most obvious during the mapping of large numbers of vectors onto the SOM, where one frequently observes pronounced clustering along the edges. Efforts to address this issue include the growing SOM discussed above and arranging neurons into lattices that are not flat, but curved such that edges are reduced (e.g. forming a cylinder). Arrangement of neurons on a sphere would completely eliminate edges, which is why a number of proposals concerning *spherical SOMs* have been put forward (Ritter, 1999; Sangole and Knopf, 2002; Wu and Takatsuka, 2005).

Other proposals have dealt with how the neighbourhood around the best-matching unit is defined during training. Teuvo Kohonen himself introduced a number of modifications and new methods building on the original SOM approach, many of which are documented in his monograph (Kohonen, 2001). For example, *learning vector quantization* (LVQ) adapts many of the same principles to provide supervised classification. *Adaptive subspace SOM* (ASSOM) and *hierarchically structured SOMs* are other proposals that have received attention in recent years. Bação *et al.* (see Chapter 2) discuss a number of SOM variants that are of particular interest to GIScience.

1.4 SOFTWARE TOOLS FOR SOMs

There are a number of software tools available for SOMs, in the commercial as well as the public domain. This section will specifically discuss the ones that are commonly employed in geographical analysis, including by the authors in this volume. This is intended as a pointer to the various software options available as a resource for GIScientists and is not meant as a comprehensive, self-help guide to these tools. For this, the reader is advised to refer to documentation and help files, which tend to be freely downloadable for public domain software or are included in the licensed versions of commercial software. Additionally, readers can refer to other volumes, such as Kohonen (2001) and Deboeck and Kohonen (1998), that include overviews of SOM software. This section also intends to provide an updated review of some of the software that was included in these previous volumes, such as SOM_PAK. Most applications described in this book are based on using public domain software, although a few, such as Bação *et al.* (Chapter 2), use self-coded variants of SOM that extend on its basic functionality in order to accommodate the specific nature of geographic data. In some cases, stand-alone software such as SOM_PAK is

used in combination with GIS software for data analysis as well as visualization of SOM output.

1.4.1 Stand-alone Software

One of the most widely used implementations of the basic SOM algorithm is found in the SOM_PAK program (Kohonen *et al.*, 1996), which is freely available from the Neural Networks Research Centre of the Helsinki University of Technology (http://www.cis.hut.fi/research/som_pak/). First made public about 10 years ago, many researchers had their first practical experience with SOM using this software. While its command-line interface can at first seem daunting, the involved commands and underlying methods are actually well explained in the accompanying documentation. The simple user interface makes it easy to port the software to various hardware platforms. The complete source code, written in C, is accessible to programmers for noncommercial uses. For example, one could implement alternative similarity coefficients, since only Euclidean distance is built in (Skupin, 2003). Novice, nonprogrammer users will be thankful that executable files for Windows are available for download, because compilation of the source code can be tricky.

Use of SOM_PAK involves four main steps: map initialization, map training (in multiple stages if desired), evaluation of quantization error, and visualization. A major downside to using SOM_PAK is the lack of convenient visualization capabilities beyond the creation of static output in PostScript format. However, the trained SOMs are stored in straightforward text files, known as codebook files (.cod), which can easily be read by other packages. In fact, the codebook format has become a de facto standard for the distribution of SOMs. Many of the software packages listed below offer respective import options (e.g. SOMine, Nenet, SOM ToolBox). It also does not take particularly advanced programming skills to turn the codebook files into something usable within GIS. For example, many of the figures in this chapter were created by transforming the content of SOM_PAK's codebook files into ArcInfo Generate files, before performing all further transformations inside of ArcGIS.

Viscovery SOMine is a commercial product of Eudaptics Inc. (<http://www.eudaptics.de>) that is distributed as a Windows application in several versions, with different capabilities depending on price. Built around a custom version of the SOM training algorithm it provides for the training and use of SOMs, including clustering, prediction, and exploratory visualization, all in a highly interactive environment. Another interesting SOM tool is *Nenet* (Neural Network Tool), which first became available in 1998, and which provides a full graphical user interface for training and visualization. A limited functionality version remains freely available (<http://koti.mbnet.fi/~phodju/nenet/Nenet/General.html>), but the current status of the full software is unknown.

1.4.2 Add-In Software and Software Components

When faced with complex exploratory visualization tasks, SOMs are best used in conjunction with other visualization and analysis tools. As an alternative to stand-alone

SOM software, one will often find that the addition of SOM capability to existing analytical tools and integration into existing software development architectures is more useful.

The *SOM Toolbox for Matlab* was developed by researchers at the Helsinki University of Technology in recognition of the large numbers of users performing numerical modelling and analysis in Matlab. While the latter is a commercial product, the SOM Toolbox is public-domain software consisting of a large number of Matlab routines (.m files). Download (<http://www.cis.hut.fi/projects/somtoolbox>) and installation of the toolbox is straightforward. Some of the most common training and visualization options are accessible through a limited graphical user interface, but full functionality and control over training and visualization require use of Matlab's command-line interface. The sequence of training steps is similar to SOM_PAK, though a number of further options are available. When it comes to visualization, the SOM Toolbox offers an attractive variety of methods and plenty of opportunity for customization. Low cost, ease of SOM training and visualization, and a large Matlab user base have made the SOM Toolbox one of the most popular SOM solutions.

The full potential of the SOM method for interactive, exploratory visualization is most easily tapped when SOM functionality can be integrated into larger visualization software architectures. However, SOM component solutions, for example based on ActiveX or JavaBeans, are still hard to find. One exception is the set of SOM beans included with GeoVISTA Studio (<http://www.geovistastudio.psu.edu>), an open-source visualization software environment developed at Pennsylvania State University (Gahegan *et al.*, 2002). GeoVISTA Studio comes with a large number of beans, including for data input, preprocessing, numerical analysis, and visualization. More importantly, it provides a code-free programming environment, where the flow of data is directed through visual manipulation of links between different beans. While this process does not involve writing code directly, previous knowledge of Java and JavaBeans greatly helps and novices need significant practice to make meaningful use of GeoVISTA Studio. One notable advantage of the system is that the interactive visualizations constructed through inclusion and wiring of beans can immediately be tested, even before being deployed as applets or applications. GeoVISTA Studio's SOM beans provide training and various visualization methods, including two- and three-dimensional U-matrix visualizations. Integration with other methods, such as geographic maps, scatter plot matrices, or parallel coordinate plots, does not only allow linked selection, but also linked symbolization, where colour choices are shared among multiple methods.

1.5 GISCIENCE AND SOMS

As a method developed outside of GIScience, it is natural to see the relationship between the SOM and GIScience primarily as the latter adopting the former to serve specific analytical purposes. Clustering, visualization, and the range of applications laid out in the various chapters of this book is indeed impressive. However, this does not have to be a one-way relationship. GIScientists may offer a unique perspective and contribute to the further development of this popular neural computing approach. This begins with the storage and manipulation of trained SOMs leveraging the ability of GIS to handle

complex geometric and attribute structures. There has so far been little reflection on how such geographic notions as *scale* become relevant when dealing with nongeographic data or how geographic conceptualizations of *fields* versus *discrete objects* are manifested in different spatializations (Skupin, 2002b). The design of SOM-based visualization stands to gain from traditional cartographic design considerations, for example regarding visual hierarchies and semiotic variables. Meaningful interaction with large SOMs will benefit more from semantically driven notions of scale dependence used in cartography than from the performance-oriented level-of-detail (LOD) approaches common in computer graphics. The notion of GIScience itself was conceived in recognition of the growing need to pursue interdisciplinary strategies, and our community is now actively engaging the SOM method accordingly. This book presents the current state of this endeavour.

1.6 ORGANIZATION OF THE BOOK

The primary aim of this book is to act as a showcase for the valuable role that SOM can play in geographic analysis. This book is about solving academic, theoretical, and applied problems and converting interesting computational methods into useful operational tools. It is also about finding new uses and about providing novel solutions to established problems. The sequence of chapters reflects this view of SOM as a method of many colourful facets.

The current chapter provides an introductory overview of principles, algorithms, and tools associated with the SOM. The method is explored further in Chapter 2 by Bação *et al.*, who demonstrate multiple SOM variants designed to address the specific nature of geographic data and problems. In Chapter 3, Koua and Kraak demonstrate the use of SOM to reveal hidden patterns within an integrated, exploratory visualization environment. In Chapter 4, Thill *et al.* describe work with a linguistic database where the SOM method is used to mine and visualize latent organization rules within the data. That chapter also demonstrates the development of an integrated environment for exploration of SOM output within a standard Windows-based GIS platform. Yan and Thill, in Chapter 5, use SOM as an exploratory data mining tool for spatial interaction data to visualize flows and movements in space within an interactive environment. An extension to the traditional notion of space–time paths is presented by Skupin in Chapter 6, where movement across geographic space is linked to simultaneous movement in n -dimensional attribute space and visualized as a SOM trajectory. Sester, in Chapter 7, is concerned with issues of typification in cartographic generalization, where the density-preserving tendencies of SOM training can be exploited for multi-scale mapping. Hewitson, in chapter 8, demonstrates the diversity of applications supported by the use of the SOM method for climate analysis. Kropp and Schellnhuber, in Chapter 9, then introduce an approach designed to derive global biogeographical prototypes that could be used in climate impact studies. Doucette *et al.*, in Chapter 10, report on experiments aimed at extracting road features from high-resolution, multi-spectral imagery. Finally, an epilogue by Goodchild (Chapter 11) weaves together the diverse strands of SOM applications described in this book, drawing connections to developments in such areas as exploratory spatial data analysis (ESDA) and commenting on how the SOM method relates to the state and future of GIScience.

ACKNOWLEDGEMENTS

We would like to thank the following individuals that performed reviews of submitted chapters and without whom this edited volume would not have been possible: Fernando Bação, Katy Börner, John Cassano, Tae-Soo Chon, Peter Doucette, Aude Esperbé, Sara Fabrikant, Peter Fisher, Stewart Fotheringham, Sven Fuhrmann, Diansheng Guo, Stefan Hinz, Bin Jiang, Jürgen Kropp, Mei-Po Kwan, Charles Schmidt, Monika Sester, Jean-Claude Thill, and Matt Thornton.

REFERENCES

- Deboeck, G., and T. Kohonen, eds. 1998. *Visual Explorations in Finance with Self-Organizing Maps*. Berlin, Heidelberg, New York: Springer-Verlag.
- Fischer, M. M. 2001. Computational Neural Networks – Tools for Spatial Data Analysis. In *Geocomputational Modelling: Techniques and Applications*, eds M. M. Fischer and Y. Leung, 15–34. Berlin, Heidelberg, New York: Springer-Verlag.
- Fischer, M. M., and Y. Leung. 2001. *Geocomputational Modelling: Techniques and Applications, Advances in Spatial Science*. Berlin, New York: Springer-Verlag.
- Fotheringham, A. S., C. Brunson, and M. Charlton. 2000. *Quantitative Geography: Perspectives on Spatial Data Analysis*. London, Thousand Oaks, CA: Sage Publications.
- Fotheringham, A. S., C. Brunson, and M. Charlton. 2002. *Geographically Weighted Regression: The Analysis of Spatially Varying Relationships*. Chichester: John Wiley & Sons, Ltd.
- Fritzke, B. 1999. Growing Self-Organizing Networks – History, Status Quo, and Perspectives. In *Kohonen Maps*, eds E. Oja and S. Kaski, 131–144. Amsterdam: Elsevier.
- Gahegan, M., M. Takatsuka, M. Wheeler, and F. Hardisty. 2002. Introducing GeoVISTA Studio: an integrated suite of visualization and computational methods for exploration and knowledge construction in geography. *Computers, Environment and Urban Systems* 26:267–292.
- Gurney, K. 1997. *An Introduction to Neural Networks*. London: UCL Press.
- Hertz, J., A. Krogh, and R. G. Palmer. 1990. *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley.
- Kohonen, T. 1990. The self-organizing map. *Proceedings of the IEEE* 78(9):1464–1480.
- Kohonen, T. 2001. *Self-Organizing Maps*. Berlin: Springer-Verlag.
- Kohonen, T., J. Hynninen, J. Kangas, and J. Laaksonen. 1996. *SOM_PAK: The Self-Organizing Map Program Package, Technical Report A30*. Espoo: Helsinki University of Technology, Laboratory of Computer and Information Science.
- Kruskal, J. B., and M. Wish. 1978. *Multidimensional Scaling, Sage University Papers Series. Quantitative Applications in the Social Sciences. 11*. Beverly Hills, CA: Sage Publications.
- Li, B. 1998. Exploring spatial patterns with self-organizing maps. Paper presented at GIS/LIS '98, 10–12 November, Fort Worth, TX.
- Lin, C., H. Chen, and J. F. Nunamaker. 2000. Verifying the proximity and size hypothesis for self-organizing maps. *Journal of Management Information Systems* 16(3):57–70.
- Longley, P. 1998. *Geocomputation: A Primer*. Chichester, New York: John Wiley & Sons, Ltd.
- Miller, H. J., and J. Han, eds. 2001. *Geographic Data Mining and Knowledge Discovery, Research Monographs in Geographic Information Systems*. London, New York: Taylor & Francis.
- Openshaw, S., and C. Openshaw. 1997. *Artificial Intelligence in Geography*. Chichester, New York: John Wiley & Sons, Ltd.
- Openshaw, S., and R. J. Abraham, eds. 2000. *GeoComputation*. London, New York: Taylor & Francis.

- Pijanowski, B. C., D. G. Brown, B. A. Shellito, and G. A. Manik. 2002. Using neural networks and GIS to forecast land use changes: a land transformation model. *Computers, Environment and Urban Systems* 26(6):553–576.
- Ritter, H. 1999. Self-Organizing Maps on Non-Euclidean Spaces. In *Kohonen Maps*, eds E. Oja and S. Kaski, 97–109. Amsterdam: Elsevier.
- Rogerson, P. A. 2001. *Statistical Methods for Geographers*. London: Sage.
- Rumelhart, D. E., and J. L. McClelland. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. 2 vols, *Computational Models of Cognition and Perception*. Cambridge, MA: MIT Press.
- Sangole, A., and G. K. Knopf. 2002. Representing high-dimensional data sets as closed surfaces. *Information Visualization* 1(2):111–119.
- Skupin, A. 2002a. A cartographic approach to visualizing conference abstracts. *IEEE Computer Graphics and Applications* 22(1):50–58.
- Skupin, A. 2002b. On Geometry and Transformation in Map-Like Information Visualization. In *Visual Interfaces to Digital Libraries (Lecture Notes in Computer Science 2539)*, eds K. Börner and C. Chen, 161–170. Berlin: Springer-Verlag.
- Skupin, A. 2003. A novel map projection using an artificial neural network. Paper presented at 21st International Cartographic Conference, 10–16 August, Durban, South Africa.
- Skupin, A., and R. Hagelman. 2005. Visualizing demographic trajectories with self-organizing maps. *GeoInformatica* 9(2):159–179.
- Sneath, P. H. A., and R. R. Sokal. 1973. *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. San Francisco: W. H. Freeman.
- Ultsch, A. 1993. Self-Organizing Neural Networks for Visualization and Classification. In *Information and Classification: Concepts, Methods, and Applications*, eds O. Opitz, B. Lausen and R. Klar, 307–313. Berlin, Heidelberg: Springer-Verlag.
- Wu, Y., and M. Takatsuka. 2005. Geodesic self-organizing map. Paper presented at Conference on Visualization and Data Analysis, 17–18 January, San Jose, CA.