Self-Organising Maps

Applications in Geographic Information Science

Editors

PRAGYA AGARWAL

Department of Geomatic Engineering, University College, London, UK

ANDRÉ SKUPIN

Department of Geography, San Diego State University, USA

John Wiley & Sons, Ltd

Self-Organising Maps

This page intentionally left blank

Self-Organising Maps

Applications in Geographic Information Science

Editors

PRAGYA AGARWAL

Department of Geomatic Engineering, University College, London, UK

ANDRÉ SKUPIN

Department of Geography, San Diego State University, USA

John Wiley & Sons, Ltd

Library of Congress Cataloging in Publication Data

Self-organising maps : applications in geographic information science / editors, Pragya Agarwal, Andre Skupin.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-02167-5 (cloth)
1. Geographic information systems—Mathematical models. 2. Self-organizing maps.
I. Agarwal, Pragya. II. Skupin, Andre.
G70.212.S45 2008
910.285—dc22

2007039305

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 978-0-470-02167-5

Typeset in 10/12pt Times by Integra Software Services Pvt. Ltd, Pondicherry, India
Printed and bound in Great Britain by Antony Rowe, Chippenham, Wiltshire
This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

Contents

List of Contributors

Pragya Agarwal Department of Geomatic Engineering, University College London, London WC1E 6BT, UK

Peggy Agouris Center for Earth Observing and Space Research, George Mason University, Fairfax, Virginia 22030, USA

Fernando Bação ISEGI/UNL, Campus de Campolide, 1070-312 Lisboa, Portugal

Irene Casas Department of Geography and National Center for Geographic Information and Analysis, University at Buffalo, The State University of New York, Amherst, NY 14261, USA

Pete Doucette Principal Scientist, ITT Corporation, Advanced Engineering and Sciences, Alexandria, VA 22303 USA

Michael Goodchild National Center for Geographic Information and Analysis, University of California, Santa Barbara, CA 93106, USA

Bruce C. Hewitson Department of Environmental and Geographical Science, University of Cape Town, Rondebosch 7701, South Africa

Etien L. Koua International Institute for Geoinformation Science and Earth Observation (ITC), PO Box 6, 7500 AA Enschede, The Netherlands

Menno-Jan Kraak International Institute for Geoinformation Science and Earth Observation (ITC), Department of Geo-Information Processing, PO Box 6, 7500 AA Enschede, Hengelosestraatgg, The Netherlands

William A. Kretzschmar Department of English, University of Georgia, Athens, GA 30602, USA

Jürgen P. Kropp Potsdam Institute for Climate Impact Research, PO Box 601203, 14412 Potsdam, Germany

Victor Lobo ISEGI/UNL, Campus de Campolide, 1070-312 Lisboa, Portugal and Portuguese Naval Academy, Alfeite, 2810-001 Almada, Portugal

Marco Painho ISEGI/UNL, Campus de Campolide, 1070-312 Lisboa, Portugal

Monika Sester Institute for Cartography and Geoinformatics, Leibniz University of Hannover, 30167 Hannover, Germany

Hans-Joachim Schellnhuber Potsdam Institute for Climate Impact Research, 14412 Potsdam, PO Box 601203, Germany

André Skupin Department of Geography, San Diego State University, San Diego, CA 92182-4493, USA

Anthony Stefanidis Department of Earth Systems and Geoinformation Sciences, George Mason University, Fairfax, Virginia 22030, USA

Jean-Claude Thill Department of Geography and Earth Sciences and Center for Applied Geographic Information Science, University of North Carolina at Charlotte, Charlotte, NC 28223, USA

Jun Yan Department of Geography and Geology, Western Kentucky University, Bowling Green, KY 42101, USA

Xiaobai Yao Department of Geography, University of Georgia, Athens, GA 30602, USA

1

Introduction: What is a Self-Organizing Map?

André Skupin[1] and Pragya Agarwal[2]
[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

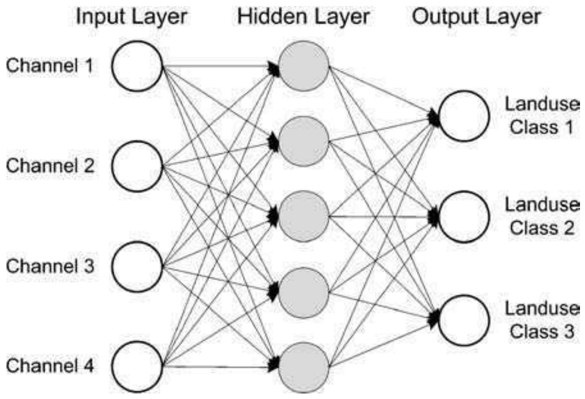

Figure 1.1 *Supervised, feed-forward, neural network trained with multispectral remote sensing data and known landuse classes*

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 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

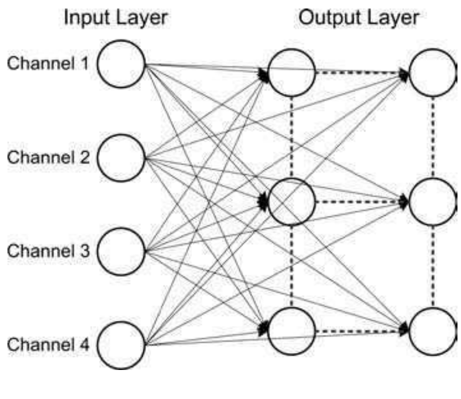

Figure 1.2 *Small SOM trained with multispectral remote sensing data*

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).

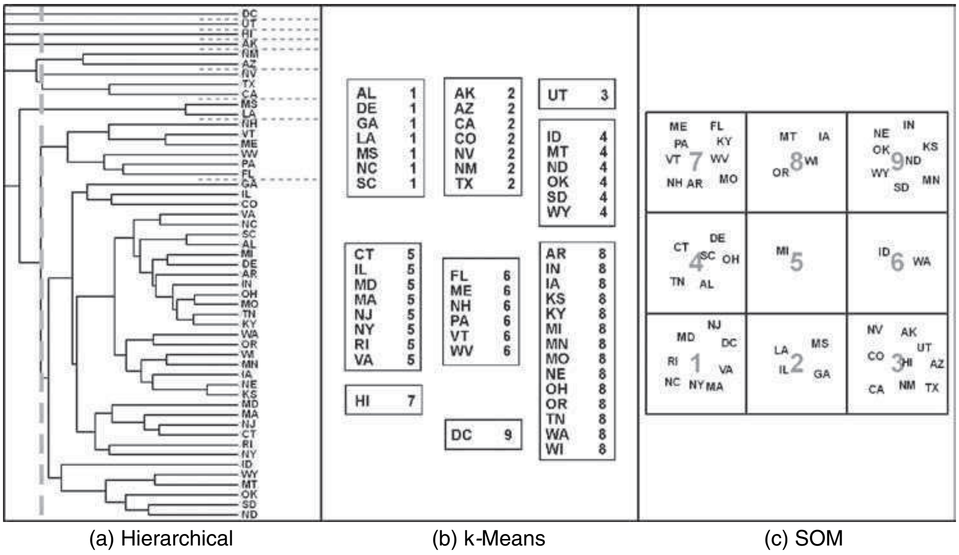
DC UT HI AK NM AZ NV TX CA MS LA NH VT ME WV PA FL GA IL CO VA NC SC AL MI DE AR IN OH MO TN KY WA OR WI MN IA NE KS MD MA NJ CT RI NY ID WY MT OK SD ND
AL 1 DE 1 GA 1 LA 1 MS 1 NC 1 SC 1
AK 2 AZ 2 CA 2 CO 2 NV 2 NM 2 TX 2
UT 3
ID 4 MT 4 ND 4 OK 4 SD 4 WY 4
CT 5 IL 5 MD 5 MA 5 NJ 5 NY 5 RI 5 VA 5
FL 6 ME 6 NH 6 PA 6 VT 6 WV 6
HI 7
AR 8 IN 8 IA 8 KS 8 KY 8 MI 8 MN 8 MO 8 NE 8 OH 8 OR 8 TN 8 WA 8 WI 8
DC 9
7 ME FL PA KY VT WV NH AR MO
8 MT IA OR WI
9 NE IN OK KS WY ND SD MN
4 CT DE SC OH TN AL
5 MI
6 ID WA
1 MD NJ DC RI VA NC NY MA
2 LA MS IL GA
3 NV AK UT CO HI AZ CA NM TX
(a) Hierarchical
(b) k-Means
(c) SOM

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

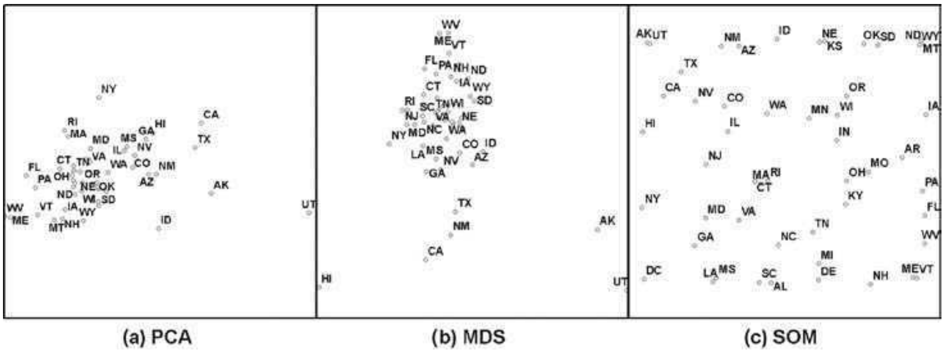

Figure 1.4 *Comparing three different dimensionality reduction techniques applied to demographic data for US states*

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).

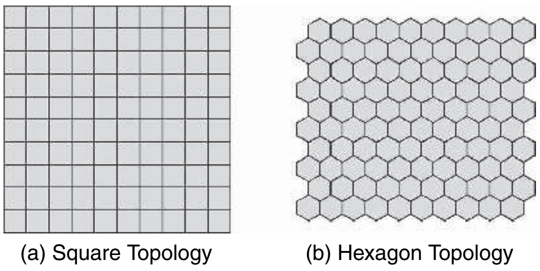

Figure 1.5 *Network size and topology type of a SOM are chosen before training begins. Notice overall shape difference for SOMs with identical size, but different topology type*

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

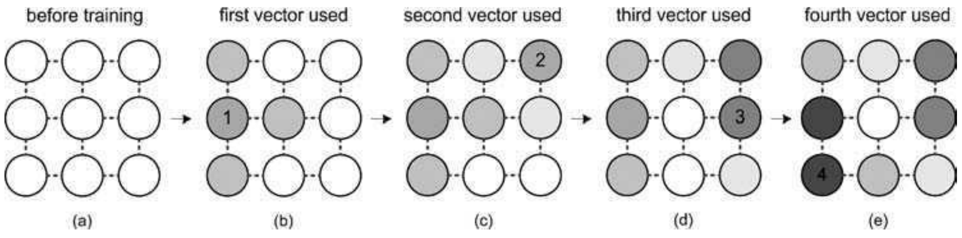

Figure 1.6 *Process of self-organization during SOM training. A 3×3 neuron SOM is trained with four observations representing two distinct groups in attribute space (See Colour Plate 1)*

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 $\boldsymbol{x}$:

$$x = [\xi_1, \ldots, \xi_n]^{\mathrm{T}} \in \Re^n \tag{1.1}$$

Meanwhile, each of k neurons has an associated reference vector $\boldsymbol{m_i}$:

$$\boldsymbol{m_i} = [\mu_{i1}, \ldots, \mu_{in}]^{\mathrm{T}} \in \Re^n \tag{1.2}$$

During training, one x at a time is compared with all $\boldsymbol{m_i}$ to find the reference vector $\boldsymbol{m_c}$ 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:

$$\|x - \boldsymbol{m_c}\| = \min_i\{\|x - \boldsymbol{m_c}\|\} \tag{1.3}$$

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

$$\boldsymbol{m_i}(t+1) = \boldsymbol{m_i}(t) + h_{ci}(t)[x(t) - \boldsymbol{m_i}(t)] \tag{1.4}$$

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):

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

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.

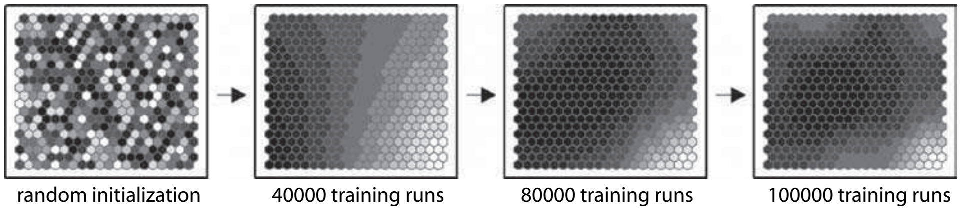

Figure 1.7 *Changes to the component plane for the variable 'vacant housing' during training of a SOM with demographic data for US states. Higher values indicated by lighter shading*

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.

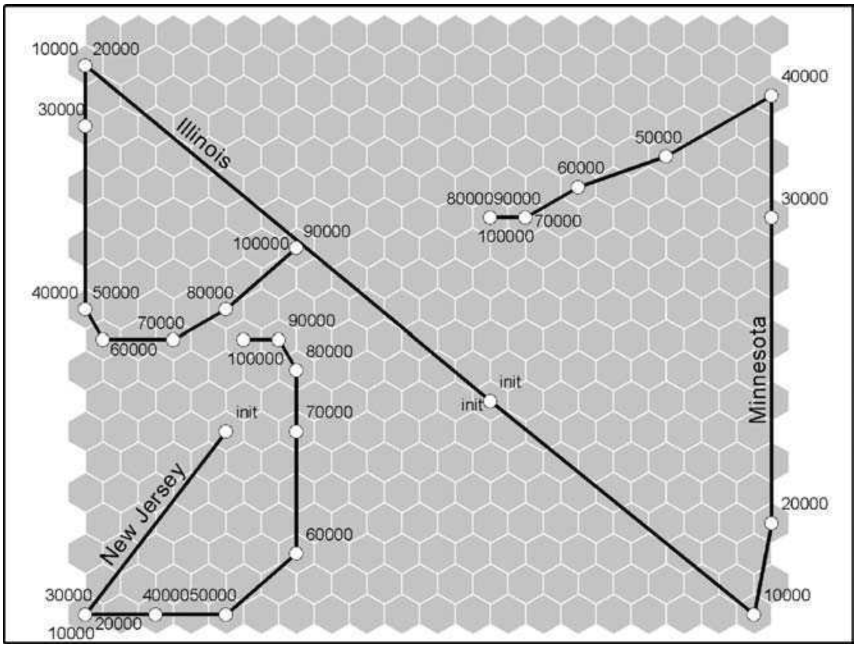

Figure 1.8 *Position changes of three states while training a SOM for a total of 100 000 runs. Starting with a randomly initialized SOM, each state is recorded once every 10 000 runs*

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 (Ultsch, 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