Self-Organising Maps: Statistical Analysis, Treatment and Applications

Hu Jun Yin

Submitted for the Degree of Doctor of Philosophy

University of York
Department of Electronics

March 1996

To the memory of my grandmother and grandfather.

Abstract

This thesis presents some substantial theoretical analyses and optimal treatments of Kohonen's self-organising map (SOM) algorithm, and explores the practical application potential of the algorithm for vector quantisation, pattern classification, and image processing. It consists of two major parts. In the first part, the SOM algorithm is investigated and analysed from a statistical viewpoint. The proof of its universal convergence for any dimensionality is obtained using a novel and extended form of the Central Limit Theorem. Its feature space is shown to be an approximate multivariate Gaussian process, which will eventually converge and form a mapping, which minimises the mean-square distortion between the feature and input spaces. The diminishing effect of the initial states and implicit effects of the learning rate and neighbourhood function on its convergence and ordering are analysed and discussed. Distinct and meaningful definitions, and associated measures, of its ordering are presented in relation to map's fault-tolerance. The SOM algorithm is further enhanced by incorporating a proposed constraint, or Bayesian modification, in order to achieve optimal vector quantisation or pattern classification. The second part of this thesis addresses the task of unsupervised texture-image segmentation by means of SOM networks and model-based descriptions. A brief review of texture analysis in terms of definitions, perceptions, and approaches is given. Markov random field model-based approaches are discussed in detail. Arising from this a hierarchical self-organised segmentation structure, which consists of a local MRF parameter estimator, a SOM network, and a simple voting layer, is proposed and is shown, by theoretical analysis and practical experiment, to achieve a *maximum likelihood* or maximum *a posteriori* segmentation. A fast, simple, but efficient boundary relaxation algorithm is proposed as a post-processor to further refine the resulting segmentation. The class number validation problem in a fully unsupervised segmentation is approached by a classical, simple, and on-line *minimum mean-square-error* method. Experimental results indicate that this method is very efficient for texture segmentation problems. The thesis concludes with some suggestions for further work on SOM neural networks.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

Abbreviations

AR	auto-regressive
BSOM	Bayesian SOM
BR	boundary relaxation
CL	competitive learning
CM	conditional Markov
DCT	discrete cosine transform
DFT	discrete Fourier transform
ECSOM	equidistortion constrained SOM
EM	expectation and maximisation
GD	Gibbs distribution
GMRF	Gaussian Markov random field
GRF	Gibbs random field
HSOS	hierarchical self-organised segmentation
HVS	human visual system
KLT	Karhunen-Loeve transform
LBG	Linde-Buzo-Gray (algorithm)
LHS	left hand side
LS	least square
LV	local voting
LVQ	learning vector quantisation
MAP	Maximum *a posteriori*
MD	mixture distribution
ML	maximum likelihood
MMSE	minimum mean square error
MPL	maximum pseudo-likelihood
MPM	maximum posterior marginal
MRF	Markov random field
m.s.	mean-square
MSE	mean square error
PCM	pulse-code modulation
PSNR	peak signal-to-noise ratio
pdf	probability density function.
RHS	right hand side
SAR	simultaneous autoregressive

SOM	self-organising map
SOS	self organised segmentation
VQ	vector quantisation, vector quantiser
WCA	winner-change-all
WTA	winner-take-all

Important Symbols

$a_k(n)$	time-vary real numbers
$b_{ci}(n)$	initial weight effect factor
C_k	the number of neurons in the k-th dimension of the neuron map
c	neuron index
$d(x,y)$, $d(\mathbf{x},\mathbf{y})$	distance measure (usually, Euclidean) between x and y, or $\mathbf{x}$ and $\mathbf{y}$
$h(c,v,n)$	neighbourhood function of winner v, neuron c, and time n
k_B	Boltzmann constant
L_{ij}	voting vector for pixel (i,j)
n	discrete time
$P(X)$	probability of X
$p(x)$	probability density function (pdf) of x
R, r	compression rate
T	time constant or temperature
t	time
$v(n)$	winning neuron at time n
$U(\mathbf{x})$	energy function of $\mathbf{x}$
$\mathbf{w}_c$	weight vector of neuron c
w_{ci}	synaptic weight of synapse i to neuron c
$\mathbf{X}$, $\mathbf{X}_c$	input space, input subspace that neuron c covers
$x(t)$, $x(n)$, $\mathbf{x}(n)$	input sample or vector
$\mathbf{x}$	a random field realisation
$X(i,j)$, X_{ij}	random variable at (i,j) in a random field
$x(i,j)$, x_{ij}	pixel grey level at (i,j) in an image
$\mathbf{Y}$	output space or neuron map
Z	partition function
$\alpha(n)$	adaptive gain or learning rate
η	image neighbourhood set
Π	image window
Θ	GMRF model parameter vector
θ_{uv}	GMRF (u,v)-th parameter
ρ	noise variance parameter of GMRF model
σ^2	variance of a Gaussian distribution
Ω	image lattice

Acknowledgements

This thesis is the product of four years continuous hard work and is intended to be a contribution to the field of theories and applications of neural networks. It would not have been possible without a great deal of support and help from many people, far too many to list here. First of all, I must thank my supervisor, Professor Nigel M. Allinson, for his invaluable guidance, constructive suggestions, and vigorous proof reading of the entire work. He has been always encouraging, open-minded, and supportive. He has been of help in these and many other ways during the course of my research. Secondly, I always felt fortunate in the past four years to have worked in a friendly, helpful, and relaxed group – IEL (the former Image Engineering Laboratory). I am grateful to my colleagues for their kind help. We have had a wonderful time and great fun. It is a pity that IEL had to end. I should also express my sincere appreciation for the Overseas Research Students Award, University of York Scholarship, and UMIST Studentship, for the funding that they have provided for my study. Finally, I would like to thank my wife, Tong Zhou, for her continued support and tolerance. She has been a constant source of moral support and has given as much help as she possibly could.

Declarations

Some parts of the work presented in this thesis have appeared in the following publications.

H. Yin and N. M. Allinson, 1993, "Stochastic analysis and comparison of Kohonen SOM with optimal filters," *Proceedings of the Third IEE International Conference on Artificial Neural Networks*, pp. 182-185.

H. Yin and N. M. Allinson, 1993, "On the distribution of feature space in self-organising mapping and convergence accelerating by a Kalman algorithm," in *New Trends in Neural Computation*, J. Mira, J. Cabestany and A. Prieto Eds., pp. 291-296.

H. Yin and N. M. Allinson, 1994, "Self-organised segmentation for textured images," *Proceedings of the International Conference on Artificial Neural Networks*, Vol. 2, pp. 1149-1152.

H. Yin and N. M. Allinson, 1994, "Unsupervised segmentation of textured images using a hierarchical neural structure," *Electronics Letters*, Vol. 30, No. 22, pp. 1842-1843.

H. Yin and N. M. Allinson, 1994, "Self-organised parameter estimation and segmentation of MRF model-based texture images," *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 645-649.

H. Yin and N. M. Allinson, 1995, "Towards the optimal Bayes classifier using an extended self-organising map," *Proceedings of the International Conference on Artificial Neural Networks*, Vol. 2, pp. 45-49.

H. Yin and N. M. Allinson, 1995, "On the distribution and convergence of feature space in self-organising maps," *Neural Computation*, Vol. 7, No. 6, pp. 1178-1187.

Chapter 1

INTRODUCTION

1.1 Self-Organising Neural Networks

One of greatest abilities of human beings is their creativity. It allows them to invent various tools and machines to assist them to build their world and carry out their work more efficiently. Human beings are also curious about the natural world around them, the way it functions, and even about human beings themselves. Exploring the brain and understanding how it works have been one of most challenging tasks for a very long time. The invention of computers and the initial success of the earliest learning machine models in the middle of this century made us almost to believe that these questions were to be solved in a short time and that thinking machines were soon to arrive (Hecht-Nielsen 1990). However when further theoretical research on these early models of brain function (such as perceptrons) showed some crucial drawbacks (Minsky and Papert 1969), the research in the field stalled. After a resurgence in the late 1970s and early 1980s it has been progressing more prudently and deeply. The previous models have been modified and improved in various ways. Many new models and theories for describing various brain functions have been proposed in the light of current biological, physiological, psychological, and mathematical evidence. They have been, and still continuously are being, examined and improved by extensive experiments and studies.

A brief history of neural networks from its beginnings and early success to later development can be found in many important and excellent articles and books in this field (e.g. Lippmann 1987; Hecht-Nielsen 1988, 1990; Widrow and Lehr 1990; Haykin 1994; Simpson 1990). Many of them also give a broad and deep discussion on the recent achievements of neural networks and their applications in various fields. The benefits of this research have not been limited to biological and computer science, they have gone far beyond their original purpose and extent. Applications have been extended to more and more fields, such as economics, information technology, electronics and electrical engineering, medical science, civil engineering, and mathematics. In these fields, researchers have found that using a neural network approach is much simpler, more adaptive, and more efficient than using traditional methods, which often require vigorous mathematical modelling and detailed parameters to describe the target phenomenon. There is little doubt that neural networks have become an important and useful methodology and will become more popular in even more application fields. Just as computers let us work more accurately and efficiently, neural networks will let us work on more complicated problems in easier and more adaptive ways.

When most research efforts were focusing on various learning principles and the sophisticated computational power of neural networks, there was also a great deal of effort being constantly dedicated to other very important observations from living brains, i.e. self-organisation in associative memories. Even in the "quiet years" of later 1960s and early 1970s, much fundamental work continued (von der Malsburg 1973; Kohonen 1974; Grossberg 1976a, b; Amari 1972; etc.). Kohonen's self-organising map, SOM, was proposed as an associate memory model (Kohonen 1982). It is an abstracted, simplified, and developed form from many previous research findings for content-addressable memories, in particular, an earlier model of von der Malsburg (1973, Willshaw and von der Malsburg 1976). It models the mapping between sensory stimuli (mostly from the retina) and the cortex. It attempted to discover how perceived information is mapped and stored in biological memories, and the properties of such associative memories. Kohonen extended von der Malsburg's 2-dimension presynaptic field to 2-dimension postsynaptic field mapping model to a generalised M-dimension to N-dimension mapping, where $N<M$, and proposed a competitive learning law and neighbourhood conscience rule, so that the input can be mapped onto a dimension-reduced output space while preserving the topological order.

Like any other model of neural networks, research activities on the Kohonen's SOM have increased dramatically. The SOM has been shown to be an efficient and powerful model for associative memories and unsupervised learning. Tremendous efforts have contributed to this model of simple structure yet demonstrating complicated dynamic processes (e.g. Kohonen 1986, 1987, 1988, 1990, 1991, 1994; Cottrell and Fort 1986; Ritter and Schulten 1986, 1988; Ritter 1991; Ritter *et al.* 1992; Erwin *et al.* 1991, 1992a, b; Luttrell 1989a, b, 1991, 1994a, b; Bauer and Pawelzik 1992; Budinich and Taylor 1995). Although some convergence and dynamic properties have been discovered, most of them have been limited to the 1-dimensional case with rare extensions to the 2-dimensional situation. The general and complete convergence theory and learning dynamics of the algorithm still need to be produced. The optimality it can and may produce are still to be revealed. A clear understanding of the model can make it more meaningful, applicable, and suitable to its application areas. Applications of the SOMs have already been found in the areas such as pattern recognition, speech and image processing, robot control, data clustering and visualisation, and function approximation (e.g. Allinson 1990; Ritter *et al.* 1992; Kim and Ra 1995; Mao and Jain 1996; Mulie and Cherkassky 1995). It is the simple computational form of the model that makes the SOM a promising and superior alternative to related classical approaches. For example, when used as a clustering algorithm, the SOM is an adaptive and stochastic gradient descent method compared to the batch version of the k-means algorithm. Thus it is more capable of escaping local minima and does not suffer from under-utilisation problems. When used as a data compression algorithm, the SOM is also superior to the standard LBG (Linde *et al.* 1980) and competitive learning methods in these points. Furthermore, with an ordered (even if only local ordered) codebook, the SOM algorithm shows greater inherent noise resistance, which cannot be found in the methods that employ non-ordered codebooks.

An important feature of neural networks is their ability to tolerant imprecise stimuli and system errors. The topographical order found in biological memory is a natural example of such a fault-tolerant system, in the sense of producing the lowest error in response to distorted inputs. The SOM is a algorithm which is naturally able to form such a topology preserving mapping due to the effect of its neighbourhood functions. The SOM's noise-tolerant ability is beginning to be appreciated by more and more researchers (Luttrell 1989a, b, 1994a; Andrew and Palaniswami 1994). Recently the algorithm has been used to produce robust vector quantisations (Carrato 1994; Chen *et al.* 1994). However the precise and quantitative role of the neighbourhood function to the ordering process is still far from clear. Even the meaning of the ordering process has not yet been explained in detail. Such important properties are certainly worth studying and exploring. They will not only provide useful strategies for optimal coding and data compression, but also may bring many important insights into fault-tolerant memory mechanisms.

1.2 Neural Network Approaches to Image Processing

Neural networks are adaptive, parallel, and distributed information processing structures with massive, independent, and interconnected simple processing elements, termed neurons. They are intended to be trained to carry out complicated and computational intensive processing tasks, such as non-linear classifications, adaptive control, and combinatory optimisation.

Vision is one of the most complex information processing tasks. It involves a large-scale parallel processing of low-level visual perceptions, transformations, abstractions, distributions and storage, and pattern classifications. Image processing problems are often computational intensive, complex, non-linear, and parallel in nature. Since vision provides us with more than 60 percent of our perceived information of the external world, image processing has become an important part of artificial intelligence. Thus it is an important application area for neural networks. More and more applications have shown the suitability, successful, and promising future for neural networks in many aspects of image processing, such as face recognition, texture classification and segmentation, scene analysis, and image compression (Nightingale and Hutchinson 1990; Van Hulle and Tollenaera 1993; Luckman *et al.* 1995; Wright *et al.* 1995; Dony and Haykin 1995; etc.).

An important primitive of many images is the perceived texture of many surfaces within a complex image. Texture analysis provides many essential clues for image recognition and segmentation. The techniques used for texture processing have become a fundamental and important methodology in image processing and computer vision. Searching for the underlying, effective, and discriminable texture features has long been the core of texture analysis. Classical features can be considered as the various statistical quantitative measures derived directly from the image texture. They are simple and direct, and can give reasonably good discrimination for textures. Modern approaches are seeking a deeper understanding of both the underlying principles of texture formation and human perceptive function of textures. Markov random field (MRF) model based descriptions and multiresolution filtering analyses have recently emerged as important and dominant methods in texture image processing (e.g. Tuceryan and Jain 1993; Geman and Geman 1984; Chellappa *et al.* 1993; Haralick and Shapiro 1992; Zhang *et al.* 1994; Daugman 1985; Mallat 1989; Manjunath and Chellappa 1993). Apparently these two methods can exist in parallel, as both can produce good performance in practical applications.

Neural networks have also started been incorporated with both model-based and filter-based methods for texture image processing, such as texture classification (Chellappa *et al.* 1993; Shang and Brown 1992; Schumacher and Zhang 1994) and textured image segmentation (Lampinen and Oja 1989; Zhang *et al.* 1994; Dony and Haykin 1994). Neural networks' adaptive, parallel processing, and non-linear properties are beginning to make image processing more efficient, adaptable, robust, and suitable for practical applications.

1.3 The Aims of this Thesis

As Kohonen and many other researchers in the field have acknowledged, the vigorous mathematical analysis of the SOM algorithm still needs to be fully explored.

> "*Apparently the memory functions of biological organisms have been implemented in the neural realms; but in spite of extensive experimental research pursued on biological memory, it seems that many central questions concerning its functional and organisational principles have been remained unanswered*" (Kohonen 1984).

The first half of this thesis is to present a detailed investigation of the general statistical and convergence properties of the SOM, to explore its application potential relating to various criteria, and to apply appropriate modifications to the standard algorithm to enhance optimum performance for these different applications. It is hoped this work will be a significant contribution to self-organisation theory. The second half aims to demonstrate some useful applications and extensions of the SOM for some pattern recognition and image processing problems, such as vector quantisation, data classification, and image texture segmentation. A detailed application structure of the SOM, in combination with other methods, is proposed for the unsupervised segmentation of textured images. The chapters of this thesis and their objectives are briefly described below:

In Chapter 2, a brief review of the SOM's derivation and formation, and a clear understanding of the functionality of the algorithm are first presented. The statistical and convergence properties of the features of the SOM algorithm are then studied from a statistical viewpoint. The effect of initial weights on the state of the final map of the algorithm is investigated. The algorithm's objectives and their meanings relating to what kind of optimality are discussed. Some representative examples are used to support this study and to provide some useful guidelines on implementation and application of the algorithm.

Chapter 3 presents some useful treatments of the SOM algorithm in order to achieve optimum performance in two major application areas of the algorithm: namely, vector quantisation and pattern classification. Some advantages of the SOM over traditional methods are revealed and examined. A constrained SOM is proposed to yield a vector quantisation with the global distortion minimum (or an improved local minimum). Another modification is proposed aimed at producing optimal Bayesian results when applied to unsupervised classification and clustering problems. The noise-tolerant ability of the SOM through its topological orders is then discussed and clearly explained for optimal vector quantisations. The definitions of ordering and their practical meaning are explored. This chapter also opens up a discussion on the role of the neighbourhood function in achieving a global optimum mapping and ordering.

A broad review on image textures and their description is presented at the beginning of the Chapter 4. Existing approaches to texture classification are briefly reviewed. Model-based approaches, mainly Markov random fields and Gibbs distributions, are then extensively examined. The simplicity and usefulness of these models are explained. The key operational problems to the model-based approach, i.e. estimation of model parameters, are also examined in detail. This forms an introduction to the next chapter.

Chapter 5 develops in a step by step approach a system for the unsupervised segmentation of textured images by using the SOM algorithm and model-based descriptions. The local property of textures and the convergence property of the SOM are incorporated. The possibility and suitability of each stage are examined carefully and logically. A novel hierarchical self-organising structure, and an extended version, are proposed for the unsupervised segmentation of textured images. The theoretical analysis of the optimality of these approaches is also presented. By using a local energy comparison scheme, a boundary relaxation method is proposed to improve the accuracy of the segmentation at texture boundaries. A simple on-line validation scheme in terms of minimum mean-square-error is finally proposed for the case when the class number is unknown and needs to be validated. The proposed segmentation structures and validation method are examined by extensive experiment on various textured images, to demonstrate their suitability.

The final chapter, Chapter 6, briefly reviews the contents of this work, summarise the major results and contributions, and discusses further potential research topics.

Chapter 2

STATISTICAL ANALYSIS OF SELF-ORGANISING MAPS

In this chapter a detailed investigation of the statistical and convergence properties of Kohonen's self-organising mapping algorithm of any dimension is presented. The feature (or weight) space of the algorithm is considered as a cumulation of random variables. We extend the Central Limit Theorem to a particular case, which is then applied to prove that the feature space during the learning is an approximation to a multiple Gaussian distributed stochastic process and will eventually converge, in the mean-square sense, to the probabilistic centres of the input subsets to form a quantisation mapping with a minimum mean squared distortion with a local or global topological ordering. The difficulties in dealing with the implicit dependence of the neighbourhood function on the winning neurons have been overcome in this analysis. As the training progresses, the diminishing effect of the initial values of the weights on the values of the final map is shown. The convergence conditions have been analysed both theoretically and experimentally. The effects of the learning rates on the convergence speeds and ordering have been analysed. Several useful guidelines for setting algorithm parameters in order to obtain good mappings are also provided.

2.1 Introduction

For many years, artificial neural networks have been used to model information processing systems based on natural biological neural structures. They not only may provide solutions with improved performance when compared with traditional problem-solving methods, but also give a deeper understanding of human cognitive abilities. Among the various existing neural network architectures and learning algorithms, Kohonen's self-organising map (SOM) model (Kohonen 1982) is one of most popular neural network models. It is an unsupervised learning algorithm with simple structures and computational forms, and is inspired by the biological retina-cortex mapping. Thus it can provide topologically preserved mapping from input to output spaces. Kohonen (1990) has provided a comprehensive review of this form of network. Although the model's computational form and structure are very simple, numerous researchers have already examined the algorithm and many of its problems; and research in this area goes deeper and deeper, there are still many aspects to be exploited. Even a most general theory of this algorithm is far from complete and lacking in vigorous mathematical explanation, as Kohonen (1984, 1991) and other researchers have remarked (Lo and Bavarian 1991; Erwin *et al.* 1992a, b; Bauer and Pawelzik 1992; etc.). The aim of this chapter is to provide a strict mathematical analysis of the learning process of the algorithm and so to determine important guidelines for how to correctly and properly implement the network.

As the reasons for the convergence and self-ordering phenomena are very subtle, the convergence and ordering of the SOM have been proved by Kohonen (1984) and Cottrell and Fort (1986) only in the simplest case, i.e. one-dimensional array of neurons in response to a one-dimensional input space with a one-step neighbourhood function. Erwin *et al.* (1991, 1992a, b) have extended the proof of this Kohonen chain's ordering and convergence from the one-step neighbourhood function to any convex neighbourhood functions centred at the winning neuron. They have also shown that non-convex neighbourhood functions may cause the existence of metastable states. Lo and Bavarian (1991) have analysed the effects of stepped and Gaussian type neighbourhood functions on the ordering of the SOM. They have given a comparison of both through simulations on two-dimensional arrays of neurons. However for higher dimensional cases or for mappings from a high dimensional input onto a low dimensional output, the convergence and the ordering remain very difficult to examine and explain. By considering the SOM's Markovian properties, Ritter and Schulten (1988) have derived a Fokker-Planck equation to describe the transitional properties of distribution function of the feature space in the vicinity of equilibrium. Luttrell (1989a, b) has related hierarchical vector quantisation principles to the SOM algorithm and has shown that the latter with a neighbourhood is a stochastic gradient descent method which minimises the mean squared distortion including the effect of code noise. For mapping from a high dimensional space to low dimensional data, Allinson (1990) has given some examples of mapping patterns. Recently, Bauer and Pawelzik (1992) have proposed the use of topographic products to measure the neighbourhood preservation or violation in the map. Further analysis of one-dimensional SOM is still being undertaken (e.g. Thiran and Hasler 1994).

Although developed from a different background, the SOM and stochastic approximation algorithms (e.g. Robbins-Monro 1951, cited in Sakrison 1966) bear some similarities in their computational form. The SOM looks like a multivariate stochastic approximation with the extra consideration of topographical ordering. Therefore, the basic convergence conditions for the adaptation gain (or learning rate) are the same (Kohonen 1984, 1994). These conditions have been relaxed by Ritter and Schulten (1988).

In this chapter, first a brief review on the neurobiological background of the SOM is presented. The development from von der Malsburg's model to Kohonen's model is described from this background. Then, Kohonen's algorithm is clearly stated and rewritten in a non-recursive form for our subsequent analysis, and this shows that each feature consists of two parts – the contribution from the initial states and the contribution from the input data. In Section 2.3, analysis of the diminishing effect of initial states, as the training progresses, on the *value* of the feature space, is shown. The limited topological effect of initial states is also examined.

In Section 2.4, we analyse the learning dynamics of the SOM algorithm by using probability theories to consider each neuron's weight or feature as a stochastic process, which is composed of random variables weighted by time-varying scalars. The contribution from the input space to each feature is proved, through an extended form of the Central Limit Theorem, to tend to a multiple Gaussian process and to converge in the mean-square (m.s.) sense to the probabilistic centre of each input subset. The dynamic properties of the neighbourhood function in the SOM algorithm are very important and are the key to topologically ordered mapping, but explicit and strict mathematical analysis of the effects of this function on the convergence and ordering of the process has long proved to be extremely difficult because of its implicit relationship with the winning neurons. This problem, however, has been overcome in our analysis. Our proof of convergence also formally reveals the algorithm's potential optimality for vector quantisation (VQ) as it will eventually match the two necessary conditions of an optimal VQ, although local minima may exist.

Section 2.5 provides an analysis of the relationship between convergence speeds and ordering of the algorithm with learning rates. Some useful guidelines, for selecting the model parameters in order to achieve a well converged and ordered map, are also presented, together with a discussion on possible ordering results and definitions of the ordering. Formal discussion on the definition, measurement, and realisation of ordering from optimisation theory aspects will be given in next chapter.

2.2 Kohonen Self-Organising Map and Its Rewritten Form

2.2.1 The neurobiological background: Form Malsburg's model to Kohonen's model

Understanding the principles of information processing in the brain, and then formulating them in mathematical forms, are one of the most demanding challenges in neurobiological studies. Humans have long been fascinated by our complex, remarkable, and powerful brains, which none of today's computers can compare with in so many aspects. Tremendous efforts have been applied in this research area and numerous results have been obtained. Gradually, the mysteries of the brain are being uncovered. Stimuli from the outside world are received by various sensory or receptive fields (e.g. visual-, auditory-, motor-, or somato-sensory), coded or abstracted by the living neural networks, and projected through axons onto the cerebral cortex, often to distinct parts of cortex. In other words, the different areas of the cortex (cortical maps) correspond to different sensory inputs. Topographically ordered maps have been widely observed in the cortex. The main structures (primary sensory areas) of the cortical maps are established before birth (cited in Willshaw and von der Malsburg 1976; Kohonen 1984; etc.), in a predetermined topographically ordered fashion (maybe we can call this the global order of the cortex). Other more detailed areas (associative areas), however, are developed through self-organisation gradually during life and in a topographically meaningful order (maybe this can be termed local ordering). Therefore studying such topographically ordered projections (both global and local), which had been ignored during the early period of neural information processing development (Kohonen 1986), is clearly important for forming dimensionality-reduction mapping and for the effective representation of sensory information and feature extraction.

The self-organised learning behaviour of brains has been studied for a long time by many people. Many pioneering works, e.g. Hebb's learning law (1949), Marr's theory of the cerebellar cortex (1969), Willshaw *et al.*'s non-holographic associative memory (1969), Gaze's studies on nerve connections (1970), von der Malsburg and Willshaw's self-organising model of retina-cortex mapping (von der Malsburg 1973, Willshaw and von der Malsburg 1976), Amari's mathematical analysis of self-organisation in the cortex (1980), Kohonen's self-organising map (1982), Cottrell and Fort's self-organising model of retinotopy (1986), still have a great influence on today's research. (Since we are concerned with self-organising maps rather than other models, many excellent pioneering works, such as McCulloch and Pitts (1943), Rosenblatt (1958), Widrow (1962), Amari (1967), Anderson (1968), Minsky and Papert (1969), Fukushima (1975), Grossberg (1976a, b), Sejnowski (1976), Hopfield (1984), Rumelhart and Mcclelland (1986), will not be discussed here). von der Malsburg (1973) and Willshaw (1976) first developed, in mathematical form, self-organising topograghical mappings, mainly from two-dimensional presynaptic sheets to two-dimensional postsynaptic sheets, based on retinatopic mapping: the ordered projection of visual retina to visual cortex (see Fig. 2.1). Their basic idea was:

......the geometrical proximity of presynaptic cells is coded in the form of correlations in their electrical activity. These correlations can be used in the postsynatic sheet to recognise axons of neighbouring presynaptic cells and to connect them to neighbouring postsynaptic cells, hence producing a continuous mapping......

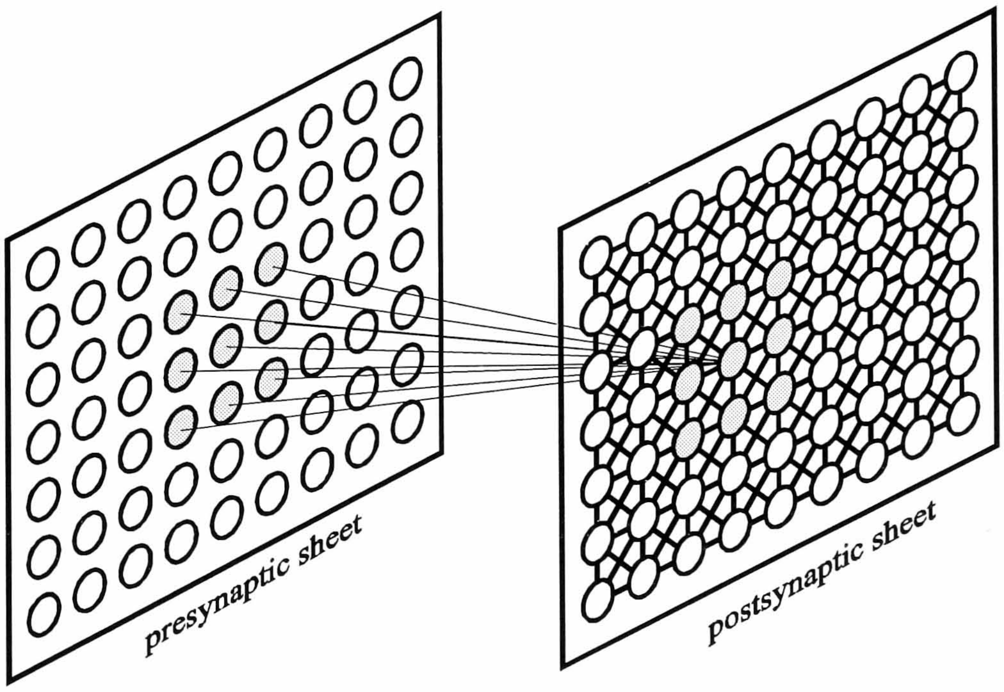


Figure 2.1: von der Malsburg's self-organising map model.
Local clusters in a presynaptic sheet are connected to local clusters in a post-synaptic sheet. There are lateral interconnections within the postsynaptic sheet (solid lines are used to indicate such connections).

The model uses short-range excitatory connections between cells so that activity in neighbouring cells becomes mutually reinforced, and uses long-range inhibitory interconnections to prevent activity from spreading too far. The postsynaptic activities $\{y_j(t), j=1, 2,...N_y\}$, at time t, are expressed by

$$\frac{\partial y_j(t)}{\partial t} + cy_j(t) = \sum_i w_{ij}(t)x_i*(t) + \sum_k e_{kj}y_k*(t) - \sum_k b_{kj}y_k*(t) \tag{2.1}$$

where c is the membrane constant, $w_{ij}(t)$ is the synaptic strength between cell i and cell j in pre- and post-synaptic sheets respectively, $\{x_i*(t), i=1, 2,...N_x\}$, the state of the presynaptic cells, equals to 1 if cell i is active or 0 otherwise, and e_{kj} and b_{kj} is short-range excitation and long-range inhibition constants respectively. $y_j*(t)$ is an active cell in postsynaptic sheet at time t. The postsynaptic cells fire if their activity is above a threshold, say θ,

$$y_k*(t) = \begin{cases} y_k*(t) - \theta & \text{if } y_k*(t) > \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

The modifiable synaptics between pre- and post-synaptic sheets are then facilitated in proportion to the product of activities in the appropriate pre- and post-synaptic cells (according to a verbal form of Hebbian learning):

$$\frac{\partial w_{ij}(t)}{\partial t} = \alpha x_i(t)y_j*(t) \tag{2.3}$$

where α is a small constant representing the organising speed.

To prevent the synaptic strengths becoming unstable, the total strength associated with each postsynaptic cell is limited by renormalisation to a constant value S after each iteration:

$$\frac{1}{N_x}\sum_{i=1}^{M} w_{ij}(t) = S \tag{2.4}$$

Kohonen (1982) abstracted the above self-organising learning principles and functions and proposed a much simplified learning mechanism which cleverly incorporates the Hebb's learning rule and neural lateral interconnection rules and can emulate the self-organising learning effect. As Ritter, Martinetz and Schulten commented in their book (1992):

> *......Kohonen's model of self-organising maps represented an important abstraction of earlier model of von der Malsburg and Willshaw; the model combines biological plausibility with proven applicability in a broad range of difficult data processing and optimization problems......*

In Kohonen's model, the postsynaptic activities are similar to Eqn. (2.1). To find the solutions of this equation and ensure they are non-negative properties, a sigmoid-type nonlinear function is applied to each postsynaptic activity:

$$y_j(n+1) = \sigma(\mathbf{w}_j^T(t)\mathbf{x}(t) + \sum_k h_{kj}y_k(t)) \tag{2.5}$$

where h_{kj} is similar to e_{kj} and b_{kj}, the input is described as a vector as the map can be extended to any dimensional input. A typical mapping is shown in Fig. 2.2.

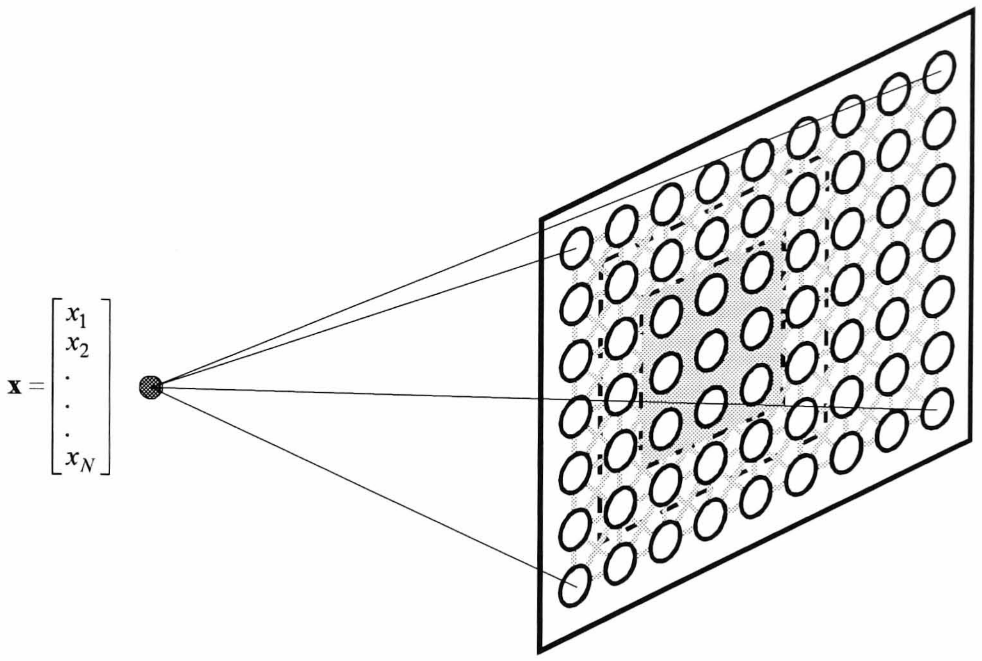

$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$

Figure 2.2: Kohonen's self-organising map model.
The input is connected to every cell in the postsynaptic sheet (the map). The learning makes the map localised, i.e. different local fields will respond to different ranges of inputs. The lateral excitation and inhibition connections are emulated by a mathematical modification, i.e. local sharing, to the learning mechanism (so there are no actual connections between cells, or in a sense we can say the connections are virtual. Hence grey lines are used to indicate these virtual connections).

A spatially-bounded cluster or *bubble* will then be formed among the postsynaptic activities and will stabilise at a maximum (without loss of generality which is assumed to be unity) when within the bubble, or a minimum (i.e. zero) otherwise,

$$y_j(t+1) = \begin{cases} 1 & \text{if neuron } j \text{ is inside the bubble}, \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

The bubble is centred on a postsynaptic cell whose synaptic connection with the presynaptic cells is mostly matched with the input or presynaptic state, i.e. the first term in the function in Eqn. (2.5) is the highest. The range or size, denoted $\eta(t)$, of the bubble depends on the ratio of the lateral excitation and inhibition.

In modifying the Hebbian learning rule, i.e. Eqn. (2.3), instead of using the form of Eqn. (2.4), a forgetting term $-\beta y_j w_{ij}$ is added to Eqn. (2.3). Let $\alpha=\beta$, and apply the function (2.6), the synaptic learning rule can then be formulated as

$$\frac{\partial w_{ij}(t)}{\partial t} = \alpha y_j(t)x_i(t) - \beta y_j(t)w_{ij}(t) = \alpha[x_i(t) - w_{ij}(t)]y_j(t) = \begin{cases} \alpha(x_i(t) - w_{ij}(t)), & \text{if } j \in \eta(t), \\ 0, & \text{if } j \notin \eta(t). \end{cases} \tag{2.7}$$

At each time the best matching postsynaptic cell is chosen according to the first term of the function in Eqn. (2.5), which is the inner product, or correlation, of the presynaptic input and synaptic weight vectors. When normalisation is applied to the postsynaptic vectors, as it usually is, this matching criterion is similar to the Euclidean distance measure between the weight and input vectors. Therefore the model provides a very simple computational structure.

The lateral interconnection between neighbouring neurons and the "Mexican-hat" excitatory or inhibitory rules are simulated (mathematically) by a simple local neighbourhood excitation form centred on the winner. Thus the neuron's lateral interconnections (both excitatory and inhibitory) have been replaced by neighbourhood function adjustment. The neighbourhood function's width can simulate the control of the exciting and inhibiting scalars. The constrained (with a decaying or forgetting term) Hebbian learning rule has been simplified and becomes a competitive learning model. The detailed form of Kohonen network will be given in the next subsection.

Most of Kohonen's work has been in associative memories (Kohonen 1972, 1973, 1974, 1980, 1982, 1984, 1986, 1988, etc.). In his studies, he has found that the spatially ordered representation of sensory information in the brain is highly related to the memory mechanism, and that the inter-representation and information storage can be implemented simultaneously by an adaptive, massively parallel, and self-organising network (Kohonen 1986). This simulated cortex map, on one hand can perform a self-organised search for important features among the inputs, and on the other hand can arrange these features in a topographically meaningful order. This is why the map is sometimes termed the *self-organising feature map*, or SOFM. In this thesis, however, it will be refereed to the self-organising map (SOM), which comes from Kohonen's original definition and purpose (i.e. associative memory).

2.2.2 The Kohonen SOM algorithm

The SOM algorithm uses a set of neurons to form a topology conserving (partially or globally) discrete mapping of the input space. Let $\mathbf{X} \in \mathbf{R}^N$ represent the input space, where N is the dimension of the input space. Let $\mathbf{Y}$ represent the neural network or *map*, which is arranged in a M-dimensional space (usually M=1, 2, or 3), so $\mathbf{Y}$ is a $C_1 \times C_2 \times \ldots C_M$ array, where $\{C_j, j=1, 2,\ldots M\}$ represents the number of neurons along each dimensional side of the neuron space, and $C=C_1 \times C_2 \times \ldots C_M$ is the total number of neurons. Every neuron or cell, $c \in \mathbf{Y}$, is connected, in parallel, to all dimensional components of the input sample, $\mathbf{x} \in \mathbf{X}$,

$$\mathbf{x} = [x_1, x_2, \ldots x_N]^T \tag{2.8}$$

The connection strengths, or weights, are

$$\mathbf{w}_c(n) = [w_{c1}(n), w_{c2}(n), \ldots w_{cN}(n)]^T \qquad \forall c \in \mathbf{Y} \tag{2.9}$$

where n is the discrete time and $n \geq 0$.

The initial weights are normally set to small random values (Lippmann 1987). The only restriction to the initial states has been stated that they should be different (Kohonen 1990, Haykin 1994). During the evolution of the weight updating, at every training step, an input sample, $\mathbf{x}(n)$, is randomly selected from the input space $\mathbf{X}$, and presented to the network. Every neuron compares its weights with the input, and a winning neuron, $v(n)$, which is said to be the "best match" with the input (i.e. closest to the current input in the Euclidean distance sense), can be found through

$$v(n) = \arg\min_{c \in \mathbf{Y}} \{\|\mathbf{x}(n) - \mathbf{w}_c(n)\|\}, \qquad \forall c \in \mathbf{Y} \tag{2.10}$$

Then the weights are updated according to the following rule:

$$\mathbf{w}_c(n+1) = \mathbf{w}_c(n) + \alpha(n)h(c,v,n)[\mathbf{x}(n) - \mathbf{w}_c(n)], \qquad \forall c \in \mathbf{Y} \tag{2.11a}$$

or, since normally only scalar valued $\{\alpha(n)\}$ and $\{h(c,v,n)\}$ terms are employed, each dimensional component of a updating weight vector receives the same degree of modifying scalar: $\alpha(n)h(c,v,n)$. The above expression can be written in another form, namely,

$$w_{ci}(n+1) = w_{ci}(n) + \alpha(n)h(c,v,n)[x_i(n) - w_{ci}(n)], \; i=1, 2, \ldots N; \; \forall c \in \mathbf{Y} \tag{2.11b}$$

where $h(c,v,n)$ is termed the neighbourhood function. There are many types of neighbourhood function. Originally step functions were used, that is,

$$h(c,v,n) = \begin{cases} 1, & \text{if } c \in \aleph_v(n) \\ 0, & \text{if } c \notin \aleph_v(n) \end{cases} \tag{2.12}$$

where $\aleph_v(n)$ is the neighbourhood set around the winner, $v(n)$, at time n. $\aleph_v(n)$ "*should be very wide in the beginning (of the training) and shrink monotonically with time*" until the winner is the only member of the neighbourhood set; "*a good global ordering*" may then be formed (Kohonen 1990). (Note, $h(c,v,n)$ is a function of time n, cell c, and *winner* v).

The coefficients $\{\alpha(n),\ n \geq 0\}$, termed *adaptation gain,* or *learning rate,* are *scalar-valued, decrease monotonically,* and satisfy (Kohonen 1984)

$$\text{(i) } 0 < \alpha(n) < 1; \quad \text{(ii) } \lim_{n \to \infty} \sum \alpha(n) \to \infty; \ \text{ and (iii) } \lim_{n \to \infty} \sum \alpha^2(n) < \infty \tag{2.13}$$

They are similar to those used in stochastic approximation (Sakrison 1966). The third condition in (2.13) has been relaxed by Ritter and Schulten (1988) to a less restrictive one, namely, $\lim_{n \to \infty} \alpha(n) \to 0$.

If the inner product similarity measure is adopted as the matching law, i.e.

$$v(n) = \arg\max_{c \in \mathbf{Y}} \left\{ \mathbf{w}_c^T(n)\mathbf{x}(n) \right\} \tag{2.14}$$

then the corresponding weight updating will be read as (Kohonen 1990)

$$\mathbf{w}_c(n+1) = \begin{cases} \dfrac{\mathbf{w}_c(n) + \alpha(n)\mathbf{x}(n)}{\|\mathbf{w}_c(n) + \alpha(n)\mathbf{x}(n)\|} & \text{if } c \in \aleph_v(n), \\ \mathbf{w}_c(n) & \text{if } c \notin \aleph_v(n) \end{cases} \tag{2.15}$$

2.2.3 The rewritten SOM algorithm

Eqn. (2.11b) can be rewritten as a non-iterative expression,

$$w_{ci}(n+1) = w_{ci}(0)\prod_{k=0}^{n}[1-\alpha(k)h(c,v,k)] + \sum_{k=0}^{n} x_i(k)\alpha(k)h(c,v,k) \prod_{l=k+1,k<n}^{n}[1-\alpha(l)h(c,v,l)],$$

$$i=1,\, 2,\ldots N;\ \forall c \in \mathbf{Y} \qquad (2.16)$$

The first term in the above equation is the contribution from the initial state to the final feature (*value*) of the map, while the second term represents the contribution of the input data. Though one might suppose that $\{h(c,v,n)\}$ is influenced indirectly by the weights at any time, including the initial weights, and inputs, so the first term might be influenced by the inputs as well, and the second term might be also affected by the initial states. However, as we will show in the next subsection this is not so at the limit. We maintain the expression of the neighbourhood function, $h(c,v,n)$, as a function of *neuron*, *winner*, and *time* for the generality.

In most applications, only *scalar-valued* $\{\alpha(n)\}$ and $\{h(c,v,n)\}$ terms are used, this means, as shown in Eqn. (2.11b) or (2.16), that all dimensional components of a weight are (or should be) unrelated. Thus the ith component, w_{ci}, of each neuron's weight, is only influenced by the ith dimensional components, x_i, of the inputs, $\mathbf{x} \in \mathbf{X}$, even though the winner is decided by all the dimensional components of all weights and the current input.

2.3 The Effect of Initial States

2.3.1 The mathematical effect of initial states

To examine the first term of (2.16), we write

$$b_{ci}(n) \equiv \prod_{k=0}^{n}[1-\alpha(k)h(c,v,k)] = \prod_{k=0, c\in\aleph_v(k)}^{n}[1-\alpha(k)] \qquad (2.17)$$

Only if the neuron, c, is in the neighbourhood set, $\aleph_v(n)$, at time n, will its weights be modified, and the corresponding terms appear in (2.17). Let $D_c(m) \equiv$ {*the number of time intervals, or steps, for which c is not in* $\aleph_v(n)$ *beginning at time m*, m=0, 1,...n; $n \geq 0$} represent the intervals or periods between updates of neuron c's weights. For each neuron, $\{D_c(m)\}$ must be a finite number set. Then let D_{cmax}=max$\{D_c(m)\}$, which will be a finite number, otherwise c will not fire again. *We assume that there are no "dead" neurons*. Hence

$$b_{ci}(n) = \prod_{k_m=0, stepD_c(m)}^{n}[1-\alpha(k_m)] \qquad (2.18)$$

Taking natural logarithms of both sides, gives

$$\ln b_{ci}(n) = \sum_{k_m=0,stepD_c(m)}^{n} \ln[1-\alpha(k_m)] \le \sum_{k_m=0,stepD_c(m)}^{n} [-\alpha(k_m)] = -\frac{1}{D_{c\max}} \sum_{k_m=0,stepD_c(m)}^{n} D_{c\max}\alpha(k_m)$$

$$= -\frac{1}{D_{c\max}} \sum_{k_m=0,stepDc(m)}^{n} \overbrace{(\alpha(k_m)+\alpha(k_m)+\ldots\alpha(k_m))}^{D_{c\max}} \le -\frac{1}{D_{c\max}} \sum_{k=0}^{n} \alpha(k) \qquad (2.19)$$

The first inequality holds because $0<\alpha(n)<1$, the last inequality holds because $\{\alpha(k)\}$ decreases monotonically. From the second condition of (2.13), i.e. $\sum_{k=0}^{n}\alpha(k) \xrightarrow{n\to\infty} \infty$, we obtain

$$b_{ci}(n) \le \exp\{-\frac{1}{D_{c\max}} \sum_{k=0}^{n} \alpha(k)\} \xrightarrow{n\to\infty} 0 \qquad (2.20)$$

Thus the first term of (2.16) will tend to zero whatever the initial states are set to provided they are finite. So the effect of initial states on the *values* of the final states will tend to zero. The monotonically decreasing property of $\{\alpha(n), n\ge0\}$ and the first two conditions of (2.13) are necessary and sufficient conditions for this effect. These results show why the initial states of the SOM can be randomly selected. The only restriction that initial weights should be different is due to the requirement that a winner has to be chosen from them when the first input is provided. Many papers restrict the initial weights to random and small values, however no formal reason is given. The reason may be that they use the exponential series for $\{\alpha(n)\}$. Then the term (2.19) will not go to minus infinity and (2.20) will not go to zero, so the initial states will have some impact on the final states, unless they are very small. However, choosing such exponential series as learning coefficients will have serious effects on the correct mapping to the centroids of the data subsets (as we will show in Sections 2.4 and 2.5). There are some incorrect ideas about weight initialisation for SOMs (e.g. Fu 1994; Wasserman 1989).

The above results apply for step, "top hat", or squared, neighbourhood functions. They can be extended to general convex neighbourhood functions. For such a function, since $\inf\{h(c,v,n)\}\le h(c,v,n)\le \sup\{h(c,v,n)\}$, hence

$$b_{ci}(n) \equiv \prod_{k=0}^{n} (1-\alpha(k)h(c,v,k)) \le \prod_{k=0,c\in\aleph_v(k)}^{n} (1-\inf\{h(c,v,n)\}\alpha(k)) \qquad (2.21)$$

Taking logarithms of both sides gives a similar result to (2.19) except for a factor of $\inf\{h(c,v,n)\}$, which will not affect the further result, (2.20).

Convex type, such as Gaussian, functions are often used as neighbourhood functions because they have better performance for the ordering than that of step functions. Although from the above we have seen that there is a small factor, i.e. $\inf\{h(c,v,n)\}$, in Eqn. (2.21), the average $\{D_c(m)\}$ in (2.19) is also very small because this type of neighbourhood function lasts spatially longer. Thus the convergence performance will not be affected. Instead, since they can make the ordering phase of the algorithm shorter, the convergence performance may actually be improved by using proper Gaussian type neighbourhood functions.

2.3.2 The topographical effect of initial states

The above results show that the mathematical *values* of the final weights or features of the SOM will not be affected by the initial states provided the adaptation gains satisfy the convergence conditions. However, in some circumstances, the initial states may affect the order, or both the order and the value, of the final states due to inappropriate implementation of the neighbourhood function and/or the adaptation gain. All possible situations can be described by three cases. In the first one, a global topological order has been formed (if it exists), but it may appear in different arrangements (or appearances). *When the mapping is between identical dimensional spaces, the globally topographical order can be well defined and it is feasible to make a mathematical measure. However when the projection is from a high to a low dimensional space, the definition of global ordering is possible but it may be not achievable. The definition is not easy visualisable (Kohonen 1991; Bauer and Pawelzik 1992). We will address this issue briefly in Section 2.5.3, and in detail in the next chapter.* These different appearances can be called different *phases or directions*. For example, a five-neuron chain when trained to quantise a uniformly distributed interval [0, 1] may result in two different globally ordered maps in two directions, see Fig. 2.3(a) and (b) respectively. Actually they are the same. *Order* should be interpreted as the inter-relationship or inter-arrangement of neurons.

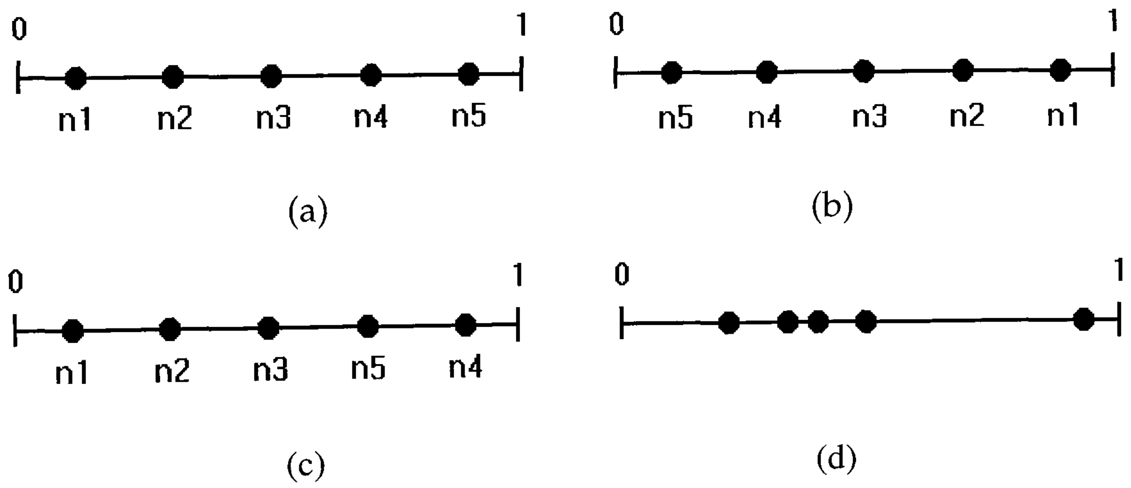


Figure 2.3: Possible mapping results of a 5-neuron SOM chain to a uniformly distributed area [0, 1]. (a) and (b) are globally ordered maps and are the same though in different direction appearances; (c) is locally ordered map; (d) is a disordered and non-optimal map.

In the second case, neurons are mapped to the correct positions; however, some local topological order, instead of the global topological order, has been achieved, see Fig. 2.3(c). The shrinking speed of the neighbourhood size might be set too high, this is probably due to unavailable measurements or monitoring of the ordering process and happens in some cases especially in high to low dimensional mapping cases. However if the adaptation gains satisfy the convergence conditions (2.13), or the far less restrictive ones of Ritter and Schulten (1988), the initial states still will not affect the *values* of the final states from the whole map point of view. The neurons will be mapped to the positions as in the ordered map, and these positions will make the distortion of representation a minimum (at least a local minimum). Thus it is still a form of optimal mapping in the sense of minimum representing errors. However, since it is not a globally ordered map, its noise tolerance to the system error will not be as high as that of the globally ordered map.

In the third case, e.g. Fig. 2.3(d), the mapping can be meaningless. Whatever the order of the map, the positions (values) of the neurons have not been mapped to the correct positions so the mapping is meaningless, i.e. not optimal in any sense. The adaptation gains might have decreased too fast, or both the adaptation gains and the size of the neighbourhood function shrunk too fast. For example, there are many attempts to use exponential functions, like e^{-n} series, as adaptation gains. However, they do not satisfy the second condition of (2.13). Thus theoretically they will not guarantee convergence though they may give a good approximation in some cases. In this situation, the initial states do affect both values and orders of the final results of the map because of inappropriate implementation of the SOM algorithm. If the adaptation gains decrease too fast, even though the size of the neighbourhood is reduced slowly, there will not be sufficient weight changing power to move neurons to the correct positions and so change their ordering unless the order is provided before the training commence or is formed at a very early stage of the training. But even if the ordering is correct (or optimal), the map is still not optimal, since the positions of the neurons are incorrect, or not optimal.

2.4 The Distribution and Convergence of the SOM Feature Space

As the effect of initial states will tend to zero, the final feature map will depend primarily on the second term of (2.16), i.e. the contribution from the input space. Since the input vectors are drawn randomly, or independently, from the input set $\mathbf{X}$, then from Eqn. (2.16) the second contribution can be treated as a time-varying weighted sum of independent random variables (r.v.s), $\{\mathbf{x}(n), n>0\}$. Each neuron receives inputs from a set, termed $\mathbf{X}_c(n)$, which is a time-varying subset of the input set $\mathbf{X}$. At the beginning of training, subsets are maximally overlapped with each other. As the training progresses and the neighbourhood size shrinks to just one neuron, the winner, input subsets $\{\mathbf{X}_c(n), c\in\mathbf{Y}, n\geq 0\}$ will eventually be mutually separated with

$$\bigcup_{c\in\mathbf{Y}}\mathbf{X}_c(n)\xrightarrow{n\to\infty}\mathbf{X}, \quad \text{and } \mathbf{X}_c(n)\bigcap\mathbf{X}_{c'}(n)\xrightarrow{n\to\infty}\phi, \quad c\neq c', \ \forall c, c'\in\mathbf{Y} \tag{2.22}$$

As time tends to infinity, $\{\mathbf{X}_c(n)\}$ will tend to $\{\mathbf{X}_c\}$, which are termed the final input subsets.

Suppose the probability density function of the input set $\mathbf{X}$ is $p(\mathbf{x})$, the probability of an input sample $\mathbf{x}(n)$ falling into a subset, $\mathbf{X}_c(n)$, is changing with time and given by

$$P(\mathbf{X}_c, n) = \int_{\mathbf{x}\in\mathbf{X}_c(n)} p(\mathbf{x})d\mathbf{x}, \qquad \forall c\in\mathbf{Y} \tag{2.23}$$

and within each input subset, $\mathbf{X}_c(n)$, the varying probability density function is

$$p_c(\mathbf{x},n) = \frac{p(\mathbf{x})}{P(\mathbf{X}_c,n)} \qquad \forall c \in \mathbf{Y} \tag{2.24}$$

As time tends to infinity, $\{\mathbf{X}_c(n), P(\mathbf{X}_c, n)$, and $p_c(\mathbf{x}, n)$, $c \in \mathbf{Y}\}$ will tend to $\{\mathbf{X}_c, P(\mathbf{X}_c)$, and $p_c(\mathbf{x})$, $c \in \mathbf{Y}\}$ respectively. So $p_c(\mathbf{x})=p(\mathbf{x})/P(\mathbf{X}_c)$.

Next we will use statistical methods to prove that in the SOM algorithm, each feature or weight component, $w_{ci}(n)$, in Eqn. (2.16), represents a stochastic process whose probability distribution density will tend to a Gaussian-type function, and furthermore its variance will tend to zero and its mean will tend to the mean of the corresponding input subset $\mathbf{X}_c$.

2.4.1 An extended Central Limit Theorem

The Central Limit Theorem is concerned with the statistical properties of a sum of independent r.v.s. The differences in the present case are that such a sum (see the second term of (2.16)) is a sum of r.v.s. weighted by *time-varying* scalars. Each variance of a weighted random variable (r.v.) will tend to zero, rather than to a finite number, because these time-varying scalars tend to zero. In the following, we will show that the variance of the sum of these weighted r.v.s will also tend to zero (otherwise the algorithm will not converge). We cannot apply directly any existing version of the Central Limit Theorem (e.g. Markov's, Liapounov's, Lindeberg's, see Papoulis 1965, Chow and Teicher 1978) to this analysis. It is necessary to extend the theorem to this particular application. We introduce an extended form of the theorem. The proof is given at the end of this chapter, in an Appendix.

Theorem 2.1: *If $\{X_n, n\geq 0\}$ are independent r.v.s with finite means of $\{m_n, n\geq 0\}$, finite variances of $\{\sigma_n^2, n\geq 0\}$, and finite higher moments, i.e. for any $\delta > 0$,*

$$\mu_n^{(2+\delta)} = \int_{X_n} X_n^{2+\delta} p(X_n) dX_n < \infty \tag{2.25}$$

where $p(X_n)$ is the density function of X_n, $\{a_k(n), k=0,1,\ldots n, n\geq 0\}$ is a set of time-varying real numbers, which satisfy

$$\text{(i) } 0 < a_k(n) < 1; \quad \text{(ii) } \sum_{k=0}^{n} a_k(n) \xrightarrow{n\to\infty} 1; \quad \text{(iii) } \sum_{k=0}^{n} a_k^2(n) \xrightarrow{n\to\infty} 0 \tag{2.26}$$

The weighted sum $\{\sum_{k=0}^{n} a_k(n) X_n, n\geq 0\}$ will tend to a Gaussian distributed process with mean (varying with time) of $\{m(n) = \sum_{k=0}^{n} a_k(n) m_k, n\geq 0\}$ and variance (varying with time) of $\{\sigma^2(n) = \sum_{k=0}^{n} a_k^2(n)\sigma_k^2, n\geq 0\}$, and with $m(n)\to E\{m_n\}$, $\sigma^2(n) \to 0$ when $n \to \infty$. Furthermore if $X_n \to X'$, then such a weighted sum will converge in the m.s. sense to m, the mean of X'.

2.4.2 The asymptotical distribution of the SOM features

Returning to the second term of (2.16), we can regard it as a time-varying weighted sum of independent r.v.s. and here the time-varying weight set $\{a_k(n), k=0, 1,...n; n\geq 0\}$ is given by

$$a_k(n) \equiv [\prod_{l=k+1,k<n}^{n}(1-\alpha(l)h(c,v,l))]\alpha(k)h(c,v,k) \tag{2.27}$$

Next we shall prove that this set will satisfy the three conditions of the above theorem, i.e. (2.26). The first condition, $0<a_k(n)<1$, holds because of (2.12) and (2.13); and the second condition holds because

$$\sum_{k=0}^{n} a_k(n) = \sum_{k=0}^{n}[\prod_{l=k+1,k<n}^{n}(1-\alpha(l)h(c,v,l))]\alpha(k)h(c,v,k)=[1\text{-}(1\text{-}\alpha(n)h(c,v,n))]$$

$$+[1\text{-}\alpha(n)h(c,v,n)][1\text{-}(1\text{-}\alpha(n\text{-}1)h(c,v,n\text{-}1))]$$

$$+\ldots\ldots$$

$$+[1\text{-}\alpha(n)h(c,v,n)][1\text{-}\alpha(n\text{-}1)h(c,v,n\text{-}1)]\ldots[1\text{-}\alpha(1)h(c,v,1)][1\text{-}(1\text{-}\alpha(0)h(c,v,0))]$$

$$=1-\prod_{k=0}^{n}[1-\alpha(k)h(c,v,k)] \tag{2.28}$$

From Section 2.3.1, we know that the second term of above will tend to zero.

For the last condition, considering

$$\sum_{k=0}^{n} a_k^2(n) \equiv \sum_{k=0}^{n}[\prod_{l=k+1,k<n}^{n}(1-\alpha(l)h(c,v,l))^2]\alpha^2(k)h^2(c,v,k) \tag{2.29}$$

Since $\sum_{k=0}^{\infty}\alpha^2(k)$ converges, so for any arbitrary small value ε, there exists a value of κ,

for which $\sum_{\kappa}^{\infty}\alpha^2(k)<\varepsilon$, and because $0<[1\text{-}\alpha(l)h(c,v,l)]<1$, then

$$\lim_{n\to\infty}\sum_{k=\kappa}^{n} a_k^2(n) = \sum_{k=\kappa}^{\infty}[\prod_{l=k+1}^{\infty}(1-\alpha(l)h(c,v,l))^2]\alpha^2(k)h^2(c.v,l) < \sum_{k=\kappa}^{\infty}\alpha^2(k)<\varepsilon \tag{2.30}$$

For a finite κ, since $\sum_{k=0}^{\infty}\alpha(k)$ diverge, $\sum_{k=\kappa}^{\infty}\alpha(k)$ will also diverge, and from Section 2.3.1,

we can see that $\prod_{l=\kappa+1}^{\infty}(1-\alpha(l)h(c,v,l))$ will also tend to zero, and since $\sum_{k=0}^{\kappa}\alpha^2(k)<\theta$, (a constant), therefore,

$$\lim_{n\to\infty}\sum_{k=0}^{\kappa} a_k^2(n) = \sum_{k=0}^{\kappa}[\prod_{l=k+1}^{\infty}(1-\alpha(l)h(c,v,l))^2]\alpha^2(k)h^2(c,v,k)$$

$$<[\prod_{l=\kappa+1}^{\infty}(1-\alpha(l)h(c,v,l))^2]\sum_{k=0}^{\kappa}\alpha^2(k) < \theta\prod_{l=\kappa+1}^{\infty}(1-\alpha(l)h(c,v,l)) \to 0 \qquad (2.31)$$

We conclude from (2.30) and (2.31) that the last condition also holds. The above results, together with the nearest neighbour matching law of the algorithm, result in the lemma given next.

2.4.3 The convergence of the SOM: A Lemma

Lemma 2.1: *The feature space of the SOM algorithm is approximate Gaussian distributed stochastic processes, and will converge in the m.s. sense to the means, or centroids, of the final input subsets, i.e.*

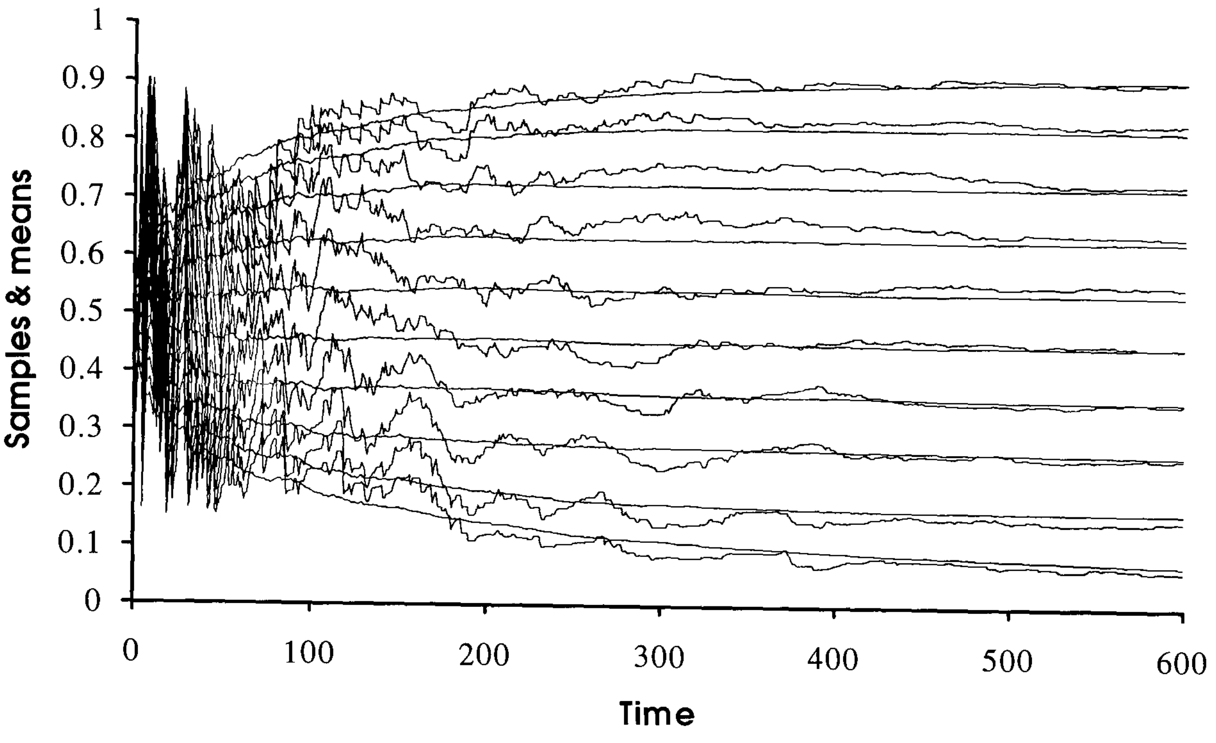
$$\mathbf{w}_c(n) \xrightarrow{n\to\infty} \mathbf{m}_c = \frac{1}{P(\mathbf{X}_c)}\int_{\mathbf{X}_c}\mathbf{x}p(\mathbf{x})d\mathbf{x}, \qquad \forall c\in\mathbf{Y} \qquad (2.32)$$

Where $\{\mathbf{m}_c\}$ *is termed the final feature space, and is the set of cluster centres of the final input subsets* $\{\mathbf{X}_c\}$. *Each final subset* $\mathbf{X}_c$ *has hyperplane boundaries which are defined by:*

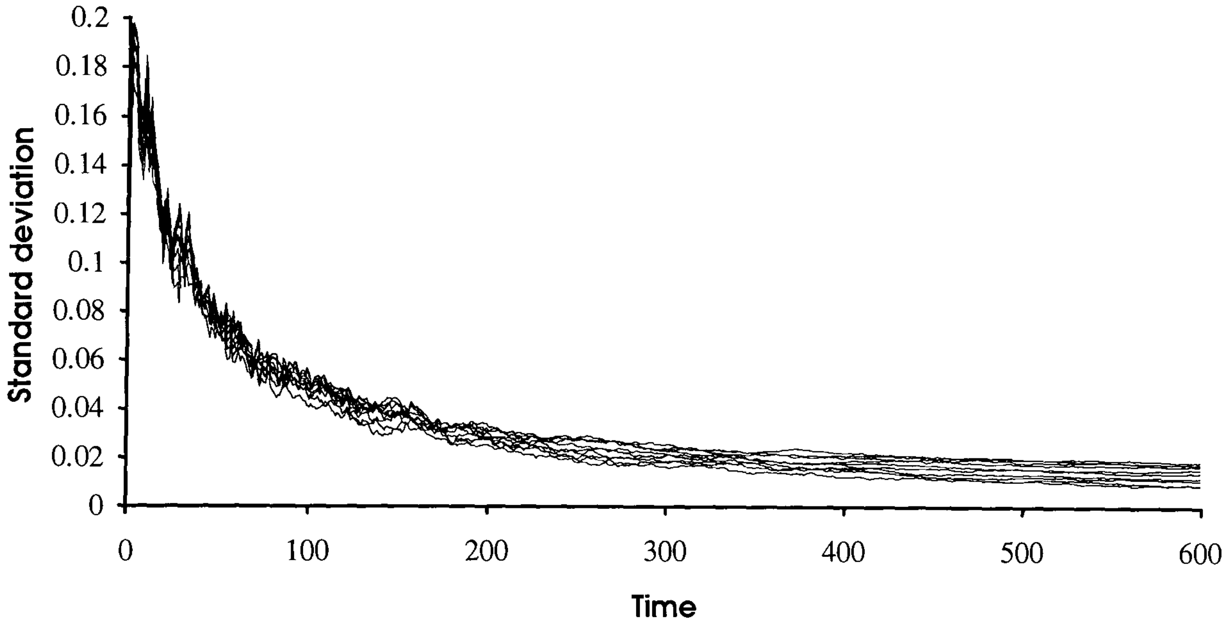
$$\|\mathbf{x}-\mathbf{m}_c\| = \|\mathbf{x}-\mathbf{m}_{c'}\|, \qquad \forall c, c'\in\mathbf{Y},\ \textit{but } c'\neq c \qquad (2.33)$$

The lemma means that the algorithm will eventually converge to positions that will meet the two well-known necessary conditions for minimising the mean squared distortion in representing input space by the map, or *quantisation error.* The SOM is naturally a multiple stochastic process, although non-stationary, but asymptotically convergent and normally distributed.

We use a simple example to conclude this subsection. A 10-neuron SOM chain is used in mapping to a uniform distributed interval [0, 1]. The parameters are set as: initial weights $\mathbf{w}(0)=0.1\times random(1)+0.5$; learning gains $\alpha(n)=0.9\times B/(B+n)$; and the step neighbourhood shrinking speed $\aleph(n)=5\times G/(G+n)\}$, where B and G are time constants and are both set to 100 in this example. The evolution random processes of the features, i.e. the weights, are shown in Fig. 2.4(a). The corresponding mean and variance processes, which are calculated from 100 independent simulations, are shown in Fig. 2.4(a) (smooth lines) and (b) respectively. Care should be taken when performing this averaging, since the results may be in different directions, as defined in Fig. 2.3 (a) and (b). When we specify neuron "0" to be the neuron whose final position is closest to the zero end of the data segment, then the averaged process for neuron "0" should be the sum over the process of the neuron whose final weight is closest to the zero end in each of these 100 trials. So when each trial begins, each of its ten processes has to be stored, and the average over this trial will not take place until the final configuration occurs.



(a)



(b)

Figure 2.4: The evolution processes or trajectories of weights of a 10-neuron chain in a uniformly distributed input. (a) Typical learning processes and averages over 100 such processes, (b) typical variance processes.

As we can see from the results, the learning process represents a multiple stochastic process, whose mean process is smooth and asymptotical, and will gradually converge to the optimal positions. The averaged variance process is also smooth and asymptotical, and will gradually go to zero. This means that, *the mean process of the SOM (instead of itself) is a gradient descent process, or in other words, the SOM is a stochastic gradient descent method.*

2.5 Convergence Conditions and Speeds

2.5.1 Ritter and Schulten's extended convergence conditions

The original convergence conditions of (2.13) agree with the ones of stochastic approximation methods. Ritter and Schulten (1988) have considered the SOM feature space as a Markov process and derived a Fokker-Plank equation to describe the time evolution of these processes in the final stage of the convergence phase. They proposed a less restrictive necessary and sufficient condition set for convergence, i.e. (of course $\{\alpha(n)\}$ still needs to decrease monotonically),

$$\text{(i) } 0<\alpha(n)<1; \quad \text{(ii) } \lim_{n\to\infty} \sum \alpha(n) \to \infty; \quad \text{(iii) } \lim_{n\to\infty} \alpha(n) \to 0 \tag{2.34}$$

i.e. the third condition of (2.13) has been replaced.

By using Theorem 2.1, we also can show that these relaxed conditions will guarantee the convergence. The proof is similar to the previous case. We simply need to examine if the variance term, i.e. (2.29), will go to zero since the other two conditions for the theorem can be easily seen to be fulfilled.

Since $\alpha(n)\to 0$, so for any arbitrary small value ε, there exists a finite T, for $\alpha(n)<\varepsilon$, $n\geq T$. Then:

$$\lim_{n\to\infty} \sum_{k=T}^{n} a_k^2(n) = \sum_{k=T}^{\infty} \alpha^2(k)h^2(c,v,k) \prod_{l=k+1}^{\infty}(1-\alpha(l)h(c,v,l))^2$$

$$< \varepsilon \sum_{k=T}^{\infty} (1-(1-\alpha(k)h(c,v,k))) \prod_{l=k+1}^{\infty}(1-\alpha(l)h(c,v,l)) = \varepsilon(1-\prod_{k=T}^{\infty}(1-\alpha(k)h(c,v,k))) \leq \varepsilon. \tag{2.35}$$

For a finite T, since $\sum_{k=0}^{\infty}\alpha(k)\to\infty$, so $\sum_{k=T}^{\infty}\alpha(k)\to\infty$, thus $\prod_{l=T+1}^{\infty}(1-\alpha(l)h(c,v,l)))\to 0$. Since

$\sum_{k=0}^{T}\alpha^2(k)<\vartheta$, a constant, thus

$$\lim_{n\to\infty} \sum_{k=0}^{T} a_k^2(n) = \sum_{k=0}^{T} \alpha^2(k)h^2(c,v,k) \prod_{l=k+1}^{\infty}(1-\alpha(l)h(c,v,l))^2 \to 0 \tag{2.36}$$

Therefore the variance will tend to zero.

2.5.2 The effect of learning rates on convergence speeds

However, different selections of the adaptation gain, or learning rate, $\{\alpha(n)\}$, under the above convergence conditions, will result in quite different convergence speeds. To make a quantitative measure of this matter. We first see how the variance processes of weights are changing in accordance with different types of learning rate.

To visualise this, we compare some typical different variance curves. For example, we choose four different decaying order series, all of which satisfy the conditions of (2.34). They are $\alpha_1(n)=n^{-1}$, $\alpha_2(n)=n^{-1/2}$, $\alpha_3(n)=n^{-1/4}$, and $\alpha_4(n)=n^{-1/10}$. To look at the effect of each set on the convergence speed, we examine the corresponding variance index of evolution process, which is similar to term (2.29) and is defined as

$$\sigma'^2_i(N) \equiv \sum_{k=1}^{N} \alpha_i^2(k) \prod_{l=k+1}^{N} (1-\alpha_i(l))^2, \qquad i=1, 2, 3, 4. \qquad (2.37)$$

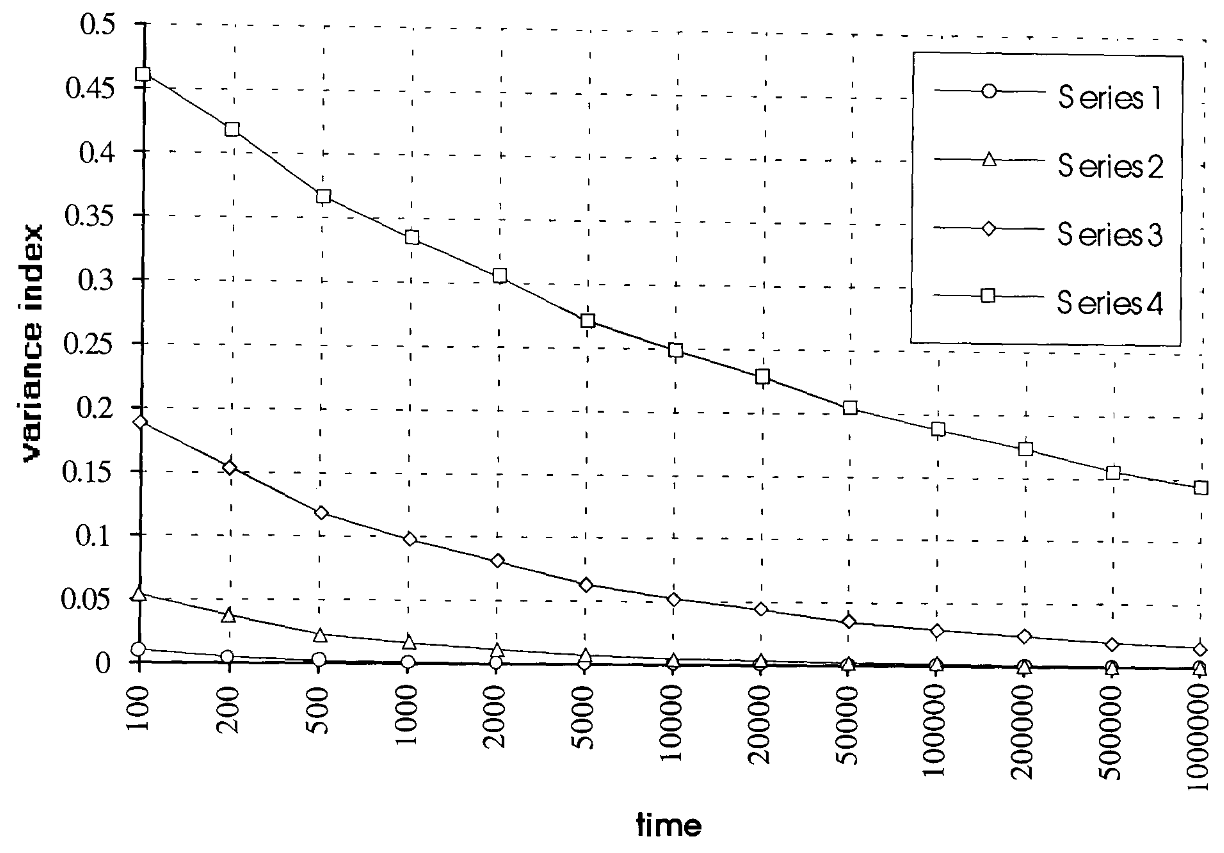


Figure 2.5: Convergence speeds.
Series $i \equiv \sigma'^2_i(N)$ (i.e. α_i(n)'s variance index), i=1, 2, 3, 4.

The results are shown in Fig. 2.5. We can see that as the rate at which α tends to zero decreases, the variation reducing rate, the most important measure of the convergence speed, will also be greatly reduced. In these examples, after 1,000 iterations, the variance index has been reduced to 0.001 for series α_1, while to 0.0162, 0.0977, and 0.3344, for α_2, α_3, and α_4 respectively. After 1,000,000 iterations, the variance indexes are 0.000001, 0.0005, 0.0161, and 0.1436, for α_1, α_2, α_3, and α_4 respectively. It seems that α_1 is better than the others. However, this depends on situations. In some cases (e.g. very many neurons, and/or high dimension data), a slow learning rate may be helpful for (i) ordering of the map and (ii) improving convergence, i.e. converging to the correct (optimal) positions earlier. A correct choice of learning rate, as well as neighbourhood function parameters, will depend on individual applications and data dimensions. In the rest of thesis, when we refer to the convergence conditions we shall mean those in (2.13), unless we specify the ones in (2.34).

Even within the same decaying rate, there is great flexibility for changing the learning rate. For example, normally we use an n^{-1} order learning rate, but in the form of (although the series $\{n^{-1}\}$ meets the convergence conditions itself, it is not used directly, as we will discuss next)

$$\alpha(n)=\alpha(0)B/(B+n) \tag{2.38}$$

where $\alpha(0)$ is initial rate and can be chosen between 0.5 to 1, and B is the rate decaying circle constant. By choosing different B, we can form a large range of learning rates.

Theoretically when the $\{\alpha(n)\}$ satisfies the convergence conditions, the neurons will *eventually* (that means as time tends to infinity!) converge to some pre-determined positions, with which the neural map provides the minimum mean squared distortion of input space. These positions depend on the number of neurons, input distribution, and dimensions of both input and neuron spaces. This means whatever we choose $\alpha(0)$ and B, they should not affect the convergence results. It will, however, affect the convergence speed indirectly by its weak ability to move neurons to the required positions. For example, if we use a small $\alpha(0)$ and/or small B to start with the training (or if the ordering phase takes long time and the convergence phase starts with a very small $\alpha(T_1)$, where T_1 is the iterations that the ordering phase has taken), it will need very many iterations to converge to the required positions. However, permitting time go to infinity is unrealistic; and after a large number of iterations, the changes in weights are very small, so it will take a long time to move those neurons whose positions are far from optimal to the correct places. Converging in a limited time is one of the most demanding tasks. The following example gives a quantitative analysis of the effect of different B in Eqn. (2.38).

A 10-neuron chain is mapped to a one-dimensional uniformly distributed area of [0, 1]. If we choose $\alpha(0)=0.8$ and $B=10$, and let us assume that the first 1,000 iterations are for ordering, i.e. $T_1=1{,}000$, then observe how much the residuals in Eqns. (2.17)-(2.20) are after 10,000 iterations. The average $D_c(m)$ in this case will be 10 because of the uniform distribution. $b_c(10{,}000)$ can be calculated from (2.18)-(2.20), i.e.

$$b_c(10,000) \le \exp(-\frac{1}{D_c}\sum_{k=1,000}^{10,000}\frac{\alpha(0)B}{B+k}) = \exp(-1.835) = 0.16 \tag{2.39}$$

which is still a large value compared with the required zero. So the initial weights, or the weight values after the ordering phase (normally far from the optimal positions), will have an influence on the map even after 10,000 iterations. In addition, the value in Eqn. (2.28) will be 1-0.16=0.84, instead of the required 1, this means the positions of neurons are still quite different from the mean values of their subsets. Therefore the map will be a certain distance from the optimal one, unless you let the process run much longer, say after 100,000 iterations ($b_c\le0.025$ and Eqn. (2.28)=0.975), or after 1,000,000 iterations ($b_c\le0.004$ and Eqn. (2.28)=0.996), or even longer.

When B is increased, however, it may be very different. For example if $B=100$, then $b_c\le1.972\times10^{-8}\cong0$ and Eqn. (2.28)$\cong$1, after 10,000 iterations (still keeping $\alpha(0)=0.8$). A result which is close to the optimal map can be expected. If the order can be formed earlier, the situation may also be different. For example, if $T_1=200$, $b_c\le0.045$ after 10,000 iterations for $\alpha(0)=0.8$ and $B=10$. So two basic rules of thumb for choosing learning parameters can be deduced: one is to make the ordering phase as short as possible, or simply initialise the neurons in a ordered fashion if you can (it is possible at least in many same dimensional mapping situations); the other is choose a sufficient value for B (the more neurons in the map, the larger B should be, since $D_c(m)$ is larger). However B should not be too large, otherwise the variance term, i.e. Eqn. (2.29) would be still very large even after a large number of iterations.

Dynamically selecting $\{\alpha(n)\}$ is also possible, but a great deal of care should be taken to guarantee convergence to the correct position. The learning rates must be in accordance with the convergence conditions. If the ordering is not important, or is already formed after a certain number of iterations, the decreasing rate of $\{\alpha(n)\}$ can be reset, or even be increased as long as it will comply with the convergence conditions. The learning rate, however, may have a stochastic relaxation effect on the convergence. A slower learning rate might give a higher possibility to escape local minima and so achieve a global optimum, when the situation is complicated, e.g. non-uniform input distribution, and/or high dimensional input data, and/or, very large number of neurons.

It may also be appropriate to use separate learning rates for different neurons. Their parameters can be set to the same, but each neuron has its own learning rate and timing, which means that each neuron's iteration number for calculating learning parameter (like n in Eqn. (2.38)), is decided by its firing frequency rather than the natural global clock. So the previously inactive neurons will not be affected by the disadvantage that a single learning rate provides, i.e. $\alpha(n)$ may be very small after many times of updating by some very active neurons. The "less fired" neurons then have a greater capacity for updating, and the more frequently fired neurons reduce their abilities to change. This interesting phenomena will be further analysed in the next chapter.

UNIVERSITY OF YORK LIBRARY

2.5.3 The effect of learning rates on topological ordering

Ordering is mainly influenced by the parameters of the neighbourhood function, specifically its shrinking rate. The slower the shrinking rate, the more likely an ordered map will be formed. The learning rate may also have a small influence on the topological ordering process. Generally large learning rates provide great capacity for adjusting disordered regions of the map when the neighbouring size has not been reduced to simply the winner. Once the ordering has been formed, it will keep the order provided convex type neighbourhood functions are used (Lo and Bavarian 1991) (This might be true only for the same dimension mapping cases, see the next chapter). To choose the learning rate we may also need to consider the local optimal situations. The more local optima, a slower relaxation speed is required. A slow learning rate does help in the ordering process when neighbourhood functions are not set correctly as when a criterion for selecting neighbourhood function is unavailable. The following examples show some limited effects of learning rate on the ordering. A formal analysis on this issue will be give in next chapter.

In the one-dimensional case: 10 neurons are mapped onto a uniformly distributed area [0, 1]. We use a Gaussian type neighbourhood function in the form of

$$h(i,v,n) = e^{-\frac{(i-v)(i-v)}{2\sigma^2(n)}} \qquad (2.40)$$

and

$$\sigma(n) = h(0)G/(G+n) \tag{2.41}$$

where $h(0)$ is the initial neighbourhood radius, G is the neighbouring radius decaying circle (time) constant.

When we set $h(0)$=3, G=10, and B (in Eqn. (2.38)) =50, then in many cases this will result in a disordered chain. However when we change B to 100 or higher, there will be an increased probability of forming a ordered chain. It is certain that if one could increase G to its correct range, a ordered map would be easily formed even when B=50 or lower. This example shows that, at least in some cases, when the correct range of neighbouring functions is not clear, slower learning rates may give higher possibilities for achieving ordered maps.

In the two-dimensional example, a 9-neuron chain is mapped onto a uniformly distributed square of [(0, 0), (0, 1), (1, 1), (1, 0)]. We choose 9 neurons because it is easy to show what are the optimal positions that will make the total distortion a minimum. In most uniformly distributed cases these positions are unique (as shown in Fig. 2.6(d) for this example). However in other cases they may not be unique, i.e. there may exist local minima. The three typical possible *Peano curves* results of this example are shown in Fig. 2.6(a), (b), and (c), corresponding to a very disordered map, a locally ordered map, and a kind of optimal ordered map respectively. Step-type neighbourhood functions were used in this example. In the first two cases, as we can assume, the neighbourhood function parameters were improperly set. The parameters were chosen as 2 and 20 for the initial neighbour radius, $h(0)$, and the neighbourhood shrinking rate, G, respectively. The neighbourhood was apparently shrinking too fast to just the winner, so that the map could not update disordered parts of the map. These parameters are normally chosen by experience, and are dependent on input and output dimensions, input probability densities, the number of neurons, etc. The Fig. 2.6(b) differs from (a) in its learning rate constant, B, which was 100 and 20 for case (b) and (a) respectively. With the parameters set as in case (b) the possibility for a better ordered map is higher than that in case (a). This shows that the learning rate in a limited range can improve the ordering. We state that Fig. 2.6(c) is a kind of globally ordered map because it does not take account of one edge, or terminal, neuron, which has violated the *order definition*, (see section 3.6). *An ordered neuron is one for which, at least, its nearest neighbouring neurons in neuron space should also be the nearest neighbouring neurons in the input metric space, i.e. neighbourhood preserving for this neuron. A globally ordered map is a map in which every neuron is a ordered neuron.* Ordering should not simply be considered as "unwrapping" of the neural positions in the map, but should be referenced to firm definitions of optimisation. Under the above definition an ordered map is more optimal than a disordered map in the sense of error tolerance, i.e. when the coding or mapping processes, and/or the transmission of the code vectors in VQ, and/or the decoding or recalling of the mapped or stored states in associative memory, involves noise, the ordered map will give the smaller distortion, or less errors, than a disordered map. In this example, we used open maps (chains), i.e. edge neurons were not wrapped or connected to other edge neurons, so the edge neurons only have one sided neighbourhood. It is very difficult (maybe impossible) to draw a totally globally ordered optimal map which can satisfy the above definitions without excluding one edge-neuron. Since this is a dimension reducing process, such distortion is to be expected.

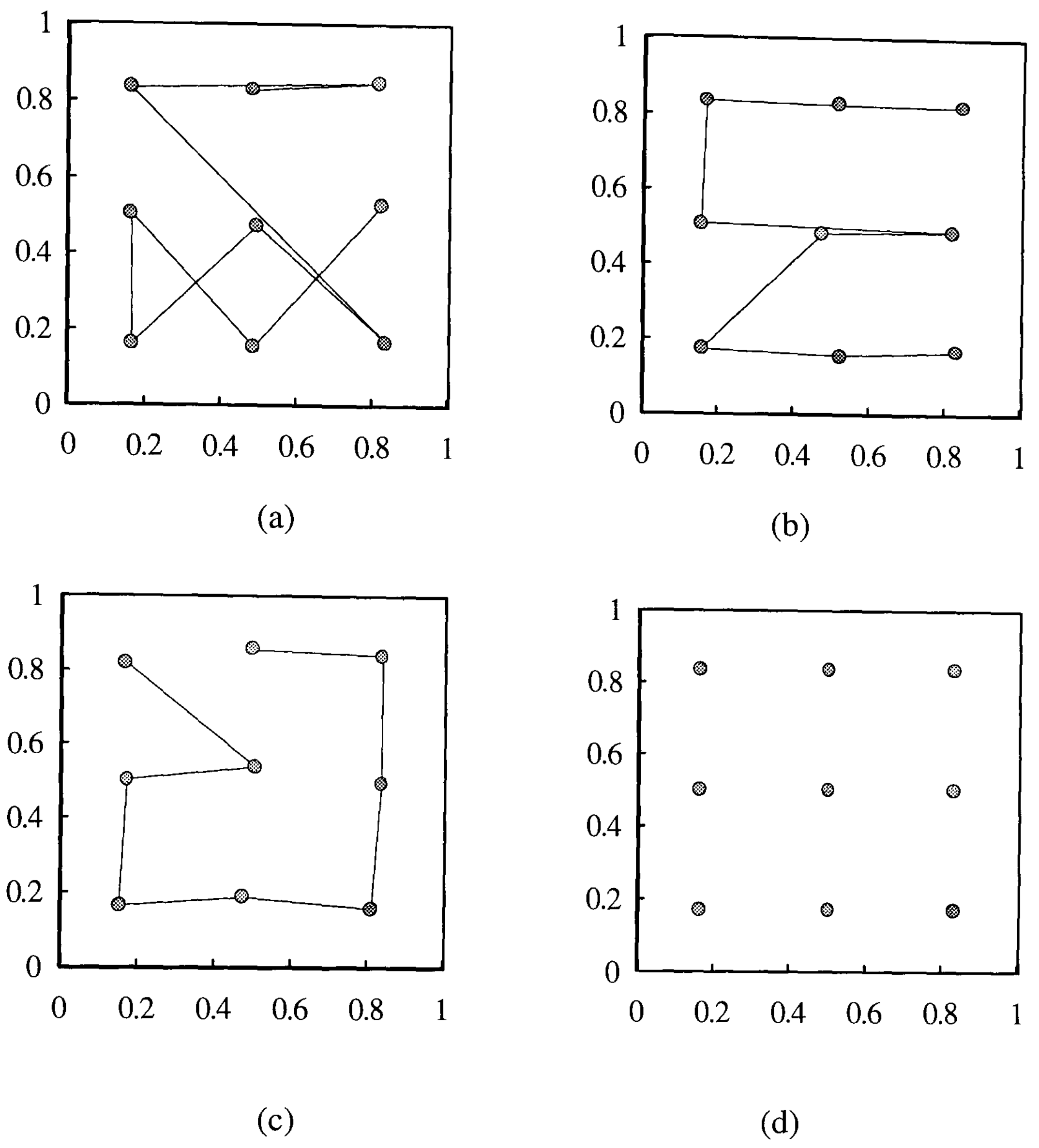


Figure 2.6: Various mapping results of a 1-D SOM chain to a 2-D input space (after 10,000 iterations). (a) very disordered; (b) locally ordered; (c) a form of globally ordering; and (d) the optimal positions for the neurons.

If we arrange these 9 neurons in two dimensions, i.e. in the same dimensional space as the input space (e.g. 3x3 grid), then a globally optimal map is obvious and can be easily obtained. However this will increase (exactly double) the total code vectors to be transmitted, or total numbers to be stored or memorised. In this example, one-dimensional mapping just needs to transmit and store 9 numbers (codebook size) and to transmit only one number for each code, while two-dimensional mapping will need to transmit and store 18 numbers for codebook and to transmit a 2-D vector, i.e. two numbers (x, y values), for each code. Of course you may number these two dimensional neurons on a scalar order, e.g. in a scanning order, that will reduce the code length for each code, but this will reduce the error tolerance ability since the 2-D neighbouring structure will be violated. So all these conditions and parameters – the number of neurons, the map dimension and shape, etc. – have to be made by considering the purpose, performance requirement and noise situations of each individual application. A trade-off may have to be made between feature representation, fault tolerance optimality, and transmission or storage efficiency.

However, whatever the map order is, the optimal positions (there may be several possible sets of such positions if there are local minima) of neurons can be predetermined; and the neurons will converge to these positions provided the adaptation gains satisfy the convergence conditions. In this example, the neuron positions in Fig. 2.6(a), (b), and (c), are close to, and are going to converge to (if time goes to infinite), these optimal positions in Fig. 2.6(d). Thus the Fig. 2.6(a) and (b) can still be optimal for noise-free transmission and/or decoding situations. We can deduce this from the mathematical analysis of the SOM presented earlier, especially Section 2.3 and 2.4, i.e. a locally or globally optimal map with local order or a global order will be formed eventually when the learning parameters satisfy the convergence conditions. The predetermined positions depend on the number of neurons (codebook size) and the distribution of input space. When the distribution is not uniform or smooth; and/or the number of neurons is small, there may not be a unique set of predetermined positions. In the next chapter we will look at these problems, and provide a kind of statistical treatment for the SOM in forming different kinds of optimal maps for different applications, such as VQs and pattern classifiers.

2.6 Conclusions

In this chapter, we have introduced the self-organising map algorithm from its historical background and have analysed the statistical properties of the feature space of the SOM algorithm. From the proof of its Gaussian distribution approximation we have also formally proved the convergence of the SOM algorithm under the original and relaxed conditions for adaptation gains. The resulting Lemma 2.1 means that the SOM algorithm will eventually minimise the mean squared distortion function. Together with its matching law, it will eventually satisfy the two necessary conditions for optimal VQ. The results are dimension irrelevant, i.e. convergence exists for any dimensional maps, provided that the learning rate complies with the convergence conditions. If the shrinking speed of neighbourhood set is not too fast, then a globally topographical ordered, or in general locally ordered, map may be formed. A clear understanding of the learning dynamics of the SOM algorithm gives some insight into an appropriate implementation and improvement of the algorithm for practical applications. Some useful guidelines for choosing the algorithm's parameters have been discussed with supporting mathematical analysis and examples. The dynamic convergence properties of the original algorithm described in this chapter will be employed in the next chapter to further analyse the optimality of the algorithm and to apply some useful constraints on, or extensions to, the algorithm in order to achieve optimal performance for VQ or pattern classification purposes.

2.7 Appendix: Proof of Theorem 2.1

First consider the zero-mean case, i.e. $\{m_n=0, n\geq 0\}$.

The following two formula can easily be obtained:

$$e^{jX} = 1 + jX - \frac{X^2}{2} + \beta\frac{|X|^{2+\delta}}{2^\delta}, \qquad \forall X\in\mathbf{R}, \quad |\beta|\leq 1 \qquad (2.a1)$$

$$1 - X = e^{-X} - \frac{\beta' X^2}{2}, \qquad \forall X\geq 0, \quad 0<\beta'<1 \qquad (2.a2)$$

In (2.a1) let $X=a_k(n)X_k\omega$ and taking the expectation of both sides, the characteristic function of $a_k(n)X_k$ is obtained:

$$\Phi_k(\omega,n) = \mathrm{E}\{e^{j\omega a_k(n)X_k}\} = 1 - \frac{a_k^2(n)\sigma_k^2}{2}\omega^2 + \beta_k\frac{a_k^{2+\delta}(n)\mu_k^{(2+\delta)}}{2^\delta}|\omega|^{2+\delta}, \qquad |\beta_k|\leq 1 \qquad (2.a3)$$

Let $X = a_k^2(n)\sigma_k^2\omega^2/2$ in (2.a2), then:

$$1 - \frac{a_k^2(n)\sigma_k^2}{2}\omega^2 = e^{-\frac{a_k^2(n)\sigma_k^2}{2}\omega^2} - \frac{\beta_k'}{2}\left(\frac{a_k^2(n)\sigma_k^2\omega^2}{2}\right)^2, \qquad 0<\beta_k'<1 \qquad (2.a4)$$

Then we can write:

$$\Phi_k(\omega,n) = e^{-\frac{a_k^2(n)\sigma_k^2}{2}\omega^2}(1+\gamma_k) \qquad (2.a5)$$

where $\gamma_k = e^{\frac{a_k^2(n)\sigma_k^2}{2}\omega^2}\left(\beta_k\frac{a_k^{2+\delta}(n)\mu_k^{2+\delta}|\omega|^{2+\delta}}{2^\delta} - \frac{\beta_k'}{2}\left(\frac{a_k^2(n)\sigma_k^2\omega^2}{2}\right)^2\right)$.

Since $a_k(n)\xrightarrow{n\to\infty}0$, thus $a_k^2(n)\sigma_k^2\omega^2 <1$ holds for any finite area of ω, and since:

$$\left(\frac{a_k^2(n)\sigma_k^2\omega^2}{2}\right)^2 = \left(\frac{a_k^2(n)\sigma_k^2\omega^2}{2}\right)^{1-\delta/2}\left(\frac{a_k^2(n)\sigma_k^2\omega^2}{2}\right)^{1+\delta/2}$$

$$< \frac{1}{2^{1-\delta/2}}\left(\frac{a_k^2(n)\sigma_k^2\omega^2}{2}\right)^{1+\delta/2} < \frac{1}{4}a_k^{2+\delta}(n)\mu_k^{(2+\delta)}|\omega|^{2+\delta} \qquad (2.a6)$$

The last inequality holds because $(\sigma_k^2)^{2+\delta} \leq (\mu_k^{(2+\delta)})^2$. So the following inequality holds:

$$|\gamma_k| < \frac{9}{8}e^{1/2}a_k^{2+\delta}(n)\mu_k^{(2+\delta)}|\omega|^{2+\delta} \qquad (2.a7)$$

Since $\{X_n, n\geq 0\}$ are independent r.v.s, then:

$$\Phi(\omega,n) = \mathrm{E}\{e^{j\omega\sum_{k=0}^{n}a_k^2(n)X_k}\} = \prod_{k=0}^{n}\Phi_k(\omega,n) = e^{-\frac{\omega^2}{2}\sum_{k=0}^{n}a_k^2(n)}(1+\gamma_1)(1+\gamma_2)\ldots(1+\gamma_n) \qquad (2.a8)$$

The error of its Gaussian distribution approximation is:

$$|\Phi(\omega,n) - e^{-\frac{\omega^2}{2}\sum_{k=0}^{n} a_k^2(n)\sigma_k^2}| < (1+|\gamma_1|)(1+|\gamma_2|)\dots(1+|\gamma_n|) - 1 < e^{|\gamma_1|+|\gamma_2|+\dots|\gamma_n|} - 1 = e^{\sum_{k=0}^{n}|\gamma_k|} - 1$$
$$< e^{2|\omega|^{2+\delta}\mu_{\max}^{(2+\delta)}\sum_{k=0}^{n} a_k^{2+\delta}(n)} - 1 \xrightarrow{n\to\infty} 0 \tag{2.a9}$$

because if $\sum_{k=0}^{n} a_k^2(n) \xrightarrow{n\to\infty} 0$, and $\delta>0$, then $\sum_{k=0}^{n} a_k^{2+\delta}(n) \xrightarrow{n\to\infty} 0$. and where $\mu_{\max}^{(2+\delta)} = \max\{\mu_n^{(2+\delta)}, n \geq 0\}$.

From (2.a9) and using a lemma of Uspensky(1937) (i.e. *if the characteristic function of r.v. S tends to the characteristic function of a Gaussian distributed variable, then the distribution of S tends to that Gaussian function*), we can conclude that $\{\sum_{k=0}^{n} a_k(n)X_k, n \geq 0\}$ tends to a Gaussian distributed process with zero mean and variance:

$$\sigma^2(n) = \sum_{k=0}^{n} a_k^2(n)\sigma_k^2 < \sigma_{\max}^2 \sum_{k=0}^{n} a_k^2(n) \xrightarrow{n\to\infty} 0 \tag{2.a10}$$

In the non-zero mean case $\{m_n \neq 0, n \geq 0\}$, if every m_n is a finite number, then the biased r.v.s $\{X'_n, n\geq0\}$ can be divided into $\{X_n+m_n, n\geq0\}$, where $\{X_n\}$ are zero mean r.v.s and according to a Corollary of Slutsky's theorem (cited in Chow and Teicher 1978) (i.e. if $\{\rho, \lambda, \rho_n, \lambda_n, n \geq 0\}$ are finite constants with $\rho_n \xrightarrow{n\to\infty} \rho$, $\lambda_n \xrightarrow{n\to\infty} \lambda$, and $X_n \xrightarrow{n\to\infty} X$, then $\rho_n X_n + \lambda_n \to \rho X + \lambda$.), the weighted sum $\sum_{k=0}^{n} a_k(n)X'_k$ is also Gaussian distributed with finite means $m(n) = \sum_{k=0}^{n} a_k(n)m_k$ ($<m_{max}$) and finite variances $\sigma^2(n)$, which will tend to zero, when n tends infinity. Furthermore, if $X'_n \xrightarrow{n\to\infty} X'$, (with the mean of m), then:

$$m(n) = \sum_{k=0}^{n} a_k(n)m_k \xrightarrow{n\to\infty} m \tag{2.a11}$$

That is $\sum_{k=0}^{n} a_k(n)X'_n$ will converge in the m.s. sense to the mean of the X'. □

Chapter 3

OPTIMAL TREATMENTS OF SELF-ORGANISING MAPS

This chapter continues the analysis of the SOM algorithm, in particular its convergence speed and stability and its suitability for various information processing tasks. The local optima and non-optimal problems are discussed. Solutions and treatments for different purposes are proposed. First, Kalman optimal filter theory is applied to reduce the asymptotical Gaussian distributed noise existing in the learning process, hence to smooth and speed up the learning. Then from the objective criteria of various information processing needs, the potential optimality of the SOM for vector quantisation is further analysed and compared. Although having great advantages over other quantisers, the SOM still suffers from local minima problems. A constrained SOM, based on the equal-distortion principle, is proposed to yield a global optimum, or near optimum, quantisation. Little extra computation costs are introduced but improved performance, both in lower distortion and in stable and fast convergence, is achieved. An explanation of noise-tolerant quantisation by using SOM related algorithms, and some meaningful and quantitatively measurable definitions of ordering, are also presented. The SOM algorithm is also widely employed as a classifier because of its simplicity and parallelism, though it is not optimal for this application. An extended SOM, in which both distance measures and neighbourhood functions have been replaced by the neuron's posterior probabilities, is proposed to achieve Bayesian classification performance when the pattern distribution is modelled as a mixture distribution and the learning is unsupervised.

3.1 Introduction

Following the statistical analysis of the self-organising map (SOM) algorithm in the last chapter, we have shown that the algorithm is potentially optimal for vector quantisation (VQ) and related data compression. However, there are two major application areas of the SOM – one is signal compression, while the other is pattern classification. Associative memories belong to the former, as they are knowledge representing methods as well. These two application tasks usually have different objectives in information processing, although sometimes they are very closely related. Directly applying SOMs to pattern classification will not normally result in optimal performance. Even when employed as a quantiser, the algorithm cannot guarantee a global optimum. How well does the algorithm perform in these two applications, or how to make the algorithm work as well as or better than conventional methods? These will be the main objectives of this chapter.

First, in Section 3.2, a Kalman filter is incorporated into the SOM algorithm as a post-filter to reduce the learning noise and so smooth and accelerate the learning process.

In Section 3.3, different optimal criteria corresponding to different information processing requirements, such as normal VQs, noise-tolerant VQs and pattern classification, are analysed. Section 3.4 first compares the use of the SOM algorithm for VQ with other VQ algorithms. Examples of image compression tasks are given. The theoretical analysis and practical examples reveal that the SOM algorithm can produce comparable performance to other compression techniques, such as the LBG (Linde *et al.* 1980), and competitive learning (CL) algorithms. It can naturally overcome some problems that other algorithms often encounter, such as under-utilisation and strong initial reference impact. However the SOM algorithm, as well as other algorithms, possess the local minima problem. A constrained SOM, an *equidistortion constrained SOM* (ECSOM), is then proposed based on an asymptotical property of optimal VQs, i.e. the equal-distortion principle. The principle is indirectly applied to the SOM by constraining the width of the neighbourhood, and this makes the principle more applicable to practical problems. The proposed ECSOM algorithm is superior to the SOM in both performance (MSE distortion) and convergence speed.

The SOM algorithm has also been widely used in data clustering for pattern classification. When used as a classifier, however, the SOM can only make reference vectors, or clustering centres, optimal with respect to the partitions, but can not produce optimal partitions or decision boundaries in the Bayesian criterion sense. Therefore a conventional unsupervised SOM will not normally produce Bayesian classification unless the input data are uniformly distributed or pattern clusters are well separated. In most cases, the pattern distributions are overlapping, and their joint distribution can be described by a mixture distribution. To produce an unsupervised Bayesian classifier, an extended SOM learning scheme is proposed in Section 3.5. In this algorithm both distance measures and the neighbourhood functions are replaced by on-line estimates of the individual neuron's posterior probabilities, so that each neuron will converge to a component of the mixture distribution. Some application examples are presented.

In Section 3.6, definitions for the ordering of the maps to any dimension have been formalised into a mathematical form. The ordering of the map is defined in two different ways: one is in geometrical sense and the other is in fault-tolerant sense. Each of these has a very clear optimisation meaning. The definitions are quantitatively measurable, can easily be used to judge the quality of the ordering of the map and can improve understanding about what are the advantages of an ordered map.

3.2 Convergence Accelerating by a Kalman Filter

3.2.1 Kalman filters

We have shown that for the SOM algorithm, the feature space is a multiple Gaussian stochastic process. These processes are time-varying, or non-stationary, but they are also asymptotic stationary and will eventually converge to finite states. This means that these processes begin with large fluctuations, or variances, but gradually the training will cause them to decrease. In this section, we combine an optimal filter, the Kalman filter, with the SOM to moderate the effects of large variances during the training process. There have been some previous examples of the application of classic optimal filter theory to neural networks (e.g., Cho and Don 1991; Ruck *et al.* 1992). A Kalman filter is a linear estimation method which can produce an optimal estimation of model states when applied to Gaussian distributed processes. It is also a recursive algorithm which is based on the prediction of system states from the latest states and linear measurements such that the expected sum of the squared errors between actual and estimated states are minimised.

The algorithm can be described as following (cited in Hostetter 1987):

System model:

$$\mathbf{S}(k+1) = \mathbf{F}(k)\mathbf{S}(k) + \mathbf{U}(k) \tag{3.1}$$

$$\mathbf{Z}(k+1) = \mathbf{H}(k+1)\mathbf{S}(k+1) + \mathbf{V}(k+1) \tag{3.2}$$

Predictor:

$$\hat{\mathbf{S}}(k+1/k) = \mathbf{F}(k)\hat{\mathbf{S}}(k/k) \tag{3.3}$$

$$\hat{\mathbf{Z}}(k+1/k) = \mathbf{H}(k+1)\hat{\mathbf{S}}(k+1/k) \tag{3.4}$$

Corrector:

$$\hat{\mathbf{S}}(k+1/k+1) = \hat{\mathbf{S}}(k+1/k) + \mathbf{K}(k+1)[\mathbf{Z}(k+1) - \hat{\mathbf{Z}}(k+1/k)] \tag{3.5}$$

Gain:

$$\mathbf{K}(k+1) = \mathbf{P}(k+1/k)\mathbf{H}^{\mathrm{T}}(k+1)[\mathbf{H}(k+1)\mathbf{P}(k+1/k)\mathbf{H}^{\mathrm{T}}(k+1) + \mathbf{R}(k+1)]^{-1} \tag{3.6}$$

Covariance:

$$\mathbf{P}(k+1/k) = \mathbf{F}(k)\mathbf{P}(k/k)\mathbf{F}^{\mathrm{T}}(k) + \mathbf{Q}(k) \tag{3.7}$$

$$\mathbf{P}(k+1/k+1) = [\mathbf{I} - \mathbf{K}(k+1)\mathbf{H}(k+1)]\mathbf{P}(k+1/k) \tag{3.8}$$

Where $\mathbf{S}(k)$ are the states to be estimated or filtered, $\mathbf{F}(k)$ is the state transition matrix, $\mathbf{U}(k)$ is the model noise vector, $\mathbf{Z}(k)$ is the observation vector, $\mathbf{H}(k)$ is the measurement matrix, $\mathbf{V}(k)$ is the measurement noise vector, and

$$\mathbf{Q}(k) = \mathrm{E}\{\mathbf{U}(k)\mathbf{U}^{\mathrm{T}}(k)\} \tag{3.9}$$

$$\mathbf{R}(k) = \mathrm{E}\{\mathbf{V}(k)\mathbf{V}^{\mathrm{T}}(k)\} \tag{3.10}$$

are the covariance matrices of model noise and measurement noise respectively.

3.2.2 Smooth and speed up the convergence of the SOM

When applying a Kalman filter to a multilayer perceptron (i.e. a non-linear system), the model of the system should be *linearised.* Cho and Don (1991) and Ruck *et al.* (1992) used an *extended Kalman filter* as an alternate training algorithm for multilayer perceptrons. Ruck *et al.* (1992) also compared it with the *back propagation* algorithm and concluded that *back propagation* is a degenerate form of the *extended Kalman filter* under the assumptions of: (1) $[\mathbf{HPH}^T+\mathbf{R}]^{-1}=a\mathbf{I}$, and (2) $\mathbf{P}=p\mathbf{I}$. In general, these conditions are not satisfied. However in the SOM, where the system model is linear, the Kalman algorithm must be applied in different way. We use a Kalman algorithm as a *post-filter* to the SOM. The updated weights of the SOM are considered as the measurements of the Kalman filter. The system models are asymptotic, thus we can use some asymptotic functions to describe them, such as exponential functions. The model states are the filtered, or true, weights of the network. Since every weight of each neuron is, or has to be, independent as we have shown in Chapter 2, the **F**, **H**, **K**, **P** matrices are all diagonal. So there will be no need for matrix computation. Furthermore, it is possible to consider every weight to have the same speed of convergence, so we can use the simplest computational form of the algorithm. A diagram is shown in Fig. 3.1.

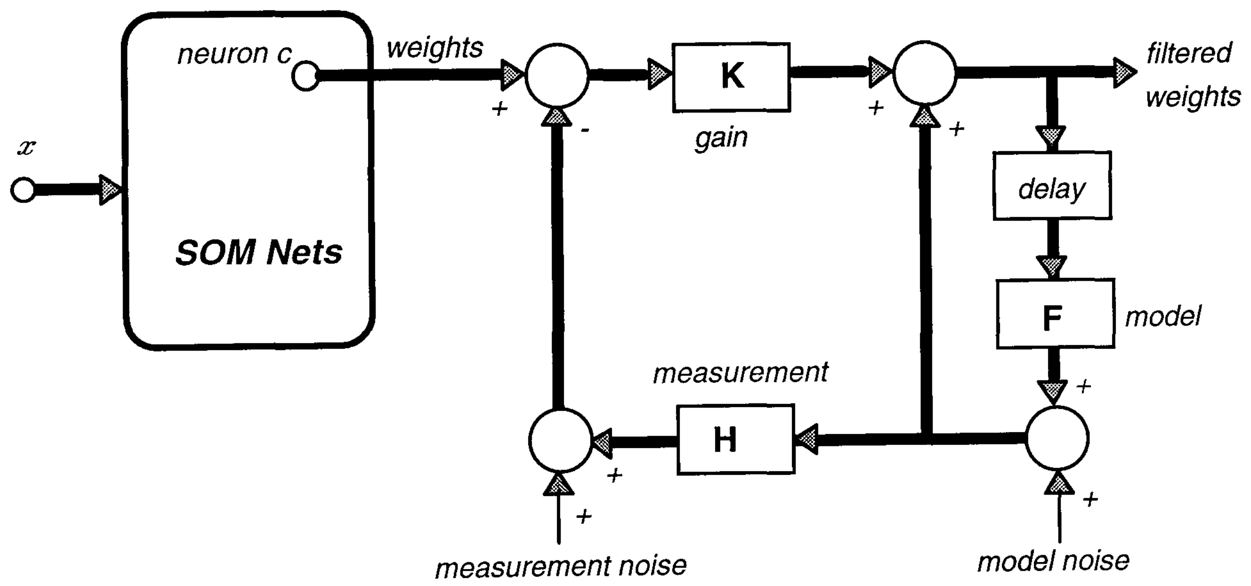


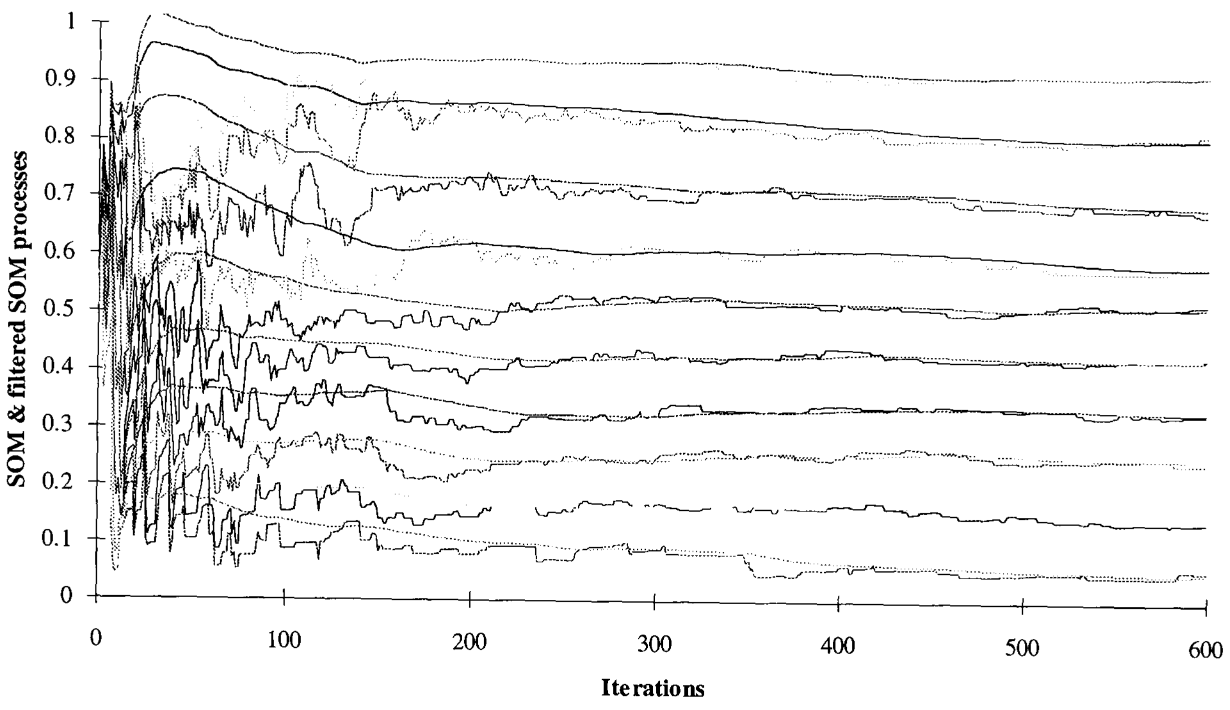
Figure 3.1: A post Kalman filter for the SOM algorithm.

For a one-dimensional (1-D) input to output mapping example, the state transition **F**, can be modelled as:

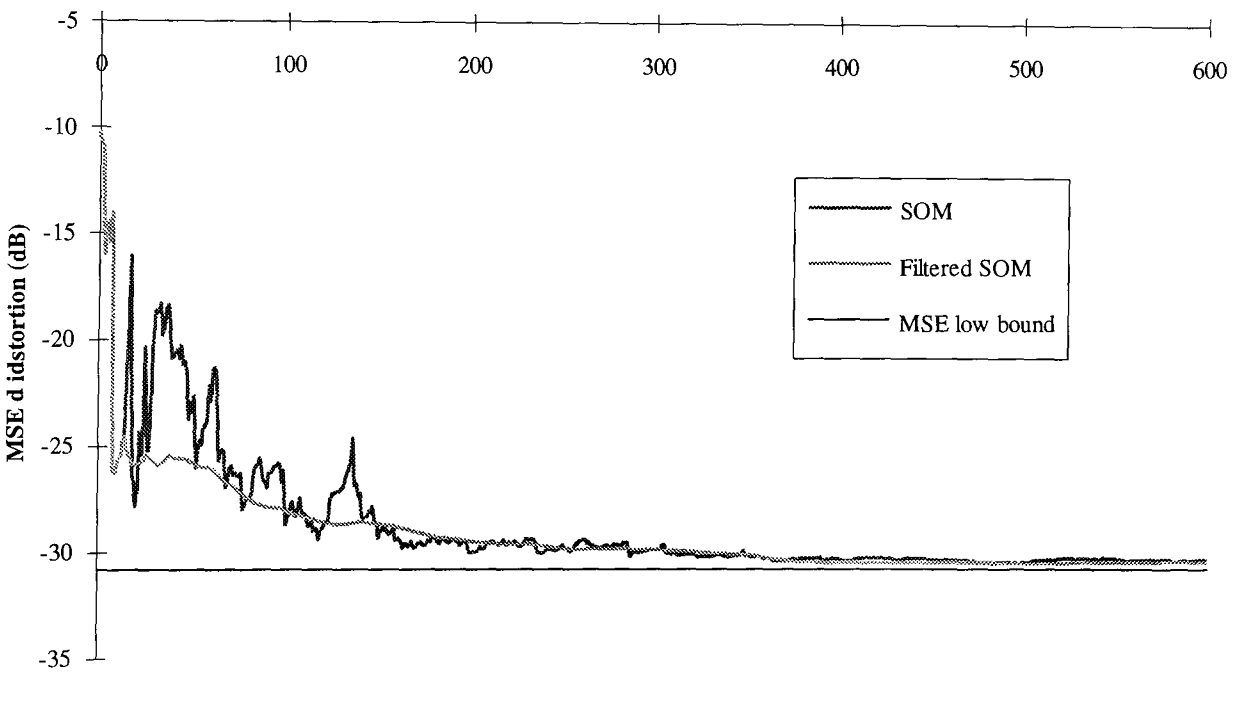
$$\mathbf{F}(k)=\frac{\partial \mathbf{S}(k)}{\partial k}=e^{\frac{g}{k(k+1)}}\,\mathbf{I} \tag{3.11}$$

Where g is the underlying converging constant, which depends on the number of neurons. Its value is not very critical and can easily be chosen experimentally.

H is identical to **I** since we treat the output of the SOM as the measurement of the filter. The noise in every neuron's state is considered to be the same scalar-value, and decreasing with time, and so is the measurement noise. Thus:



(a)



(b)

Figure 3.2: Comparison of the original SOM and Kalman Filter modified SOM (10-neuron chain and uniform distribution input case). (a) Typical SOM learning and Kalman filtered SOM learning processes; (b) The corresponding distortion level processes for both algorithms.

$$\mathbf{U}(k) = e^{-k/u}\,\mathbf{I} \tag{3.12}$$

$$\mathbf{V}(k) = e^{-k/v}\,\mathbf{I} \tag{3.13}$$

where u, v are decay constants of the state noise and measurement noise respectively.

Fig. 3.2 shows an example of applying such a Kalman filter to the SOM algorithm in the mapping of a 10-neuron chain to a uniformly distributed interval [0, 1]. The SOM and the filtered SOM evolution processes are shown in Fig. 3.2 (a), while their corresponding performances, i.e. MSE distortions (see Section 3.3), are compared in Fig. 3.2 (b). The parameters in the SOM algorithm were set similar to the example in Section 2.4.3. The Kalman filter parameters are all scalar values in the 1-D case and in this example were set: g=10, u=v=100, R=0.2×10^{-3}, Q=0.5×10^{-6}. Normally the filter has to be applied after a certain number of iterations (say 10-20) to avoid high noise at the beginning of the learning. As we can see the filter processes are much smoother and more stable than the original ones. They demonstrate low variance especially in the initial phase. Their performances show that the filtered processes can reduce the distortion earlier in the learning. Although both algorithms will reach the same distortion minimum (in this case it is – 30.79 dB) after very many iterations, more stable and faster converging algorithm can still be a great advantage.

This example shows that optimal filter theories can be applied to reduce the learning dynamic noise, which in the SOM algorithm is Gaussian distributed.

3.3 The Optimality Criteria for SOMs

3.3.1 Minimum mean squared error criterion

Since the SOM uses a finite, usually only a few, points or vectors, which though defined in the input space lie in the output space, to represent a possibly infinite or large number of points of the input space, distortion is unavoidable. Information contained in the input space has to be extracted or compressed by mapping to a limited and smaller output space. This is not a one-to-one mapping from a global point of view (although some one-to-one correspondences do exist in living brains). Most information or knowledge we learn from the outside world is captured, sorted, and assigned to some specific regions of our memory, each of which represent an abstraction of a certain knowledge domain. This learning and knowledge-building process continues throughout our entire life. Knowledge can be updated and new knowledge can be incorporated. For example, when we are learning some general domain knowledge (e.g. doors, windows, lights, cars), such learning starts from the first time we encounter such objects, and is gradually enhanced and generalised as more and more objects that have the same function are perceived either through explicit teaching or through self-learning. Thus when we consider one of these knowledge domains in general, what we are recalling from our memory is an ambiguous generalised image (not a specific one) but one with a clear meaning and functional description. This generalised knowledge is constructed by extracting representative symbols and meanings from associated objects. By doing this we can recognise a previously encountered or learnt item or identify a new one within the same class. So we reduce the amount of information stored in our brains. Imagine if we had to remember every example of door, window, car, etc., that we ever encountered, what size of brains would we need! There is no doubt, our brains can remember clearly a very large number of specific items and individual objects, like our names, friends' names, personal belongings, etc. However, extraction and generalisation for more general knowledge are so important in our cognitive processes. Such a process has been employed for a long time in information processing technology such as signal compression (e.g. signal coding) and in information storing (e.g. associative memories) in order to reduce the transmission burden /or storage space. According to Shannon's rate-distortion theory, better performance can always be achieved by coding vectors instead of scalars (Shannon 1959, cited in Gray 1984). There have been considerable efforts in vector quantisation (Linde *et al.* 1980; Gray 1984; Makhoul *et al.* 1985; Nasrabadi and King 1988). A mapping of data blocks in 2-D input space to individual reference vectors is show in Fig. 3.3. This forms the basis of VQ.

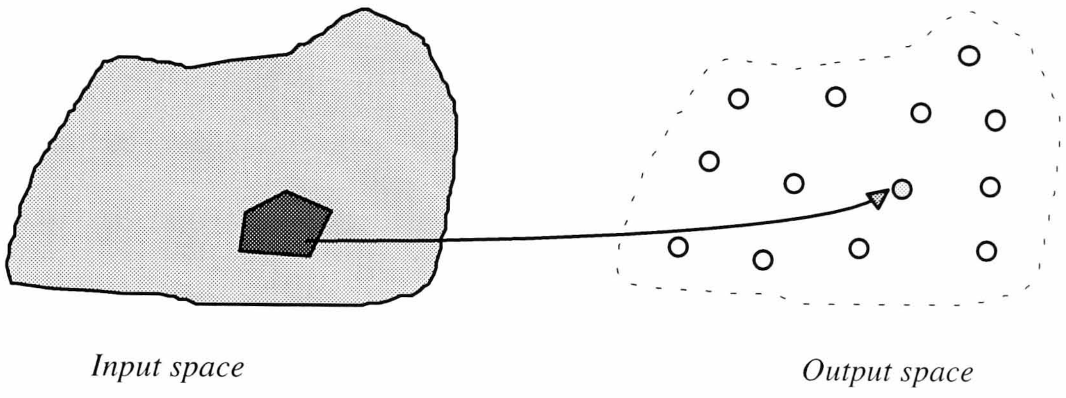


Figure 3.3: Quantising input space.

The degree of distortion caused by such extraction and generalisation reflects the soundness of the learning process. The lower the distortion, the better the learning. Although for living neural networks a measure for such distortion may not exist or cannot be expressed in a mathematical form (e.g. how could we measure the general difference between doors and windows in a precise mathematical format), such measures of distortion, if available and quantitatively comparable, would be very useful to guiding our learning, and indicating how we have learnt about various perceptual classes. Natural brains have incredible abilities to extract the most representative and meaningful features from various objects. To exploit such abilities and apply them to artificial intelligence would be very beneficial.

Since the SOM is an abstract form of mathematical modelling of the brain's information mapping mechanism, inputs to the map and its outputs are (or should be) all formally defined in mathematical forms, e.g. real values, multi-dimensional points, or vectors. If we could (!) express any object by such vectors or represent objects by extracting some distinguishing features, our next objective is to abstract these features and project them onto a representing space (like the cortex) in a strict mathematical sense. The quality of the approximation is measured by a metric distortion factor, normally the Euclidean distance. The difference and similarity between features are measured by such distances. The quality of a representative for a class is measured by the average distortion between the representing vector and the class members over the entire class range, which is given by the mean-squared error (MSE), i.e.

$$D_c = \int_{\mathbf{X}_c} \left\| \mathbf{w}_c - \mathbf{x} \right\|^2 p(\mathbf{x}) d\mathbf{x}, \qquad c \in \mathbf{Y} \tag{3.14}$$

where $\{\mathbf{x}\}$ is the input data, $\mathbf{w}_c$ is the c-th reference vector, $\mathbf{X}_c$ is the input sub-space or the data subset that reference c covers, $\mathbf{Y}$ is output or feature space, and $p(\mathbf{x})$ is the probability density function (pdf) of the input data. The integration is over all dimensional directions of the input space.

The measure, D_c, is the variance for each reference to represent its subset in the input space. The less the variance, the more accurate is the corresponding reference. (Measures using other distances, e.g. absolute Euclidean distance, Mahalanobis distance, are also possible). Individual distortions normally indicate the fitness of the individual references. Only in some cases, when the subsets are well separated (i.e. clearly clustered subsets) can their individual minimum distortions be accepted as an overall optimal criterion. In most cases, the subsets are close to each other, or even overlapped, and/or the variability of each subset is large, the individually minimising the distortions of each reference will not lead to a globally optimal mapping. The overall distortion has to be adopted as a fitness criterion. Therefore, the mapping, Φ: $\mathbf{X} \rightarrow \mathbf{Y}$, is said to be optimum (global optimum), if and only if the following overall distortion is minimised with respect to all variables, and over the entire input and output spaces:

$$D_1 = \sum_{c \in \mathbf{Y}} \int_{\mathbf{X}_c} \left\| \mathbf{w}_c - \mathbf{x} \right\|^2 p(\mathbf{x}) d\mathbf{x} \tag{3.15}$$

This is a sum of all individual variances, i.e. the total MSE distortion, over a usually finite and discrete output or feature space. The input space, however, can be any kind of continuous or discrete or both.

Although for speech or image signal compression, such difference measures may not exactly emulate human auditory or visual perceptual measures, which may need more complex integration of various discriminations including psychological aspects, the above measures are the most widely employed criteria because of their simplicity and mathematical tractability.

Minimising the total MSE distortion, however, is certainly not an easy task as it is a combinatorial optimisation problem. Generally, a direct solution does not exist. A great deal of effort has been applied to this problem; and many methods have been proposed – most notably; methods of gradient descent, least-mean-square, stochastic annealing, and neural network methods.

As we have seen from the previous chapter, the goal of the SOM is to minimise the total distortion D_1. The SOM algorithm will eventually reach a minimum (in most cases, a local minimum). We have already proved that the SOM will meet the two necessary conditions for minimising D_1, derived from signal coding and VQ theory. This also means that the SOM is inherently an information compressing algorithm and is a potential candidate for VQ and related signal compression. In Section 3.4, a detailed mathematical analysis for optimal VQ will be given, and an optimal treatment to the original SOM is proposed in order to achieve a globally optimal mapping. With regard to the ordering properties, the modified SOM can produce a noise-tolerant VQ together with a globally minimal distortion.

The criterion for information compression and signal transformation are basically the same; that is the difference between the original signal and its reconstruction should be as small as possible. *Transform coding* theory (e.g. Karhunen-Loeve transform (KLT), discrete Fourier transform (DFT), discrete cosine transform (DCT)) is one realisation of such principles. However in realising such compression optimally, the signal's characteristics may need to be taken into account. For example, sound and speech signals are recorded and represented in the form of time-series signals. However due to their production and reception mechanisms, i.e. vibrations, they may be better represented in the frequency domain. So it is not surprising why Fourier transforms and more recently wavelets (an extended, improved, and multiscale version of the Fourier transform) are so useful and popular in speech and sound signal processing. It has been known for some time that there are some parts of the auditory cortex, which respond to various frequency bands (cited by Shepherd 1988, Kohonen 1984). These auditory regions are called tonotopic map, in which different strips or fields represent different sound frequency receptive fields and are usually arranged in some ordered way.

The Fourier transformation itself can be seen as a signal compression process. The relation can be seen from Fig. 3.4.

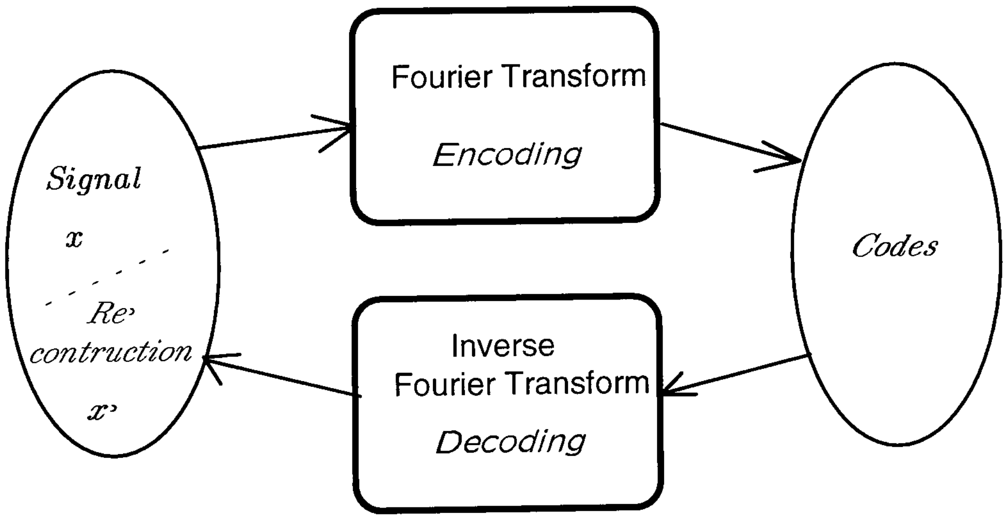


Figure 3.4: Fourier transform.

The Fourier transform maps the signal from the time domain to the frequency domain. The number of coefficients of Fourier transform can be infinite, but most coefficients are redundant as only a limited number of the most significant ones are used in practical applications. This process can seen as a near optimal encoding, and the remaining most significant coefficients can be used as the codes. Hence the reconstructed signals (via the inverse Fourier transform) have the minimum distortion when using the same number of coefficients. However if the signal spectrum is widely spread (e.g. noise corrupted signal, white noise), such compression may be a disadvantage, as the number of the dominant coefficients could be very large. VQ compression techniques in time domain have given impressive results and have become more and more popular in speech or sound coding and compression (Gersho and Cuperman 1983; Makhoul *et al.* 1985; Tsao and Gray 1985; Luttrell 1989a, b).

3.3.2 Minimum hierarchical mean squared error criterion

If the encoded or stored references can be fully recalled, or if the signal transmission channel is noiseless and every code can be received without error, or if the input signal or pattern (recalling pattern) is noiseless, then the above MSE criterion, i.e. D_1, will be the optimal criterion. However, in some cases, these assumptions are not true. For example, signal channels are often noisy, and transmitted codes (or stored codes) will be disturbed over a certain range by noise. Although many transmission techniques use high noise tolerant modulation and transmission methods, e.g. FM, PCM, Hamming code; and the *source coding* and the *channel coding* or *error control coding* can be designed separately to construct nearly optimal communication systems according to Shannon's coding theory, the principle of the noise-tolerant coding is still very important not only to signal compression but also to associative memory for fault-tolerant ability (e.g. when the input signals or recalling patterns contain noise). Some channel coding techniques pay the price for noise immunity by increasing the number of transmission bits, which reduces the compression ratio, for example, Hamming (7, 4) coding uses 7 bits to transmit a 4-bit signal, so that a single-bit channel error can be detected and corrected (Hamming 1950, cited in Gonzalez and Woods 1992). Noise-tolerant coding and compression also have some similarity to natural cognition processes and may provide some useful theories for many other artificial intelligence areas. For example even when you incorrectly recognise a word, or a object, the fact is that this word, or the object, is very similar in some sense to what you originally thought. When incorrect recall does happen, the optimal solution should limit the error to an as low as possible value.

Luttrell (1989a, b) first related hierarchical noise tolerant coding theory to the SOM principles. When considering the channel noise, a two-stage (hierarchical) optimisation has to be done not only in minimising the representing distortion D_c but also in minimising the distortion caused by the channel noise. He revealed that the SOM can be interpreted as such a coding algorithm. The neighbourhood function can be interpreted as a channel noise distribution and should not go to zero as it does in the original SOM. Kohonen (1991) has also defined a similar distortion measure for the objection function, which the SOM seeks to minimise, but concluded that the original convergence conditions are not sufficient for the ordering, by an example which indicated the difference between the results of the original SOM and his modified version. He also suggested that this modified version might be not worth applying to practical applications. This indicates that the self-organising and ordering processes, with regard to the noise tolerance optimal criterion, are still far from being clearly understood. The theories of such systems need to be exploited as they have a close relationship to the ordering theory of the SOM. We will expand this analysis in Section 3.6. Here we give the objective function for the globally optimal minimisation of the compression distortion over a noisy channel (Luttrell 1989a, b, 1994a), i.e.

$$D_2 = \sum_{\mathbf{c}\in\mathbf{Y}} \pi(\mathbf{c}) \int_{\mathbf{X}_c} \|\mathbf{w}_c - \mathbf{x}\|^2 p(\mathbf{x}) d\mathbf{x} \qquad (3.16)$$

where $\pi(\mathbf{c})$ is the channel noise density function, and index $\mathbf{c}$ is in vector form as the maps are usually arranged in an array rather than a scalar line. Eqn. 3.16 is slightly different from the one given by Luttrell in that the discrete integration over the channel noise is used here as we note that reference books (or codebooks) are always discrete.

The importance of this optimal criterion is that it can clearly explain in a well-defined mathematical form the meaning of a topologically ordered mapping, and why such a biological inspired mapping is so important and desirable in information compression and storage.

SOMs can produce a map that will satisfy the D_2 criterion, if the algorithm is properly guided and constrained during training AND the neighbourhood function is fixed (not shrinking) to the channel noise density. However the resulting reference (or feature) map will differ from the one obtained using D_1 criterion (e.g. the original SOM algorithm) in two aspects, one is the ordering fashion of the map, the other is the positions of the final references. What kind of mapping to apply and what kind of algorithm to use depend entirely on the application.

3.3.3 Minimum classification error criterion

The goal of pattern classification is to classify or group various pattern samples into different meaningful classes. This process is sometimes called *data clustering*. Each class or cluster usually has certain internal structures in the distribution of its members. Such structures can be pre-determined by some valid assumption or learnt from known samples; or have to be estimated from unknown class data by on-line learning and testing. In the former case, the learning is carried out in a supervised manner; while in later case, the learning is carried out in an unsupervised manner.

An example is shown in Fig. 3.5. The requirement is to group data into some kind of meaningful classes, where some kind of underlying pattern structures can be seen. Each class then can be represented by its cluster centre. The boundaries between classes are defined in some way to perform optimal clustering, i.e. produce least possible errors.

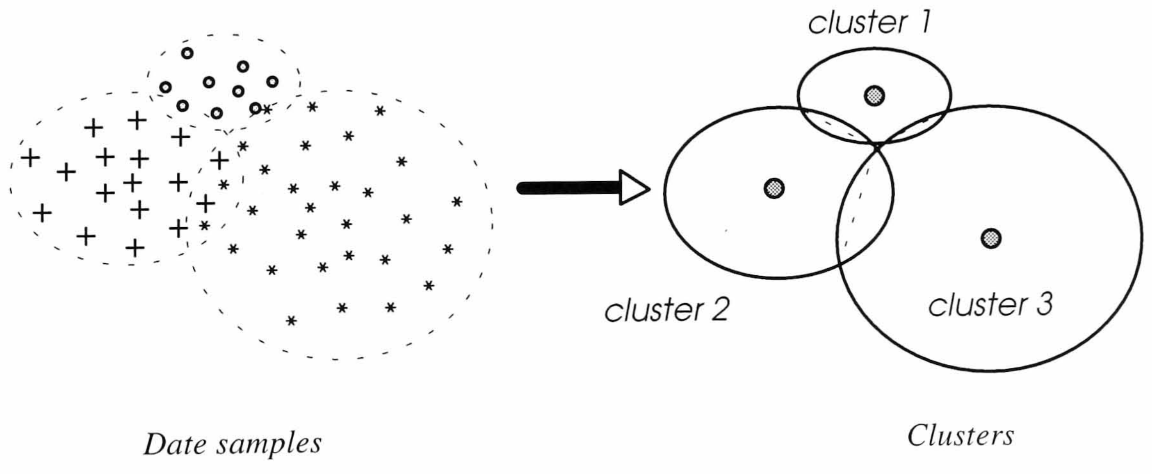


Figure 3.5: Data clustering.

The criterion for verifying the quality of classification can only be the *minimum classification errors*. Optimal clustering results in the minimum mis-classification. In Bayesian terms, this can be expressed as

$$D_3 = \sum_{i=1}^{K} P(\omega_i) \int_{\mathbf{x} \notin \Omega_i} \mathbf{x} p_i(\mathbf{x} / \omega_i) d\mathbf{x} \tag{3.17}$$

where K is the total (correct) number of the pattern classes, Ω_i is the data set for the i-th class, $P(\omega_i)$ represents the probability of the i-th class, and $p_i(\mathbf{x}/\omega_i)$ is the density function for class i.

Sometimes when the class probabilities and density function are not available (as in many discrete cases where only a finite training data are available) the above criterion for error measurement has to be simplified to

$$D_3' = \sum_{i}^{K} \sum_{\mathbf{x} \in \Omega_i} \{1 - \delta[m(\mathbf{x}) - i]\} \tag{3.18}$$

where the membership function $m(\mathbf{x})$ represents the true class member for $\mathbf{x}$, and the delta function δ equal to 1 when its variable is 0, and 0 otherwise. In this case, each class's density is its histogram's discrete distribution from its samples, and each sample is assumed to have equal probability (which might not be true in some practical problems).

$\{\Omega_i, P(\omega_i), p_i(\mathbf{x}/\omega_i), i=1,2, \quad K\}$ are the parameters to be estimated. In supervised learning, class probabilities and density functions are known, or can be learnt from known training samples. The only task is to assign the input test samples so that Eqn. (3.17) or (3.18) reaches the minimum. When some, or all, of the above parameters are unknown, the learning becomes unsupervised. Usually the class densities have to be assumed (with possible later adjustment) to be a particular function such as Gaussian. Other parameters can then be learnt on-line from samples. This is usually more difficult than in supervised cases. Sometimes even the class number K is unknown, so a validation process is needed to find out the best class number. Normally, an initial K has to be assumed and when the clustering converges, select a different K, and apply clustering algorithm again. Repeating doing this and comparing the errors of different K's, the one giving the minimum errors is the best answer.

According to Bayesian theory, the minimum classification error is obtained by assigning or labelling each input $\mathbf{x}$ according to its maximum *a posteriori* (MAP) probability, i.e.

$$l(\mathbf{x}) = \arg\max_{i} \{P(i / \mathbf{x}) \equiv \frac{P(i) p(\mathbf{x} / i)}{p(\mathbf{x})}\} \tag{3.19}$$

It has been proved that multilayer perceptron neural classifiers estimate these Bayesian *a posteriori* probabilities when the network has one output for each pattern class; and desired outputs are *1* of M (one output unity corresponding to the correct class, all other zero) (Richard and Lippmann 1991). The estimation accuracy depends on network complexity, the amount of training data, and the fitness of training data to likelihood distributions or class densities, $\{p(\mathbf{x}/i)\}$, and prior class probabilities, $\{P(i)\}$. However, for self-organising networks, when used as classifiers, there is no formal analysis to show whether they will converge to Bayesian classifiers or what kind of performance they can achieve. Many experimental results have suggested that SOMs will not converge to Bayesian performance unless classes are well separated. In Section 3.5, we will explore this by comparing the asymptotical properties obtained in the previous chapter with the properties of Bayesian classifiers. An optimal treatment of the original SOM, which can lead to forming the Bayesian classification, is proposed.

In much of the literature, clustering algorithms and VQ algorithms are often confused. Only when in very specific circumstances (e.g. uniform distribution, well separated classes), will one algorithm produce optimal performance for both VQ and classification. Because the objective functions for VQs and pattern classification are different, one algorithm which is optimal for VQ (or classifier) will not be optimal for the other. As the SOM and CL algorithms are potentially optimal for VQ, they are normally not optimal for pattern classification. However, they are often applied for such tasks. The SOM and CL algorithms can produce an optimal classification performance only if they are appropriately modified. In Section 3.5, an extended SOM is proposed to yield Bayesian classification performance.

3.4 Towards the (Global) Optimal VQ

3.4.1 Two necessary conditions for optimal VQs

The VQs we will discuss in this section are noise-free but lossy VQs, or (lossy) source coding methods. Discussions on noise tolerant VQs will be given in Section 3.6.

The two well-known necessary conditions for an optimal VQ that minimise D_1 are (**i**) the nearest neighbour condition, or Voronoi partition; and (**ii**) the centroid condition (Linde et al. 1980, Gersho 1979). They can be directly derived from the differential of the distortion D_1. They can be stated as:

1. ***Nearest neighbour condition:*** Given reference vectors $\{\mathbf{w}_c\}$, the VQ is a minimum distortion mapping, i.e. it assigns each input vector according to the nearest neighbour rule:

$$Q(\mathbf{x}) = \mathbf{w}_i \;\text{ or }\; \mathbf{x} \in \mathbf{X}_i \;\text{ iff}\, d(\mathbf{x},\mathbf{w}_i) \le d(\mathbf{x},\mathbf{w}_c) \;\; \forall\, c \in \mathbf{Y} \tag{3.20}$$

   where the distance measure is normally Euclidean.
   This is also called the Voronoi partition, and it minimises the average distortion for given reference vectors.

2. ***Centroid condition:*** Given a partition $\{\mathbf{X}_c\}$, the reference vectors should be the centroids of the partitions, i.e.

$$\mathbf{w}_c = cent(\mathbf{X}_c) = E[\mathbf{X}_c] = \int_{\mathbf{x}\in\mathbf{X}_c} d(\mathbf{x},\mathbf{w}_c)p(\mathbf{x}/\mathbf{X}_c)d\mathbf{x} \tag{3.21}$$

   Each reference vector locally minimises the distortion in its corresponding partition.

There is another necessary requirement for optimal VQs, i.e. the ***zero probability boundary condition*** (Gersho and Gray 1992), which is identical to Eqn. (2.22). It requires that partitions are exclusive and cover all input space, which is clearly the case in many algorithms.

The above are the necessary conditions for an optimal VQ. The sufficient conditions can only be that it minimises the average distortion, e.g. D_1, among all possible mappings for input samples. Since local minima, which also satisfy the two conditions, are often the results for practical applications, the requirement for a globally optimal VQ becomes increasing important. There are no direct solutions to this problem. An asymptotical property of optimal VQ was given by Gersho (1979) which shows that for an optimal VQ, each Voronoi partition has the same contribution to the total distortion, under the conditions that the input distribution is smooth and the number of reference vectors is very large. This property will be applied in Section 3.4.3 to the SOM algorithm as a constraint to guide the SOM to converge to the (global) optimum.

3.4.2 A comparative study of various VQ algorithms

There are many signal compression methods for both lossless and lossy compression, such as lossless bit-plane encoding (Gray code, Run length encoding, etc.), lossless predictive coding (DPCM, Huffman, etc.); lossy predictive coding (e.g. DPCM, ADPCM), transform coding (e.g. K-LT, DFT, DCT, JPEG DCT), block truncation coding, subband coding, and VQs (e.g. LBG, Residual VQ, Classified VQ, Finite-state VQ, Predictive VQ). For a comprehensive review of theoretical and technical aspects of this topic see Gersho and Gray (1992), and Rabbani and Jones (1991). In this section we will limit our attention to the learning methods in lossy VQ and related compression techniques.

Before we discuss any technical details of the various methods, we shall first mention some commonly used quantitative measures for VQ performance.

Compression Rate:

General definition: $$R = \frac{\textit{Number of bits for original image}}{\textit{Number of bts for compressed image}} \tag{3.22}$$

Commonly used in VQs: $$r = \frac{\log_2 K}{k} \tag{3.23}$$

where K is the codebook size, i.e. the number of reference vectors; and k is the block size, i.e. the vector dimension, r means how many bits are needed for per pixel, i.e. bit/pixel or bpp.

The distortion measure is the distortion form D_1, i.e. Eqn. (3.15). In the case of discrete input space, its second integration has to be changed to a summation form, and it is also termed the MSE:

$$D_1' = MSE = \frac{1}{M} \sum_{c \in \mathbf{Y}} \sum_{\mathbf{x} \in \omega_c} (\mathbf{w}_c - \mathbf{x})^2 \tag{3.24}$$

where M is the total number of data points or pixels, i.e. the size of the original image.

Sometimes, this is expressed as the peak signal-to-noise ratio (PSNR) and SNR, and is used as a measure of quality in image and other source coding, respectively. They are defined by

$$PSNR = 10 \log \frac{255^2}{MSE}, \quad \text{and} \quad SNR = 10 \log \frac{P_s}{MSE} \tag{3.25}$$

where P_s is the signal power.

(1) *Linde-Buzo-Gray (LGB) Algorithm* (Linde, *et al.* 1980)

This is the most widely used algorithm for generating codebooks. The LBG algorithm is a generalisation of the Lloyd-Max iteration algorithm (GLA) (Lloyd, 1957). This algorithm is a direct implementation of the two necessary conditions for optimal VQs, and is very similar to the *k*-means algorithm developed by MacQueen (1967) for cluster analysis and pattern classification. The LBG algorithm is a batch version of the least squares method for minimising total MSE distortion. The algorithm involves the following steps:

(0) Initialisation: Given the number of code vectors, K, and a distortion threshold $\varepsilon \geq 0$. The initial codebook $\{\mathbf{w}_c(0)\}$ is chosen randomly, or by using first K input vectors. The average distortion is set $D_1(0)=\infty$.

(1) Given $\{\mathbf{w}_c(0)\}$, find the minimum distortion partition $\Phi\{\mathbf{x}\in \mathbf{X}\}=\{\mathbf{X}_c(n)\}$, from (3.20), i.e. $\mathbf{x}\in \mathbf{X}_i$, if $d(\mathbf{x}, \mathbf{w}_i(n))<d(\mathbf{x}, \mathbf{w}_c(n))$, $\forall c\in \mathbf{Y}$, but $c\neq i$. The partition goes over all input data. Compute the resulting average distortion $D_1(n)$.

(2) If $(D_1(n)-D_1(n-1))/D_1(n) \leq \varepsilon$, then halt with codebook: $\{\mathbf{w}_c(n)\}$ and partition: $\{\mathbf{X}_c(n)\}$ describing the final quantisation. Otherwise continue.

(3) With the partition: $\{\mathbf{X}_c(n)\}$, find the new code vectors: $\{\mathbf{w}_c(n+1)\}$ by (3.21), i.e. $\mathbf{w}_c(n+1)=cent(\mathbf{X}_c(n))$, $\forall c\in \mathbf{Y}$. Then go to (1).

In the above, we did not specify how the code vectors are arranged. Normally the code vectors are arranged in a numerical order, i.e. c=1, 2, ...K, and there is no relation between neighbouring references. In some cases, however, we would like the neighbouring references to be close to each other in the input space (i.e. an ordered codebook), so that it can demonstrate tolerance to signal and transmission noise.

For the above algorithm, in the training and/or encoding, each step is performed over all reference vectors and data samples. If the codebook size is not small, searching for the minimum distorted representative can be a time consuming task. There are methods for fast reference searching, e.g. using *tree-structured codebooks*. A very popular structure is the binary codebook tree.

In practical codebook design (after training), another useful strategy is to use *product codebooks*. The reason is that when the bit rate is fixed, the codebook size grows exponentially with the block size. For example, for an 8-bit 512×512 image at 1.0 bpp rate, and 4×4 block size, the codebook size will be 64K! Therefore if we can use separate codebooks, N_{c1}, N_{c2} for example, then the effective size of a product codebook will be $N_c=N_{c1}*N_{c2}$ instead of $N_{c1}+N_{c2}$. There are several types of product codebooks, namely, mean/residual VQ (M/RVQ), interpolative/residual VQ (I/RVQ), and gain/shape VQ (G/SVQ).

Here, these techniques are not our major concern. We focus on the learning mechanisms that describe how to learn from inputs efficiently, effectively and optimally so that the code references converge and become a good representation of the input space.

(2) *Competitive Learning (CL) VQ Algorithm*

Competitive learning methods have been very popular for unsupervised learning (Grossberg 1976a, b, 1987; von der Malsburg 1973; Amari and Takeuchi 1978; Kohonen 1982; Bienenstock *et al.* 1982; Rumelhart and Zipser 1985), and for adaptive vector quantisations (Ahalt *et al.* 1990; Kosko 1991; Yair *et al.* 1992).

The CL-VQ is an adaptive version of the least squares method for minimising total MSE distortion. For each iteration, only one input vector is presented and the closed code vector, i.e. the winner, adapts towards the input.

(0) Initialisation: Given the number of code vectors, K, and a distortion threshold $\varepsilon \geq 0$. The initial codebook $\{\mathbf{w}_c(0)\}$ is chosen randomly, or from (the first) K input vectors. Set the average distortion $D_1(0)=\infty$. Set initial adaptive learning rate $\alpha(0)$, $0<\alpha(0)<1$.

(1) Randomly present an input vector from the input space, find the winning reference by

$$v = \arg\min_{c \in \mathbf{Y}} d(\mathbf{w}_c(n) - \mathbf{x}(n)) \tag{3.26}$$

(2) Update the winning reference:

$$\mathbf{w}_v(n+1) = \mathbf{w}_v(n) + \alpha(n)[\mathbf{x}(n) - \mathbf{w}_v(n)] \tag{3.27}$$

then decrease the learning rate $\alpha(n)$ to $\alpha(n+1)$ monotonically, usually on the scalar inverse to the time n.

(3) Theoretically it needs $n \to \infty$ for the algorithm to converge to the (local or global) minima. In practice, $(D_1(n)-D_1(n-1))/D_1(n) \leq \varepsilon$ can be used to indicate when to stop the algorithm. Otherwise go back to (1).

(3) *Self-Organising Map (SOM) VQ Algorithm*:

There has been considerable research in applying the SOM algorithm to image or speech VQs (e.g. Carrato 1994; Chen *et al.* 1994; Kim and Ra 1995). As has been described in detail in Chapter 2, the algorithm is similar to the CL algorithm in that it has the same steps of (0), (1), and (3) of the CL algorithm, but the difference is that in step (2), the updating not only applies to the winner, but also to its neighbouring references. The notation for the neuron index is also different for image VQs: c should become a vector $\mathbf{c}$ in the SOM algorithm as the codebook is often arranged in 2-D space.

To see the differences between these three representing algorithms, let us first consider a simple example: using four code vectors to quantise the input data in a 2-D space as shown in Figure 3.6, which is similar to the one discussed by Ahalt *et al.* (1990).

There is no doubt that all three algorithms can find the best quantisation result, i.e. each reference at the centroid of each data subset, provided the initial references are carefully selected. However among these algorithms, the LBG and CL algorithms are very sensitive to the initial states. Normally their initial states cannot be randomly selected, otherwise some neurons will not have an opportunity to update. This is the so-called "*under-utilisation*" problem. In this example, if all initial references are randomly set in a small range around the origin (this kind of initialisation is very popular in practice when the distributions of the training set are not known or when the training is carried out in real time), the LBG and the CL algorithm will often encounter such problems. Some neurons will stay in their original positions forever. Sometimes there is another problem, termed "*over-utilisation*" problem, i.e. some neurons will over-react and become responsive to too much of the data. This will result in one neuron converging to the middle of two subsets. The SOM algorithm is better in overcoming these two problems. As we have already seen in Chapter 2, the SOM algorithm does not depend on the initial states (*the order of the map may depend on the initial states, if the neighbourhood function is not implemented correctly*).

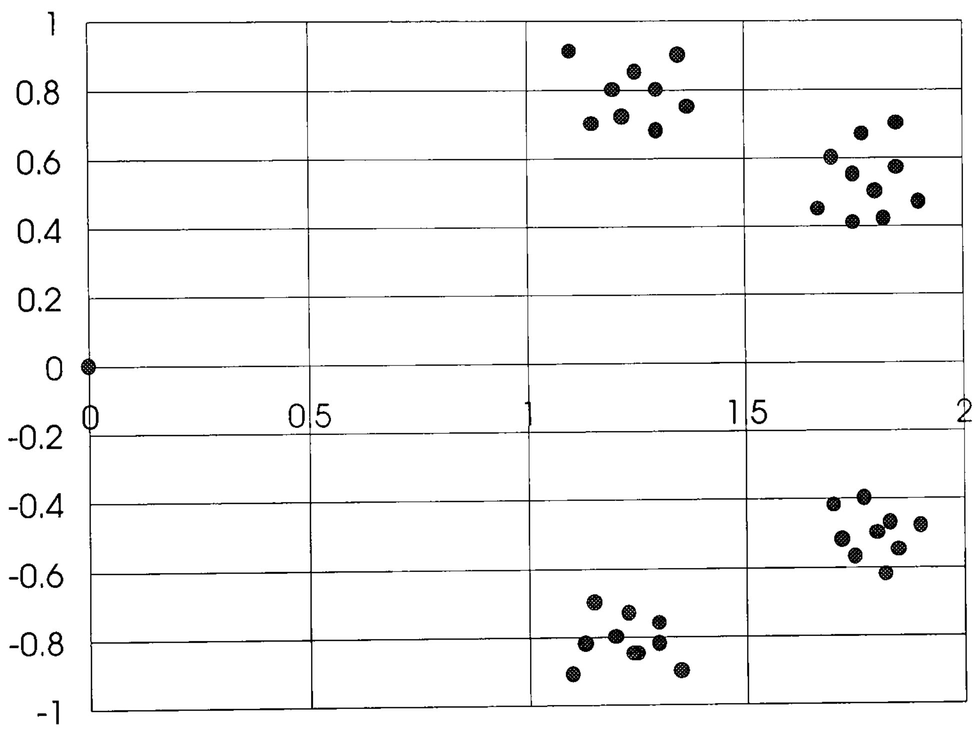
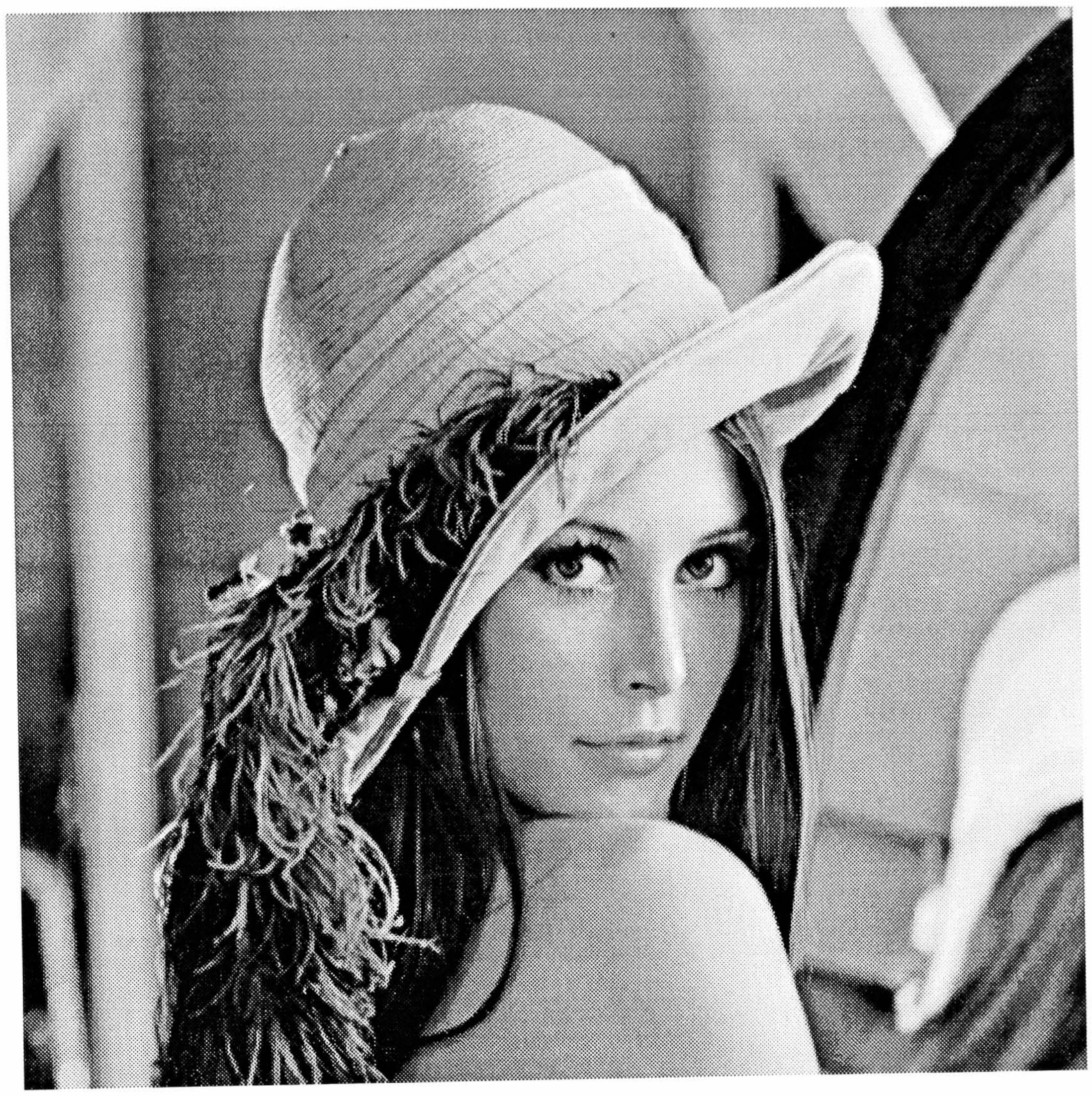


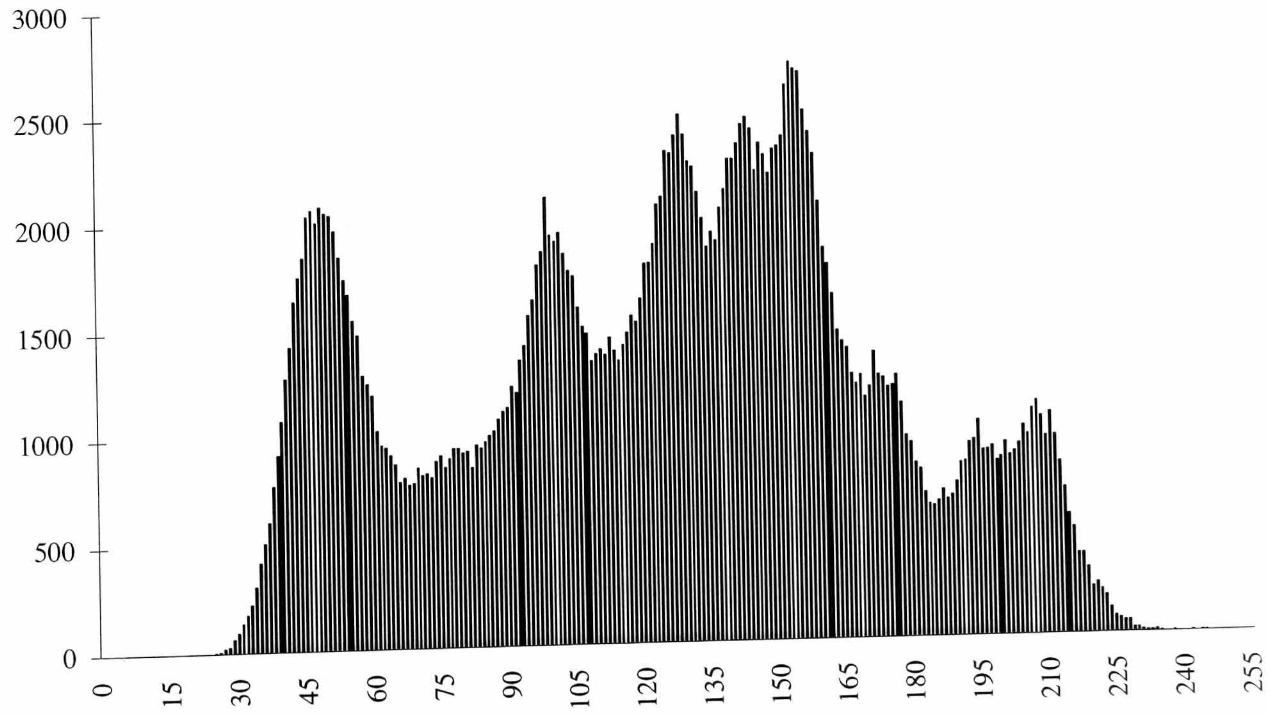
Figure 3.6: An input space to be quantised.

Next, a more practical, complex, and popular example will be considered — compression of the image "Lena". The original image is in 8-bit and 512x512 pixels shown in Fig. 3.7. The codebook size chosen is 256. In the LBG and CL algorithms, the codebooks are not ordered, so can be arranged in any numerical order, e.g. 1-D array; while in the SOM algorithm, we arrange the codebook in to a 2-D 16x16 array. The block size chosen is 4x4. So the bit rate of the quantisation is 0.5 bpp (from Eqn. (3.23)). The training samples are from the entire image, thus there are totally 128x 128=16,384 samples for 4x4 block size.

For these three algorithms, the LBG and CL algorithms cannot use random values as their initial references, and have to use some of data samples, e.g. 256 raster scan samples starting from any point, e.g. (0, 0), or (100, 0), or (200, 0), of the image. While the SOM can use randomly selected values as its initial references. The final results of the SOM algorithm show very little dependence on the initial states. Typical results after 30 iterations for these three algorithms are shown in Figs. 3.8 and 3.9, in which all algorithms used (0, 200) and onwards pixels as their initial references (we found that they were the best for the LBG algorithm). Since it is meaningless to show the codebooks of the LBG and the CL algorithms, only their quantisation results (i.e.

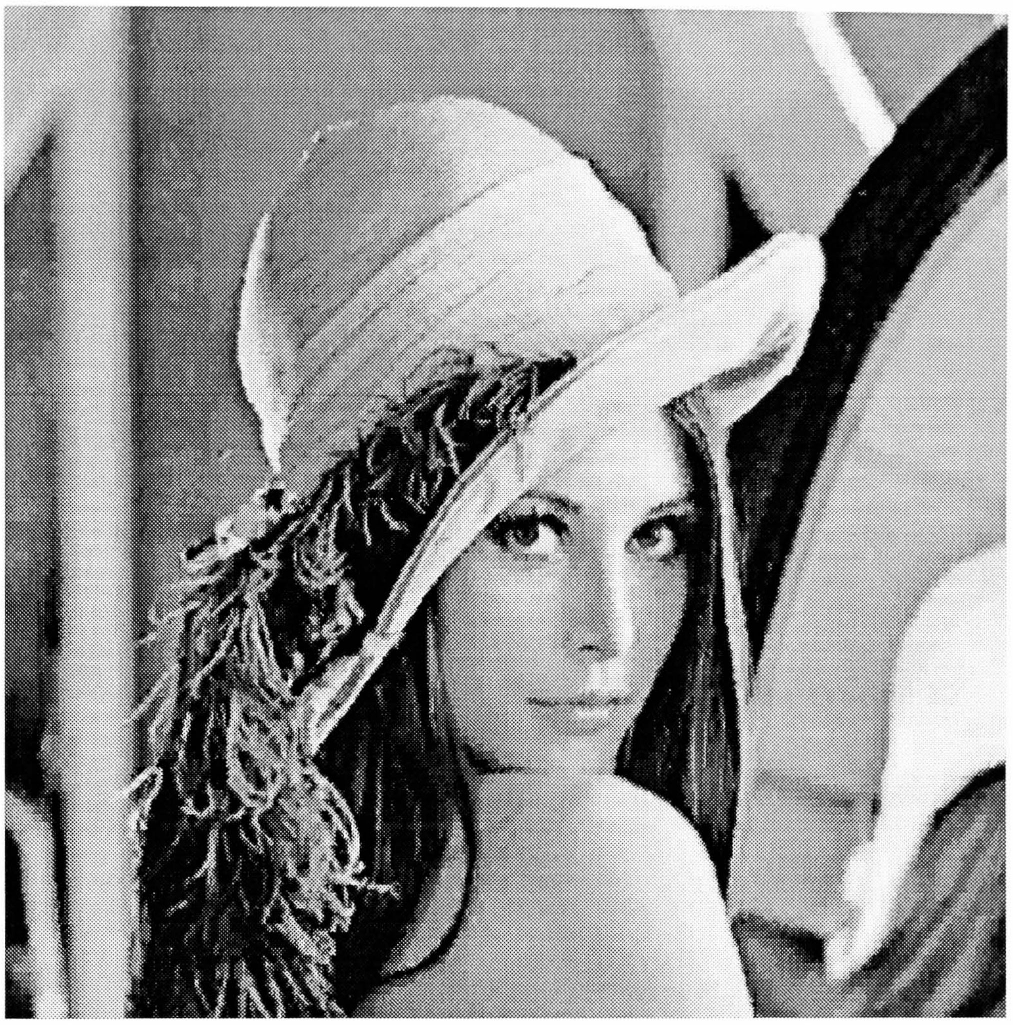


(a)

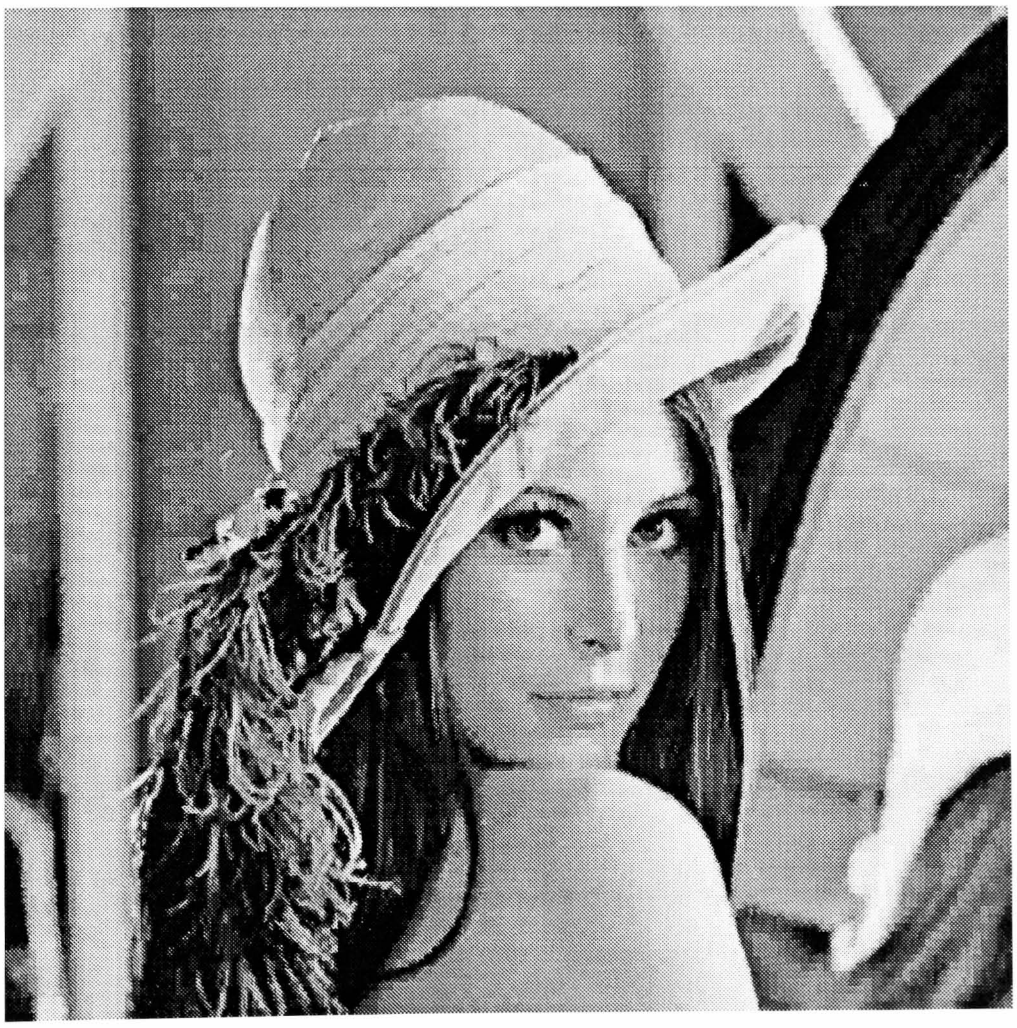


(b)

Figure 3.7: (a) Original Lena image (8-bit, 512×512 pixels); (b) Its histograms.

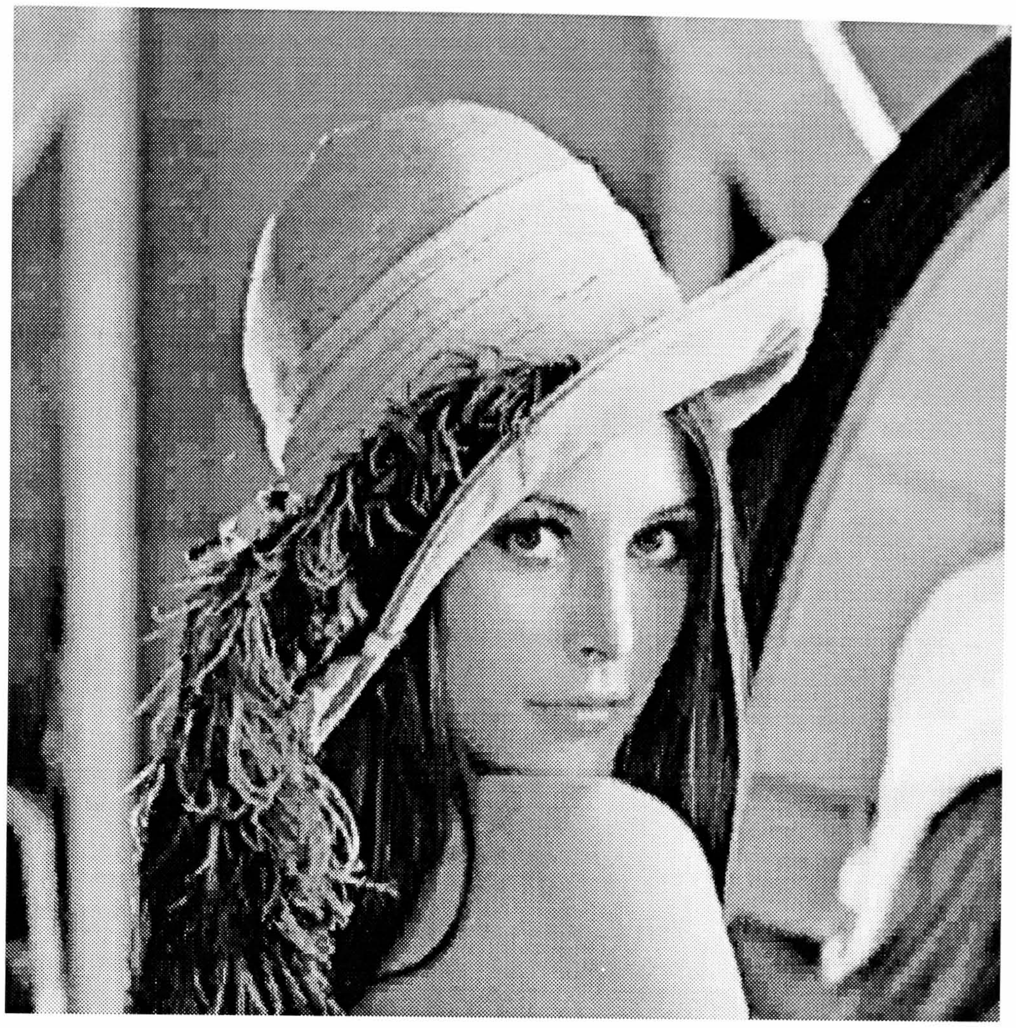


(a)

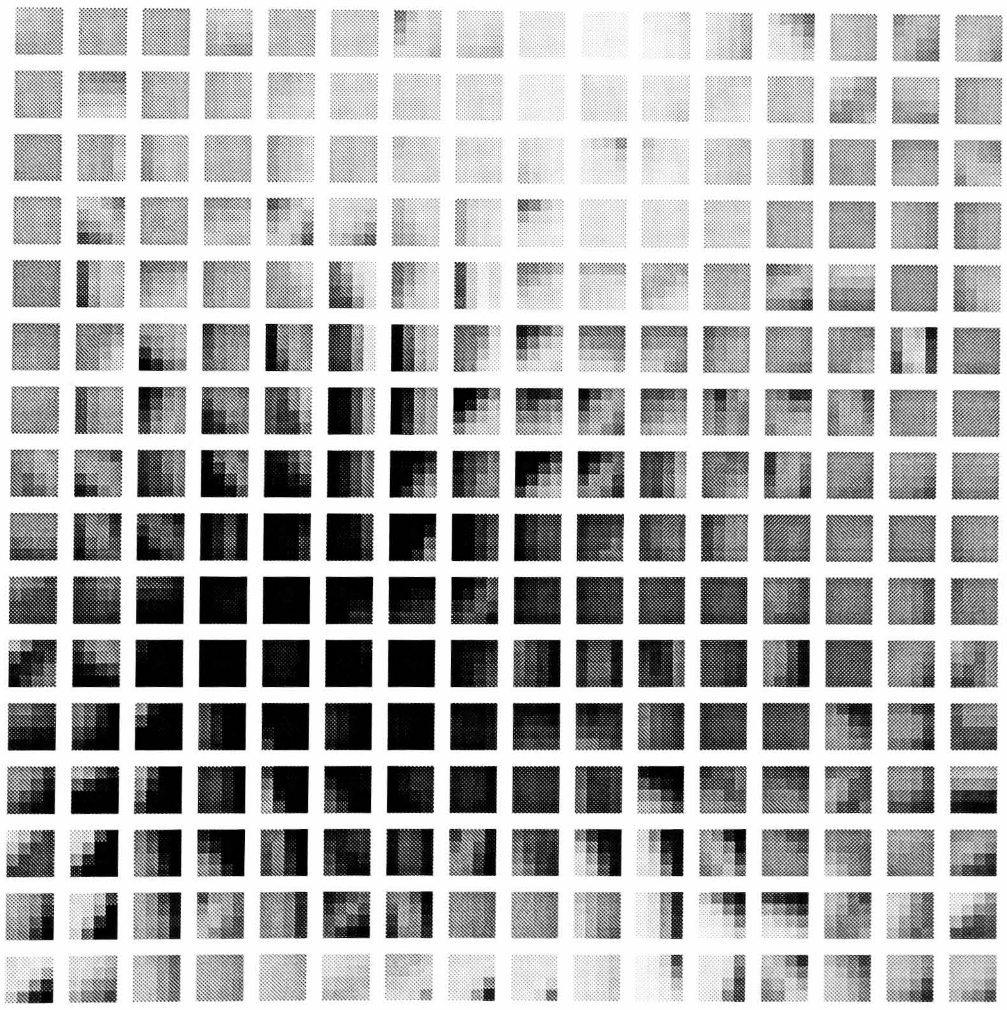


(b)

Figure 3.8: (a) LBG-VQ; (b) CL-VQ, at 0.5 bpp, 30 iterations.



(a)



(b)

Figure 3.9: (a) SOM-VQ; (b) Its codebook, at 0.5 bpp, 30 iterations.

reconstruction images) are shown, which is obtained after 30 *entire iterations* (one entire iteration requires supplying the entire data set, i.e. 16,384 samples). For the LBG, the samples can be input in a natural raster scan order, while for the CL and SOM, the samples are normally input in a random order, but each sample must be used only once in an entire iteration. In Fig. 3.9 the results from the SOM algorithm and its codebook are shown. The codebook is in a 2-D array and in a full wrap fashion, i.e. left side and right side are connected, and top side and bottom side are connected. As we can see the local ordering of references exists in many places. Because it is a very high dimensional reduction case, global order may not exist. Further analysis of ordering will be given in Section 3.6. The best local ordered map should have as many local ordered areas as possible.

These results are visually very similar. The slight differences between algorithms can only be seen in their performance curves. Here we use the widely adopted PSNR values instead of MSE distortions, which are shown in Fig. 3.10. Since the SOM needs to adjust both winners and their neighbouring neurons, the distortions are higher than those of the LBG and CL algorithms in the early stages. Updating over a neighbourhood can not only provide an ordered codebook but also escape some local minima in the MSE distortion. However it will slow down the convergence. Therefore the SOM will normally take a longer time to converge than the LBG and CL algorithm. The final PSNR results (after 30 iterations) are 31.72 dB, 31.92 dB, and 31.90 dB for the LBG-, CL-, and SOM- VQs respectively.

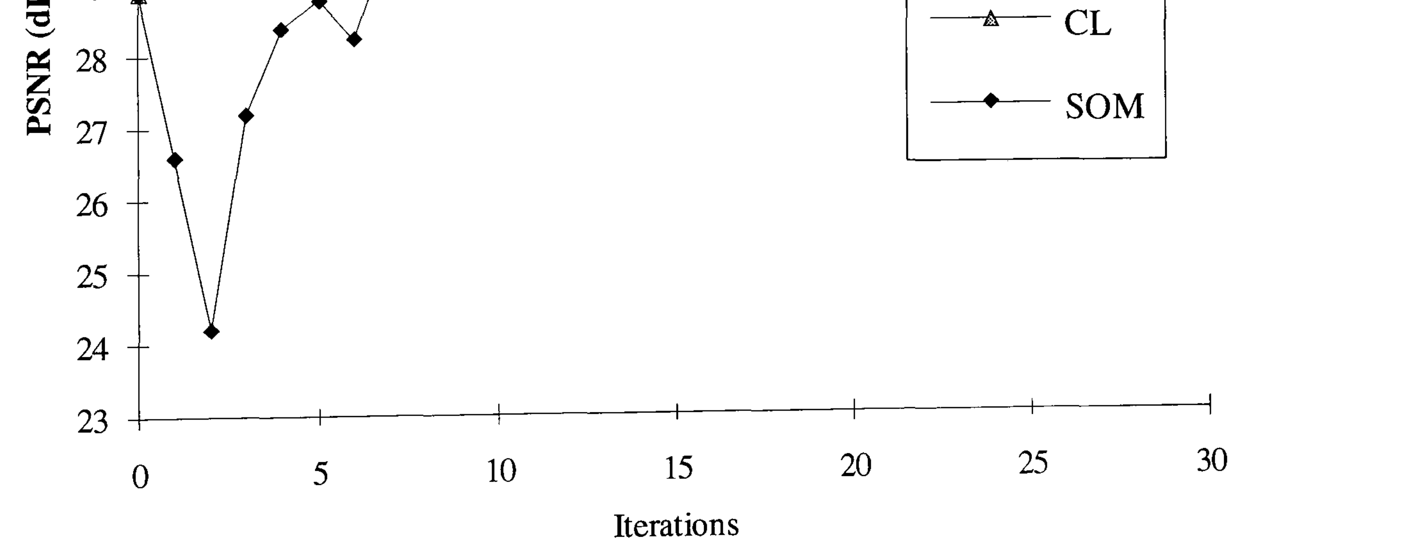


Figure 3.10: PSNR performance comparison for the LBG-, CL- and SOM- VQs.

The three methods can produce very similar results. The CL- and SOM-VQs yield just a slight better result than the LBG-VQ. By letting neighbourhoods shrink slower, learning rates decrease slower, and iterations go on even longer, the SOM algorithm will give the best PSNR results, even though there is not much difference. The LBG algorithm is a gradient descent method for reducing the distortion, at each iteration the total distortion always goes down (PSNR goes up); while the CL and SOM are stochastic gradient descent methods, which do not necessarily reduce the total MSE distortion after each iteration because of their local minimising properties, but will reduce statistically the mean of the total distortion at every iteration. This property can give the CL and SOM algorithms greater possibility in escaping local minima. The SOM algorithm results in even more dynamic processes in reducing the total MSE distortion, because updating within a neighbourhood cannot even minimise local distortion at each iteration. However, since the neighbourhood will shrink to zero, the SOM will tend to the CL algorithm and will minimise every local distortion. The neighbourhood updating may find an even better local minimum and will result in an ordered reference book. In summary, for the LBG and CL algorithms, the initial references have to be chosen very carefully, otherwise they will suffer from "under-utilisation" and/or "over-utilisation" problems. However for the SOM algorithm, if the neighbourhood function is properly set (initially wide neighbourhood radius, and gradually decreasing with time), and the initial references are different, the under-utilisation problem normally will not appear. However the SOM algorithm is slower (or has to be slower, in order to produce ordered maps and escape from local minima) than the LBG and CL algorithms if not implemented in parallel.

There are also some constrained versions of the CL algorithm, like *the frequency-sensitive competitive learning* (Ahalt *et al.* 1990), that apply "conscience" learning to the algorithm, so that less-fired neurons are encouraged while over-fired neurons are restricted in order to avoid under-utilisation problems. This is intended to lead to an approximately equal firing frequency of neurons, or *equal probability mapping*. The idea comes from Grossberg's work (1976a, b, 1987) and others (DeSieno 1988; Hecht-Nielsen 1988; etc.).

The neighbourhood function of the SOM can naturally provide a solution to the under-utilisation problem. Since the neighbourhood size or radius is very large at the beginning, this will cause all references to respond to the input samples. This is a natural "conscience" learning. A large neighbourhood results in a large cluster of references; and similar references, in the input space distance sense, will be close together in output space. While a small neighbourhood results in a spreading of references. When the neighbourhood size is shrinking correctly (slowly enough), the input space will be well covered and quantised by all references. Such a role also gives the SOM algorithm an ability for escaping local minima, as updating in the neighbourhood can disturb the minimum state. Although it can not be guaranteed that the SOM will find the global minimum, the algorithm indeed can provide a better local minimum compared to other algorithms. The problem is that the emergence, frequency, and quality of such local minima are unpredictable, i.e. the reliability of the algorithm is not very high. The results heavily depend on the algorithm parameters and training procedures. In next subsection, an improved version of the algorithm is proposed, in order to make the algorithm more robust at finding the global or a "better" local minimum.

Another great advantage of the SOM algorithm is that its codebook is ordered, which can inherently increase the noise tolerant ability of encoding. Therefore, when the data to be quantised and/or the transmission channel are noisy, the SOM-VQ can provide better results.

3.4.3 A constrained SOM towards the global optimal VQ

As we have described before, VQ algorithms are optimisation methods which use a very limited number of references to approximate the input space. A general principle and criterion for these methods is to limit the approximation distortion as low as possible. Most existing algorithms can not guarantee to reach the globally minimum distortion. The local minimum problem has become a very challenging issue in VQ design. Some algorithms, which combine the LBG or CL learning with some combinatorial optimisation methods from statistical physics (e.g. stochastic relaxation) have been proposed for escaping local minima (e.g. Zeger *et al.* 1992). Theoretically such relaxation schemes, also called simulated annealing, can converge *in distribution* to the global minimal energy configuration, if a slow enough annealing schedule is followed (Geman and Geman 1984). However this schedule is too slow to use in practice. Instead most relaxation methods actually use a fixed temperature or a fast annealing schedule, e.g. "sub-optimal" exponential schedule (Kirkpatrick *et al.* 1983). This has been used by Zeger *et al.* (1992). Though they will not lead to the global optimum, they can find a much better local optimum.

More recently, some more active searching strategies have been proposed and added to VQ algorithms for escaping local minima, as relaxation methods are passive and "blind" in avoiding local minima, and more importantly they are too slow to use in real applications. Ueda and Nakano (1994) have proposed a so-called *competitive and selective learning* VQ algorithm based on competitive learning, genetic selection mechanism, and the equal-distortion or *equidistortion* principle (Gersho 1979) (to be explained below). Another interesting method, called the optimal adaptive *k*-means algorithm, which also applies the equidistortion principle, has been proposed by Chinrungrueng and Sequin (1995). This algorithm adaptively weights the distance measure and adjusts the learning rate of competitive learning VQs, according to the variances of the subsets. Both methods apply the asymptotical equidistortion property of the optimal VQ, i.e. *for a smooth underlying probability density and large number of the code vectors, all regions in an optimal Voronoi partition have the same within-region variances* (Gersho 1979).

The above two new approaches are CL based, whose codebooks are randomly arranged. Since SOM algorithm updates not only the winner but also its neighbouring references, this may result in slowing down the convergence, but will result in a better quantisation. To ensure that the SOM algorithm converge to the global or a near-global minimum, its learning procedure has to be monitored and properly guided. In the following, an equidistortion constrained SOM (ECSOM) is proposed. The ECSOM algorithm differs from the above two methods in that the asymptotical equidistortion principle is indirectly applied. The above two methods require two asymptotical conditions, i.e. a very large number of code vectors and smooth input density, so that resulting partitions have exactly the same variance. This may not be true in many practical applications. Especially, in Chinrungrueng and Sequin's methods, the equidistortion principle is applied through variance weighted distances. So if the final variances of each partition are not the same, the first of two necessary optimal VQ conditions will be violated. However, in the ECSOM the constraints are indirect and soft, because they are applied through controlling the radii of the neighbourhood function. The two necessary optimal conditions for VQ will not be violated. Updating in a neighbourhood will result in the neighbouring references being closer to the winner than just updating the winner, therefore the winner will have lower variance. Using this function of the neighbourhood, the estimated variance of each reference in its partition is employed to control the width of the neighbourhood function of that neuron. The variances can be easily estimated through its distance measure calculations in the original SOM algorithm at no extra computational costs. Details of the ECSOM are given below.

(1) ***Equidistortion Constrained SOM (ECSOM) VQ Algorithm:***

(0) Initialisation: The initial codebook $\{\mathbf{w}_c(0)\}$ is chosen randomly, or by using (first) K input vectors. Set the initial variances, $\{\mathrm{var}_c(0)\}$ of partitions to the same value, say, 1.0, and initial firing frequencies $\{P_c\}$ to the same value. Set initial adaptive learning rate $\alpha(0)$, initial common neighbourhood size $\aleph(0)$, and their shrinking rates, as in the original SOM. Set the starting time, T_0, for applying constrains.

(1) Randomly present an input vector from the input space. Find the winning reference by

$$\mathbf{v} = \arg\min_{\mathbf{c}\in\mathbf{Y}} d(\mathbf{w}_c(n) - \mathbf{x}(n)) \tag{3.28}$$

(2) Calculate the learning rate $\alpha(n)$ as usual. Calculate the neighbourhood size or radius, $\aleph_v(n)$ as usual if $n<T_0$, otherwise, calculate the $\aleph_v(n)$ and multiply a constraint factor:

$$F_v = \frac{\sqrt{\mathrm{var}_v(n)P_v}}{average\{\sqrt{\mathrm{var}_v(n)P_v}\}} \tag{3.29}$$

(3) Update the winning reference and its neighbourhood:

$$\mathbf{w}_c(n+1) = \mathbf{w}_c(n) + \alpha(n)[\mathbf{x}(n) - \mathbf{w}_c(n)], \qquad if\, \mathbf{c} \in \aleph_v(n)F_c(n) \tag{3.30}$$

Update the variance estimate of the winner in its partition:

$$\mathrm{var}_v(n+1) = \mathrm{var}_v(n) + \alpha(n)\{\min d[\mathbf{w}_c(n) - \mathbf{x}(n)] - \mathrm{var}_v(n)\} \tag{3.31}$$

(In step (1), the distances, $\{d[\mathbf{w}_c(n)-\mathbf{x}(n)]\}$ have already been calculated, so there will be no increase in computational costs in adding this variance estimation step).
Update the firing probability of the winner by: $P_v=P_v+1$.

(4) Theoretically it needs $n\to\infty$ for the algorithm to converge to the (local or global) minima; however, in practice the total variance change can be used to indicate when to halt the algorithm. Otherwise go back to step (1).

As we can see, in the ECSOM algorithm, when a winning neuron has a large variance, its neighbourhood size is also large so more of its neighbouring neurons will be updated towards to it. This will make its variance smaller. This kind of constraint is not strict, so there is no need to require the final partitions to have the same variance. However this constraint will bring the variances of the partitions as close as possible. This is the general principle in more practical applications, where the input probability density of the input space is not smooth and/or the number of code references is not very large.

This constraint principle can be also applied through the learning rate instead of the neighbourhood or both, e.g. large variance partitions have large learning gains. Here we will only demonstrate the constraint to the neighbourhood, i.e. the above algorithm. Two typical examples are given below.

(2) ***Example 1***: *A Lloyd's example: Using two scalar references, {w_1, w_2}, to quantise the input space [0, 1] with the probability density function (pdf) as shown in Fig. 3.11, in which p_1 and p_2 represents the probability density for the first half and second half of input space respectively, where $p_1 \geq 0$, $p_2 \geq 0$, and $p_1 + p_2 = 2$.*

The strict mathematical calculation, which is possible in this case, has shown that when the ratio $p_1/p_2 \leq 3$, there is only one MSE distortion (i.e. D_1) minimum at $\{w_1, w_2\}=\{0.25, 0.75\}$. However, when the ratio $p_1/p_2 > 3$, there are two minima: the local minimum is always at $\{w_1, w_2\}=\{0.25, 0.75\}$, while the global minimum depends on the ratio. For example, when $p_1/p_2=4$, the global minimum is at $\{w_1^*, w_2^*\}=\{0.21875, 0.65625\}$.

In this example when there are two minima, i.e. $p_1/p_2>3$, most VQ algorithms, e.g. LBG, CL, etc. will usually converge to the local one if the initial references are not carefully selected. The SOM algorithm does not depend on the initial states and is better in finding the global or a good local optimum. However the SOM cannot ensure convergence to the global minimum. Sometimes it goes to the global one, while other times it goes to a local one. When constraints are applied, the ECSOM algorithm will indeed converge to the global optimum. This has been confirmed by extensive simulations.

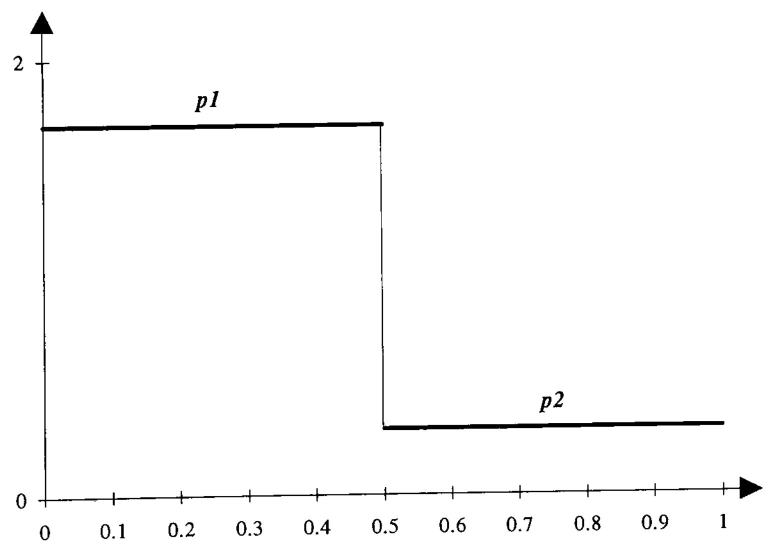


Fig. 3.11: The input density function in a Llyod's example.

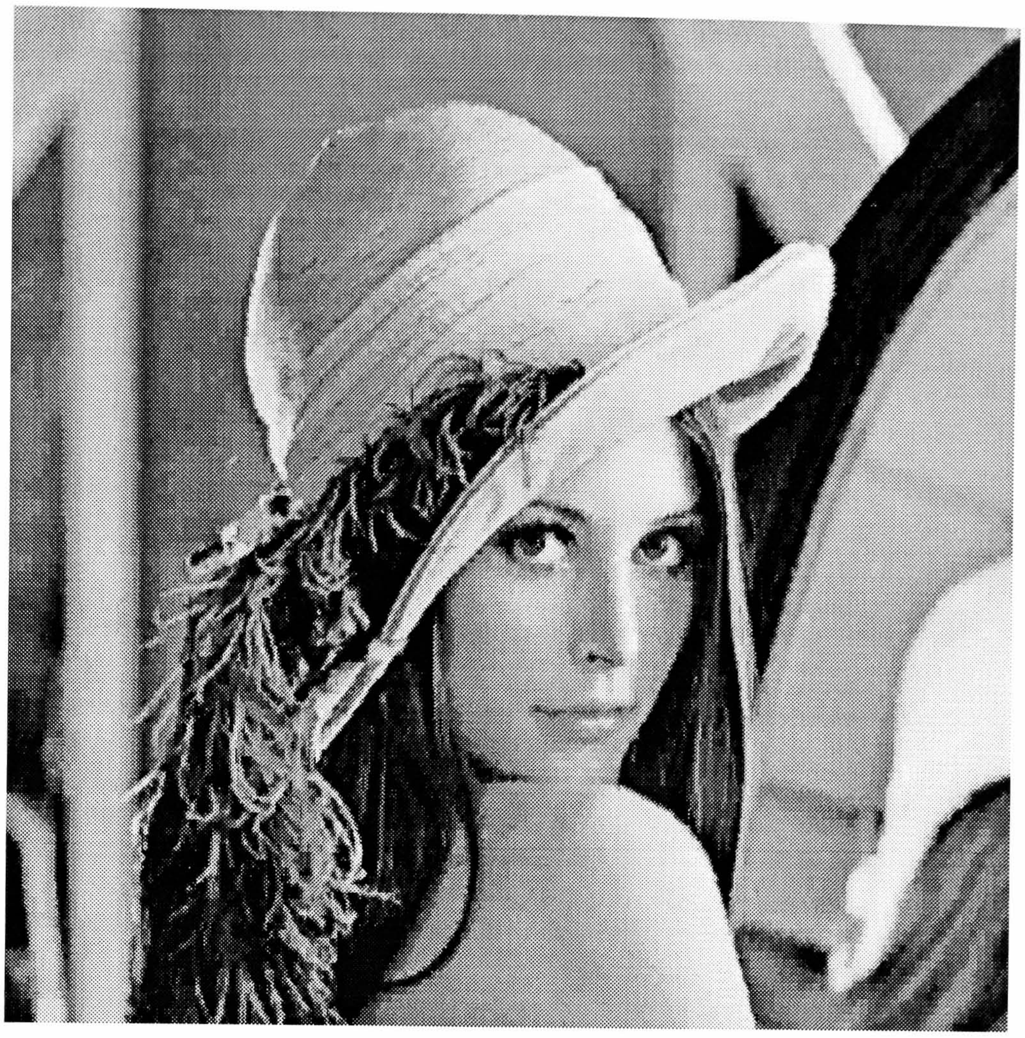
At the global minimum, in this example, the variances in two partitions are not the same, but have a smaller difference than that at the local optimum. This is a simple example showing that un-equidistortion optimal partition exists in many practical cases where the number of reference vectors is not large and the probability density is not smooth. When the conditions for the asymptotical optimal VQ are not satisfied, the equidistortion rule can not easily be applied and many so-called global optimal VQs may not lead to the true global optimum. However the proposed ECSOM algorithm works well for realistic situations, and does lead to global or "better" local minima, since the equidistortion rule in the ECSOM is applied indirectly. The effort of this constraint is to reduce the differences between variances of partitions to be as small as possible, and not necessarily to be exactly the same.

(3) ***Example 2:*** *Image compression (quantising "Lena" image)*

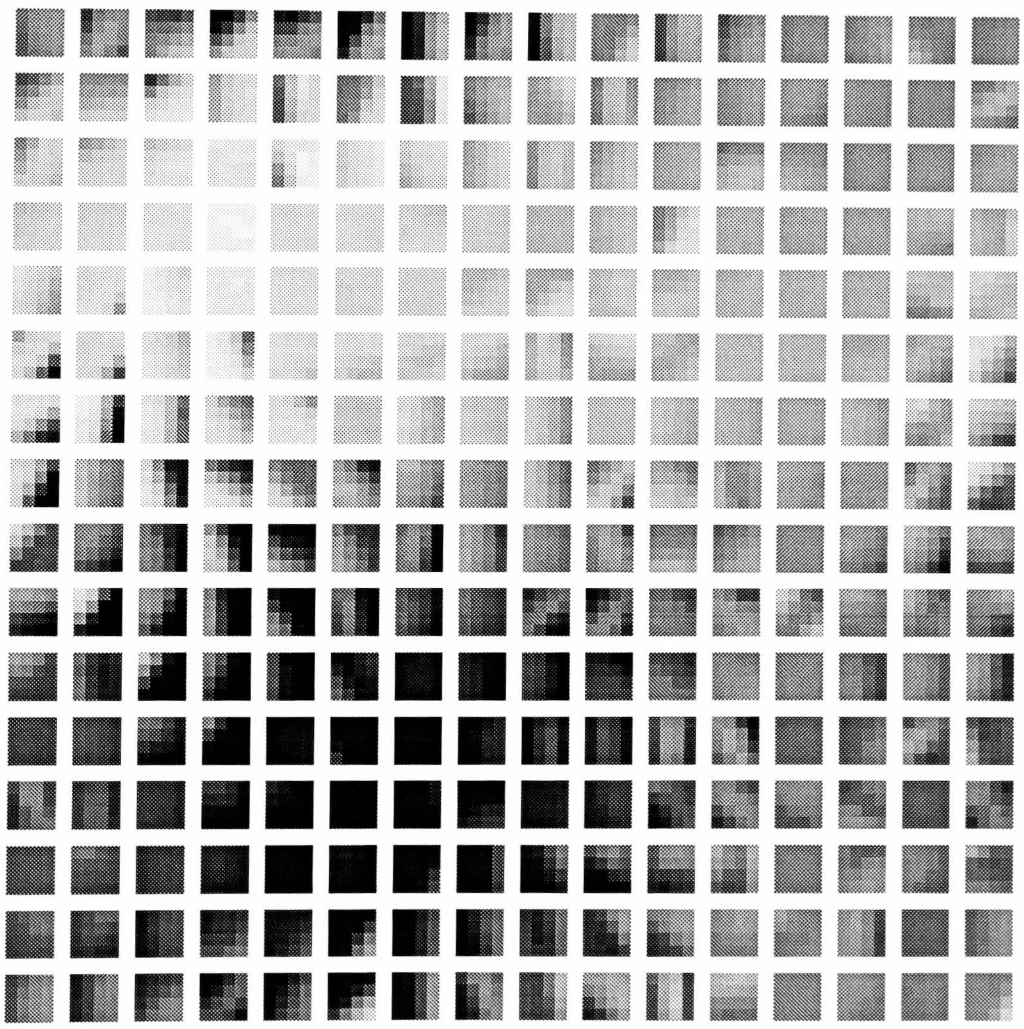
The test image is the familiar "Lena" image shown in Fig. 3.7 and is encoded using 256 4×4 references (i.e. at 0.5 bpp). For high dimensional data and/or many reference vectors, the global or local minima cannot be easily observed as in the above example. When the algorithm reaches convergence or near-convergence, it is hard to tell whether this is a global optimum, a local optimum, or a "better" local optimum unless the process is repeated many times with different parameters, and then choose the best one, i.e. the one which has the minimal total distortion. There is no general method to identify the global optimum. In real applications, it is impossible to apply the algorithm repeatedly to search for the global or a "good" local optimum. An algorithm which can generally produce good quantisations and is less demanding in parameter setting and application circumstances is most desirable in practice. The SOM algorithm is better than LBG- and CL-VQ algorithms, in the sense of it does not rely heavily on the algorithm's parameters. The SOM-VQ normally can produce better quantisation than the LBG- or CL-VQ can, when given the same initial conditions. With different initial states, the SOM-VQ yields almost the same result. Thus the SOM-VQ is a more generalised algorithm than others. The proposed ECSOM-VQ is an even more generalised algorithm. It normally yields even lower total distortion than the SOM-VQ does during extensive experiments on the data.

Typical results, after 60 iterations of the SOM- and ECSOM- VQs on the Lena image, are given in Figs. 3.12 and 3.13 respectively. Some difference between these two results, SOM- and ECSOM-VQs, can be seen from their PSNR performance comparison shown in Fig. 3.14, although differences between Figs. 3.12 and 3.13 are not easy to detect visually. It can be seen that finally the ECSOM-VQ does yield better performance, i.e. lower distortion. It can also be seen that when the constraint applies, the MSE distortion does reduce quicker. This means that the constrained processes converge faster than the original ones. These two advantages will be very useful in practical applications.

One disadvantage of SOM-based algorithms, however, is that the SOM- or ECSOM-VQ normally needs more iterations for convergence. If not implemented in parallel, the parallel updating in a neighbourhood has to be carried out serially, which will dramatically slow down the process.

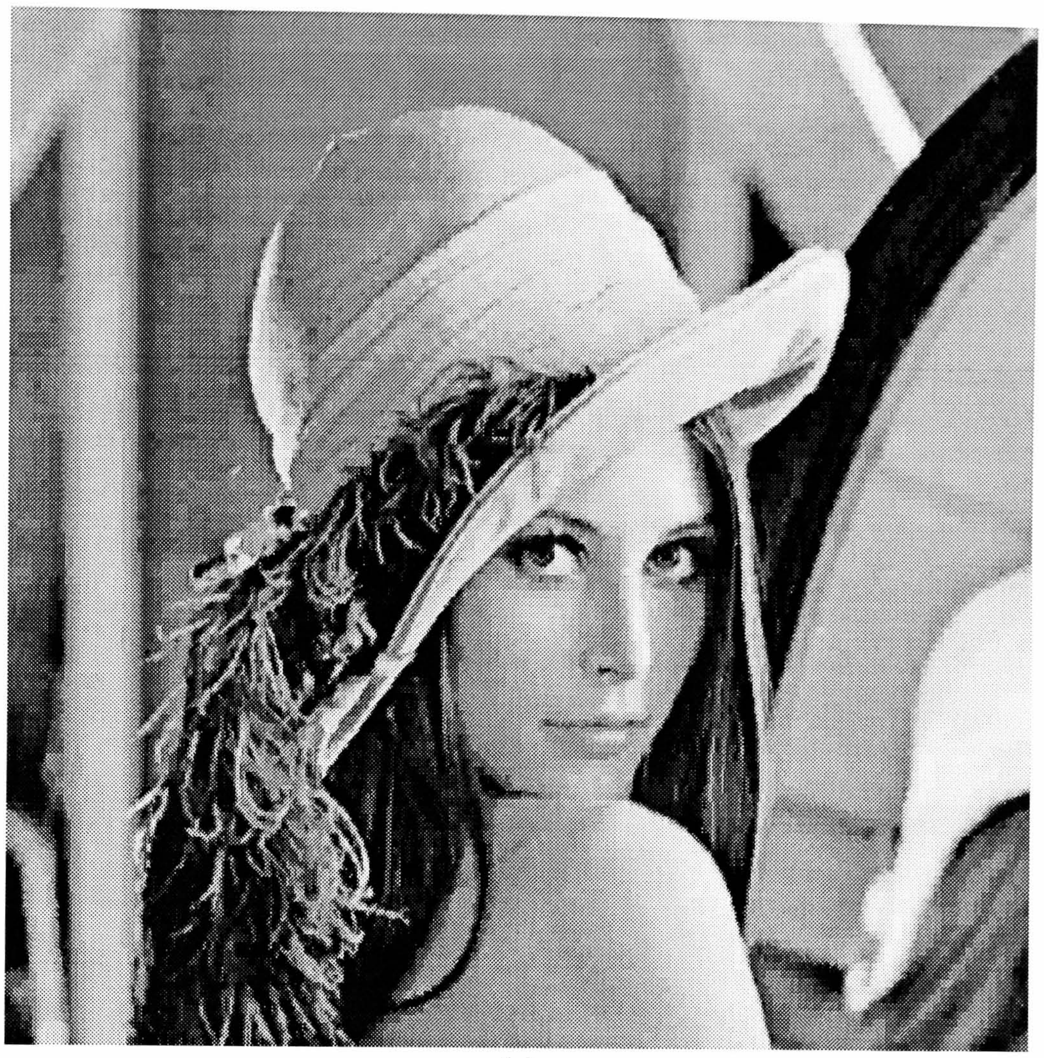


(a)



(b)

Figure 3.12: (a) SOM-VQ; (b) Its Codebook, at 0.5 bpp, 60 iterations.


(a)

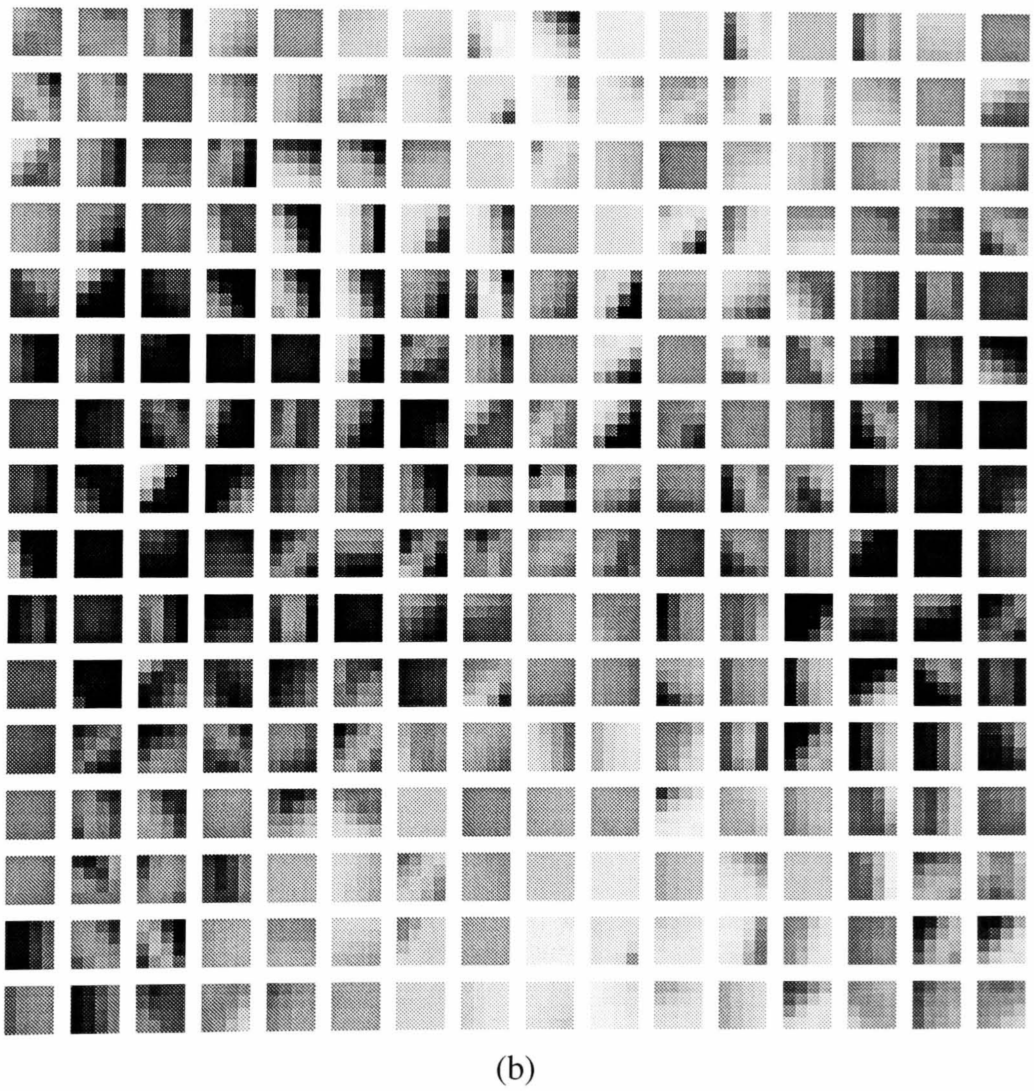

(b)

Figure 3.13: (a) ECSOM-VQ; (b) Its codebook, at 0.5 bpp, 60 iterations.

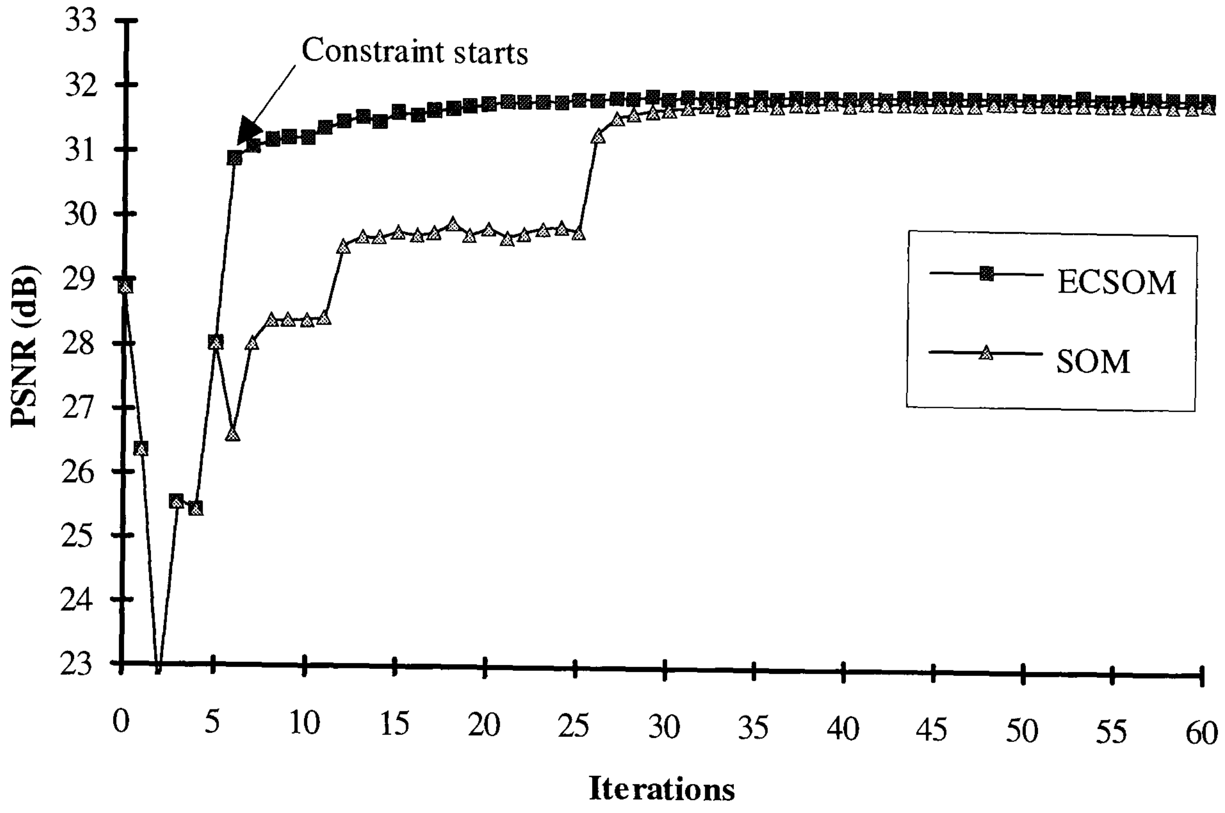


Figure 3.14: PSNR performance comparison of the SOM and ECSOM.

3.4.4 To match the input probability density

Gersho (1979) and Zador (1966, 1982) derived the asymptotic performance of the optimum VQs. The minimum distortion, i.e. the low bound, can be expressed as

$$D^* = C(N,r)K^{-r/N}\|p(\mathbf{x})\|_{N/(N+r)} \tag{3.32}$$

where $C(N, r)$ (or $A(N, r)$ in Zador's paper) is a quantisation coefficient, N is the dimension of the input space; r (normally, 2), is the power of the Euclidean norm, which is defined as

$$\|p(\mathbf{x})\|_\alpha \equiv \left[\int p(\mathbf{x})^\alpha d\mathbf{x}\right]^{1/\alpha} \tag{3.33}$$

This property shows that when the input probability density is smooth, and for a sufficient large codebook, there exists a quantiser with a total distortion arbitrarily close to this bound. It also reveals that the optimum point density $\lambda(\mathbf{w})$, i.e. the density of code references occupying in input space, should be proportional to $p(x)^{N/(N+r)}$.

The SOM algorithm is a density-related mapping. It maps more neurons to regions with high probability densities, and less neurons to the low density regions. However, it is not like Kohonen's early supposition (Kohonen 1982), i.e. the SOM approximates the input density: $\lambda(\mathrm{w}) \cong \mathrm{p}(\mathrm{x})$. Later Kohonen (1994) referred to this by stating "*It is not so important for the SOM to approximate the detailed form of the input density function, but to find its main dimensions!*". Ritter (1991) first reported his research on the point density issue of the SOM algorithm, and discovered that the asymptotic point density of the SOM in the 1-D case is in the form of

$$\lambda(w) \propto p(x)^{2/3-1/[3h^2+3(h+1)^2]} \tag{3.34}$$

where h is the number of neighbours on each side of winner for a rectangular or stepped neighbourhood function. When h=0, the SOM is reduced to the CL algorithm, and $\lambda(w)\propto p(x)^{1/3}$, which is consistent with (3.33) for the 1-D case. With $h \to \infty$, infinite neighbourhood, $\lambda(w)\propto p(x)^{2/3}$.

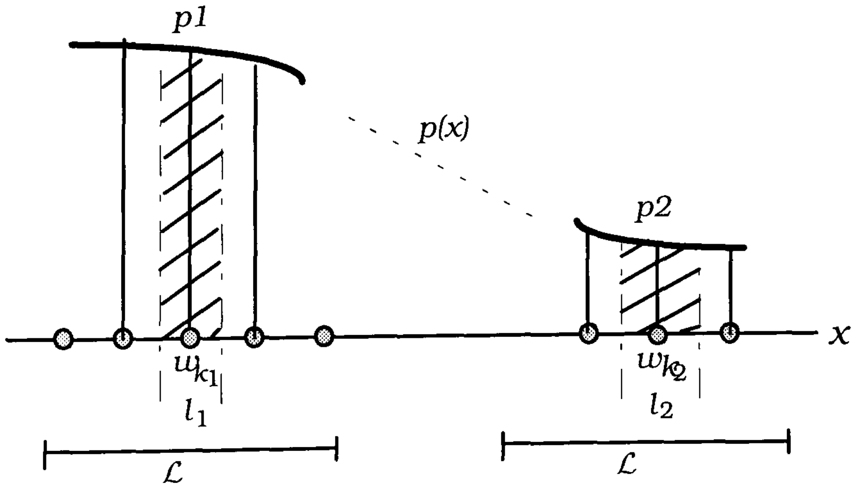
More recently, Dersch and Tavan (1995) have extended this result from stepped neighbourhood functions to any arbitrary monotonously decreasing neighbourhood functions, yielding

$$\lambda(w) \propto p(x)^{(1+12R)/3(1+6R)} \tag{3.35}$$

where R is the normalised second moment of the neighbourhood function.

Luttrell (1991) has derived a different result from Ritter's for the scalar topographic SOM, in which the monotonically decreasing neighbourhood function specifies the channel noise density function (see Section 3.3.2). In minimising the hierarchical distortion, i.e. (3.16), the point density of topographic SOMs will tend to $\lambda(w)\propto p(x)^{1/3}$ instead of (3.34) when $h\neq 0$. Luttrell (1991) explained that "*the difference arises entirely from using minimum distortion rather than nearest neighbour encoding*".

We will not discuss this issue further. This property concerns the asymptotic performance of the SOM and is valid only when the K is very large and the input density is smooth. In practical problems these conditions may not be satisfied, so the point density may be different from the above forms and may be difficult to derive, especially when local minima exist. In some cases, there are requirements for exact, or as close as possible, matching of the input density, i.e. $\lambda(w)\propto p(x)$, which the original SOM cannot do. As mentioned earlier in Section 3.4.2, the CL algorithm with a "conscience" mechanism can lead to a good matching to input density. The principle is to let the firing frequency of each neuron be the same, sometimes this is termed the *equiprobable* principle. This can be easily seen in Fig. 3.15, in which the shaded areas should be equal in order to match the input density.



since:

$$\lambda(w_{k_1}) \approx \frac{\mathcal{L}}{l_1} \quad \& \quad \lambda(w_{k_2}) \approx \frac{\mathcal{L}}{l_2}$$

let

$$\frac{\lambda(w_{k_1})}{\lambda(w_{k_2})} = \frac{p_1}{p_2}$$

we obtain:

$$P_1(\equiv p_1 l_1) = P_2(\equiv p_2 l_2)$$

Figure 3.15: The equal probability rule.

In many applications, the equiprobable principle is applied through weighting the distance measure by the firing frequency of corresponding neurons (e.g. Ahalt *et al.* 1990). This may violate the basic nearest neighbour partition principle as the number of winning times for each neuron cannot be the same in many practical cases, e.g. in Example 1 (Fig. 3.11). We have found that to match the input density, the constraint method proposed in this section can also be employed by using the firing frequencies to control the width of the neighbourhoods instead of using distortion measures.

3.5 Towards the Optimal Bayes Classifier

Another domain, in which the SOMs has been widely used, is data clustering and pattern classification. However, when used as a classifier, especially in an *1 of M* implementation (i.e. one output unit corresponding to one class), a conventional SOM will not normally perform optimal Bayesian classification unless the input data are uniformly distributed or pattern classes are well separated. This is because when the pattern classes are not well separated (or even overlapped), the distributions around boundaries are not the same for different classes and the nearest neighbour partition rule will not give Bayesian boundaries. A simple example can be seen in Fig. 3.16, when two Gaussians have equal prior probability but different means and variances.

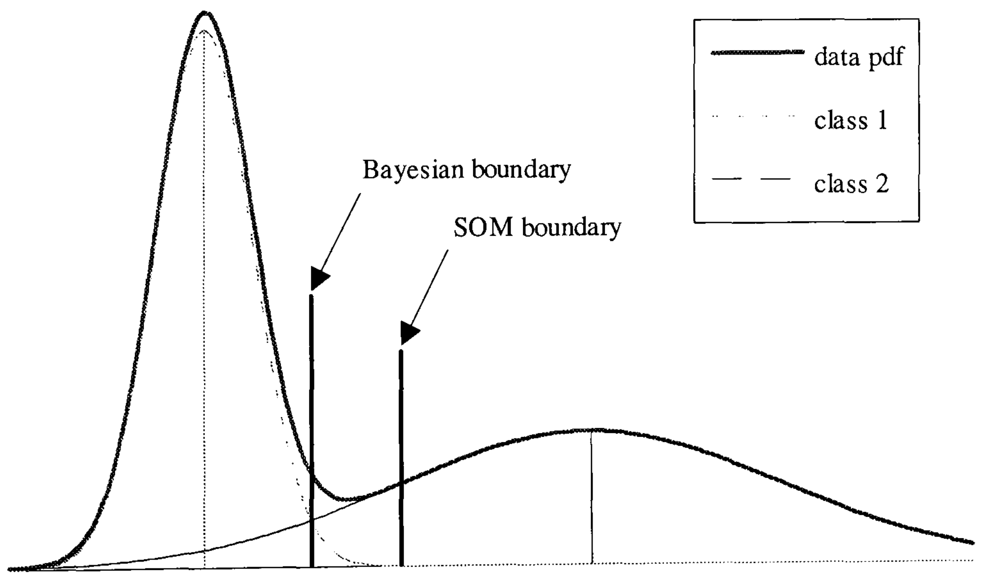


Figure 3.16: Difference between SOM and Bayesian classification.

As we have already formally proved in the last chapter, the SOM algorithm is potentially optimum for VQ. In most pattern classification cases, the pattern distributions are overlapped, and their joint distribution can be described by a *mixture distribution* (MD). Like the *k*-means algorithm, the SOM often results in more classification errors than the Bayes classifier. To form a Bayesian classification, the

SOM needs some form of supervision to label and adjust a pre-turned SOM like the two-phase LVQ1, LVQ2, and LVQ3 (Kohonen 1990) algorithms, or the SOM algorithm cascaded with another linear supervised layer (Haykin 1994). These methods, however, defeat the unsupervised learning principle, and lack rigorous mathematical proof. Thus there is no guarantee for them to form a good Bayesian classification performance. An entirely unsupervised learning algorithm, which can accurately capture the underlying MD, is highly desirable to these problems, and it would also be helpful for the supervised learning phase in the above two-stage classification methods.

3.5.1 Mixture distribution models

Mixture distribution models can be seen in many practical pattern classification applications. Each sample, **x**, from an N-dimensional input space, $\mathbf{X}\in\Re^N$, is to be assigned to one of K distinct classes, ω_1, ω_2, ...ω_K, which have prescribed class-conditional distributions. The probability density of the samples is a mixture distribution, given by (Duda and Hart 1973; Everitt and Hand 1981)

$$p(\mathbf{x}|\Theta)=\sum_{i=1}^{K} p(\mathbf{x}|\omega_i,\theta_i)P(\omega_i) \tag{3.36}$$

where $p(x|\omega_i, \theta_i)$ is the i-th class-conditional density, and θ_i are the sufficient statistics, or parameter vector for the i-th class-conditional density, $\Theta=(\theta_1, \theta_2, \ldots \theta_K)^T$. $P(\omega_i)$ is the prior probability of the i-th class and is sometimes called the mixing parameter, or mixing weight. For a Gaussian distribution, $\theta_i=\{\mathbf{m}_i, \Sigma_i\}$, where $\mathbf{m}_i$ and Σ_i are the i-th class multivariable Gaussian distribution's mean vector and co-variance matrix respectively.

For most unsupervised learning applications, only the number of classes and their class-conditional density forms are known, the other parameters have to be learnt unsupervised from a set of M un-labelled independent samples, $\chi=\{\mathbf{x}_1, \mathbf{x}_2, \ldots\mathbf{x}_M\}$. In this case, *m*aximising the joint *l*ikelihood (ML) of all observed samples, $p\{\chi|\Theta\}=\prod p(\mathbf{x}_k|\Theta)$, may lead to a singular solution. When restricted to the largest finite maximum and Gaussian components it results in the following implicit equations for the parameters (Duda and Hart 1973):

$$\hat{\mathbf{m}}_i=\frac{\sum \hat{P}(\omega_i|\mathbf{x}_k,\hat{\theta}_i)\mathbf{x}_k}{\sum \hat{P}(\omega_i|\mathbf{x}_k,\hat{\theta}_i)} \tag{3.37}$$

$$\hat{\Sigma}_i=\frac{\sum \hat{P}(\omega_i|\mathbf{x}_k,\hat{\theta}_i)(\mathbf{x}_k-\hat{\mathbf{m}}_i)(\mathbf{x}_k-\hat{\mathbf{m}}_i)^T}{\sum \hat{P}(\omega_i|\mathbf{x}_k,\hat{\theta}_i)} \tag{3.38}$$

$$\hat{P}(\omega_i)=\frac{1}{M}\sum \hat{P}(\omega_i|\mathbf{x}_k,\hat{\theta}_i) \tag{3.39}$$

$$\hat{P}(\omega_i|\mathbf{x}_k,\hat{\theta}_i)=\frac{p(\mathbf{x}_k|\omega_i,\hat{\theta}_i)\hat{P}(\omega_i)}{\sum_{j=1}^{K}p(\mathbf{x}_k|\omega_j,\hat{\theta}_j)\hat{P}(\omega_j)} \tag{3.40}$$

These equations can only be solved by using some non-linear optimisation methods. For example, the expectation-maximisation (EM) (Dempster *et al.* 1977) method has been used to obtain an iterative procedure for these parameters (e.g. Tarassenko and Roberts 1994). Normally they require very intensive computation. Only local optima can be guaranteed and results depends on the initial states. The MD model covers the overlapping of single-modelled distributions. Luttrell has extended the MD model to a broader one, called a *partitioned mixture distribution,* which covers a set of overlapping mixture distributions. He has proposed a corresponding training algorithm using the EM method (Luttrell 1994b). When the number of pattern classes is unknown, a clustering validation procedure and/or verifications from statistical hypothesis testing for individual distributions are definitely required to find a global optimum solution. Such cases, with examples, will be studied in Chapter 5. Such verifications may be needed even when the number of classes is known in order to escape from local minima.

3.5.2 Unsupervised classification learning

(1) *K-means Clustering Algorithm*

The *k*-means algorithm was proposed by MacQueen (1967) and has been widely used as a clustering algorithm. The LBG algorithm used for VQ is very similar to the *k*-means algorithm, except that the latter stops after the re-calculation of the centroids from the updated partitions, i.e. step (3) in the LBG algorithm (see Section 3.4.3) instead of step (2). That is, in the *k*-means algorithm, the final references are optimal (for the MSE distortion standard) for the final partition but not *vice versa.* However there will be no difference, or little difference, if the algorithm converges or is close to convergence. Therefore the *k*-means algorithm is also potentially optimal for VQ. For MD pattern classification problems, the algorithm works well only when the pattern classes are well apart.

(2) *CL or Winner-Take-All (WTA)-CL, and SOM*

The CL algorithm is indeed a "winner-take-all" learning algorithm, so CL or WTA-CL is an adaptive version of the LBG or *k*-means algorithm. While the SOM algorithm is similar to WTA-CL in the later stages of learning. They usually achieve very close or comparable results. The roles of introducing neighbourhood functions are to produce topographically ordered reference books, to avoid the influence of the initial states, and to escape from local minima. So the SOM algorithm may yield a slightly better performance. However they are all potential VQ algorithms, and face similar problems in pattern classification applications. Traven (1991) claimed to be first in applying neural network methods to the MD problems. In his method, the parameters of the components in a MD are obtained by using a SOM, which is trained from a set of known class samples (i.e. in supervised manner).

(3) ***Maximum Likelihood Competitive Learning (ML-CL)***

Nowlan (1990) has proposed two possible modifications to the WTA-CL algorithm for MD problems. One is the *maximum likelihood competitive learning* (ML-CL), in which the winner, v, is selected according to its weighted likelihood function instead of Euclidean distance, i.e.

$$v = \arg\max_i \hat{P}(\omega_i) p(\mathbf{x}_k | \omega_i, \hat{\theta}_i) \tag{3.41}$$

This comes from choosing the maximum component as a better approximation to the MD at $\mathbf{x}_k$ than choosing any other. However this is true only when components are well separated or slightly overlapped. The ML learning gives better clustering results than the simple Euclidean distance WTA-CL algorithm. When the components of the mixture are Gaussian with equal prior probability, the Mahalanobis distance measure is equal to the ML measure.

The other possible modification is "soft" competitive learning, in which neurons share responsibility in proportion to their posterior probabilities. In "soft" competitive learning, or sharing schemes, all other neurons in addition to the winner are updated, i.e. **all** neurons adapt to the inputs weighted by their probability distribution proportions. This may correspondingly increase the computation costs where it is not implemented in parallel, especially when the number of neurons is very large.

(4) ***Probabilistic WTA Learning***

Osman and Fahmy (1994) have proposed a so-called probabilistic WTA, in which the winner is chosen probabilistically (using a random number generator) according to the neurons' posterior probabilities, i.e. (3.40), to avoid updating all neurons' weights. This will increase the total learning iterations, because each sample has to be input very many times to let all possible units learn.

3.5.3 Bayesian SOM, an extended self-organising learning for optimal classification

In this section, the SOM algorithm has been extended and applied to the kernel learning networks for the MD. The network places K units in the input space in the same or reduced dimensionality. Each unit has a Gaussian kernel, with its mean vector, $\mathbf{m}_i$, covariance matrix, Σ_i, and mixing weight, $\hat{P}(\omega_i)$, *or* $\hat{P}_i$, as learning parameters or self-organising learning weights. At each time step, n, a sample denoted by $\mathbf{x}_k(n)$ is randomly taken from the input set χ. The winner is chosen according to its kernel output, i.e. maximum estimated posterior probability, as in Eqn. (3.41). The weights are then updated within a neighbourhood of the winner, η_v. The updating rule is modified to:

Bayesian SOM Algorithm:

$$\hat{\mathbf{m}}_i(n+1) = \hat{\mathbf{m}}_i(n) + \alpha(n)\hat{P}(\omega_i | \mathbf{x}_k, \hat{\theta}_i)(\mathbf{x}_k(n) - \hat{\mathbf{m}}_i(n)) \tag{3.42}$$

$$\hat{\Sigma}_i(n+1) = \hat{\Sigma}_i(n) + \alpha(n)\hat{P}(\omega_i | \mathbf{x}_k, \hat{\theta}_i)\{(\mathbf{x}_k(n) - \hat{\mathbf{m}}_i(n))(\mathbf{x}_k(t) - \hat{\mathbf{m}}_i(n))^T - \hat{\Sigma}_i(n)\} \tag{3.43}$$

$$\hat{P}_i(n+1) = \hat{P}_i(n) + \alpha(n)(\hat{P}(\omega_i | \mathbf{x}_k, \hat{\theta}_i) - \hat{P}_i(n)) \tag{3.44}$$

where *the adaptive gain*, $\alpha(n)$, is the same as in the original SOM. The calculation for $\hat{P}(\omega_i | \mathbf{x}_k, \theta_i)$ is same as (3.40).

As can be seen, the original SOM's neighbourhood function (fixed in shape but shrinking in size) has been replaced by an adaptive estimated posterior probability function. The neighbourhood size, which depends on the covariance of components, however, should be fixed in this case. The topological order property of the SOM ensures that the posteriors of the components that are outside the neighbourhood are very small. Therefore,

$$p(\mathbf{x}|\hat{\Theta}) \approx \sum_{i \in \eta_v} p(\mathbf{x}|\omega_i, \hat{\theta}_i)\hat{P}(\omega_i) \tag{3.45}$$

The proof of the convergence of this algorithm can be deduced by using a similar method as in Chapter 2 (Theorem 2.1 and Lemma 2.1). The learning parameters will eventually converge to the conditions (3.37)-(3.40), which is at least a local minimum.

3.5.4 Experimental results

This extended SOM, or Bayesian SOM, has been employed for some typical MD classification problems. Firstly the algorithm is applied to the 1-D example shown in Fig. 3.16. It was found that the resulting boundary is positioned very close to the Bayesian boundary; and in addition, each pattern class's pdf parameters as well as the mixing parameter were correctly estimated.

Problems arise only when the classes are very close to each other (i.e. heavily overlapped), where the result is not unique. In these situations, however, the multiple solutions are possible in reality as shown in Fig. 3.17. The MD for classes A and B and the MD for classes C and D are very close, even though the pair of classes A and B is quite different from the pair of classes C and D. This means that when the data set is not infinite, as it is often the case, the decomposition of some MDs is not unique. In this test, the MD is of two Gaussian components, which were set to m_1=0.0, σ_1=1.0, P_1=0.25, m_2=-2.0, σ_2=1.0, as shown for classes A and B in the figure. However, the results are not unique, sometimes the algorithm converges close to the pre-set parameters, sometimes it converges to another set of parameters: m_1'=-0.94, σ_1'=1.29, P_1'=0.57, m_2'=-2.30, σ_2'=0.84, as classes C and D shown in the figure. As we can see, the two possible results have very close MDs. Thus we should not be surprised about multiple results, since the problem itself is multi-resolution possible, when the components are too close to each other.

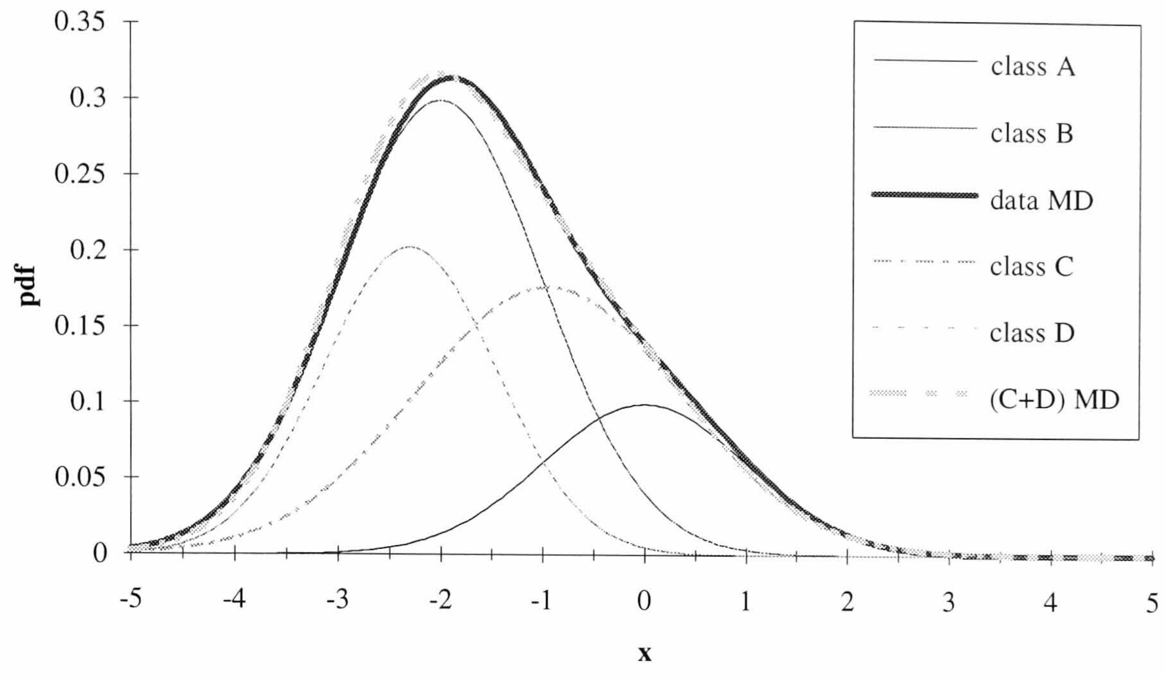


Figure 3.17: Multiple solutions for heavily overlapped MD.

The algorithm has also been applied to a practical problem, namely the unsupervised segmentation of textured images. The whole network structure will be discussed in Chapter 5. With this proposed algorithm replacing the original SOM a better estimate of the underlying patterns' mixture distribution is obtained. Improved results have been achieved and are shown in Fig. 3.18, especially the estimating layer's results (Fig. 3.18 (d) and (i) are better than (b) and (g)) by using original SOM, and they are closer to the true output. This implies that the proposed algorithm gives a better interpretation of the sample distribution.

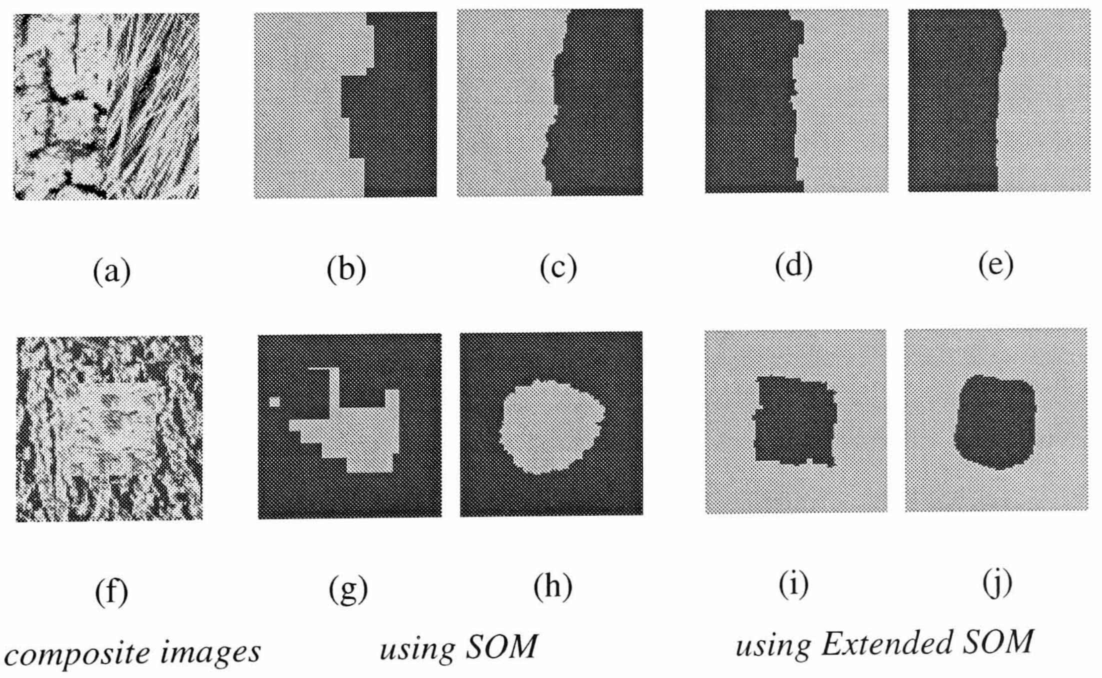


Figure 3.18: Textured image segmentation. (a) and (f) The composite textured images; (b), (d), (g), and (i) The outputs of the estimating layer; (c), (e), (h), and (j) The outputs of the whole network.

The algorithm can be used in unsupervised kernel-like learning, e.g. radial-basis-function networks, to estimate the underlying density modelled by a mixture of overlapped components. Like the SOM, the extended algorithm is a simple algorithm and easy to implement. Its neighbourhood function can form a topologically ordered map, which may provide higher noise tolerance, and makes local learning possible. It can also overcome under-utilisation or singularity problems. Since it generally is not an exact gradient descent method and is independent of initial states, it has a higher probability of escaping local minima. Proper monitoring of its learning procedure, or an on-line validation program, may be useful for forming globally optimal classification mapping.

3.6 Discussions on Topological Ordering

3.6.1 Definitions and their meanings

In attempting to form an ordered map, to examine the order, and to quantitatively measure the quality of the order, a clear definition of *order* is needed.

However, what is order, what is an ordered map, and how well is a map ordered? These questions have not been fully answered. Even a general definition has not been formally given except for the very obvious 1-D definition (Kohonen 1984), in which order means sequenced neurons have their weight values arranged either monotonically increasingly or monotonically decreasingly. When this idea is extended to higher dimensions (which is not so simple as Kohonen originally proposed), it defines a *fully ordered map*, i.e. it requires that any neuron, for which any possible sized neighbourhood in the neuron space, should reflect or preserve the neighbourhoods of the input space. Such fully ordered maps may not exist when the dimensions of input and output space are not the same. This means that when the mapping is from a high dimension space onto a low dimensional space, the original 1-D order definition can not be extended. From topology space concepts (Mendelson 1990), order is not defined and only the topology preserved mapping can be defined. SOMs will always result in topology preserved mappings as long as no neuron represents the "empty set" in the input space (i.e. no "dead" neurons) and the mapping is one-to-one (not one-to-many), since the definition of neighbourhood is too broad in topological theory. The neuron spaces are discrete lattices, and the neighbourhood concept in neuron space is some forms of regular lattice around certain neurons and is not like the very general neighbourhoods of topology. It seems that there are two ways to define the order in the SOMs: one is a geometric or single-distortion definition, the other is a group-distortion or channel-optimised definition.

We first present a ***generalised definition,*** which covers the original one, gives clear geometric meaning, and is quantitatively measurable. Then we discuss a ***strict definition,*** which is directly related to the fault tolerance requirement so is meaningful in optimisation terms.

(1) ***Definition One, Generalised Ordering Definition***:

Using the same notations as before, we assume that a N-dimensional input space $\mathbf{X}$ is mapped to a reference map $\mathbf{Y}$ which is arranged in M-dimensional space. $\mathbf{Y}$ is a

$C_1 \times C_2 \times \ldots C_M$ array, where $\{C_k, k=1, 2, \ldots M\}$ represents the number of neurons along each dimensional side of output space, and the total number of neurons is $C_1 \times C_2 \times \ldots C_M$. Each neuron is indexed fully by a vector, $\mathbf{c}=[i_1, i_2, \ldots, i_M]^T$, which represents a point in the neuron map or output space, where $\{i_k=0, 1, 2, \ldots, C_k-1, k=1, 2, \ldots M\}$.

Let us first define a *hypercubic neighbourhood* in the neuron space (the neighbourhood structure depends on the neuron structure used).

If a set of neurons is indexed by

$$\aleph_{\mathbf{c}}^{(m)} = \{\mathbf{c} + \begin{bmatrix} \pm j_1 \\ \pm j_2 \\ \cdots \\ \cdots \\ \pm j_k \\ \cdots \\ \pm j_M \end{bmatrix}, j_k = 0,1,\ldots m,\ and\ 0 \le (i_k \pm j_k) \le C_k - 1,\ k = 1,2,\ldots,M\} \tag{3.46}$$

where $\{i_k, k=1,2,\ldots,M\}$ is the index of neuron $\mathbf{c}$. Then $\aleph_{\mathbf{c}}^{(m)}$ is called the m-th order hypercubic neighbourhood of neuron $\mathbf{c}$. $\aleph_{\mathbf{c}}^{(0)}$ is $\mathbf{c}$ its self.

A generalised hyper-rectangular neighbourhood, in which the neighbouring radii along each dimension may not be the same, can be readily extended from the above.

A. Zero-order map

This term (zero-order) has been used by Kangas *et al.* (1990) to identify maps without any topological ordering properties.

B. 1st-order map

A map, in which each neuron has its nearest neighbouring neurons along each dimension of output space closest to it in input space among other neurons in that neighbouring neuron's direction. In mathematical terms, it can be expressed as:

$d(w_c, w_{c_l^{(1)}}) \le d(w_c, w_{c_l^{(1)+d}})$, $\forall c \in Y;\ \forall c_l^{(1)} \in \aleph_c^{(1)}$, *$c_l^{(1)}$ is a neuron in the 1st-order neighbourhood of c; $c_l^{(1)+d} \notin \aleph_c^{(1)}$, and is called an extension of $c_l^{(1)}$, defined by*

$$c_l^{(1)} = c + \begin{bmatrix} \pm l_1 \\ \pm l_2 \\ \cdots \\ \cdots \\ \pm l_k \\ \cdots \\ \pm l_M \end{bmatrix}, l_k = 0,1;\ then\ c_l^{(1)+d} = c + \begin{bmatrix} \pm(l_1 + d_1) \\ \pm(l_2 + d_2) \\ \cdots \\ \cdots \\ \pm(l_k + d_k) \\ \cdots \\ \pm(l_M + d_M) \end{bmatrix}, d_k \begin{cases} = 0,\ if\ l_k = 0, \\ = 1,2,\ldots,\ otherwise, \\ (k = 1,2,\ldots M) \end{cases} \tag{3.47}$$

where d( ) is a distance measure, usually the Euclidean distance.

C. 2nd-order map

A map, in which each neuron has its nearest neighbouring neurons in each dimension of output space closest to it in input space, and has its second nearest neighbouring neuron second closest to it, in all directions. These can be expressed as:

$$d(w_c, w_{c_l^{(1)}}) \le d(w_c, w_{c_l^{(1)+1}}) \le d(w_c, w_{c_l^{(2)+d}}),\ \ \forall c \in Y;\ \forall c_l^{(1)} \in \aleph_c^{(1)},\ c_l^{(2)} \in \aleph_c^{(2)},$$
$$c_l^{(1)+1} \textit{ is an extension of } c_l^{(1)} \textit{ but within } c_l^{(2)};\ c_l^{(2)+d} \textit{ is the extensions of } c_l^{(1)+1}. \tag{3.48}$$

......

This recursive definition can be extended to ***m-th order ordered map.***

......

Here, order or order-level, m, could be up to min$\{C_k$, i.e. the number of neurons on each dimensional side, k=1, 2, ...$M\}$ if an open map is used, or half the number of neurons on each dimensional side if a wrapped map structure is used. We say that the above definition is a single-distortion definition because the order is measured according to a single distance. If in a map, each neuron satisfies the above m-th-order ordering definition, we can state the map is a **globally m-th-order ordered map**. If only some neurons meet the conditions, the map is said has **local m-th-order ordering.** It is not difficult to prove that if the neighbourhood function is fixed to a m-th neighbourhood (step or monotonically decreasing), the resulting map will be a globally m-th-order ordered map or will have some local m-th-order ordered areas.

When a map is ordered to the highest order level in each of its dimensions, then it is a **fully ordered map**, which may exist only when the dimensions of input and output spaces are the same. When the mapping is from a high dimensional space onto a lower dimensional space, only low order ordered maps exist. In some cases, even 1st-order fully ordering is impossible. An **optimum ordered map** is the one which not only has the required order-level of ordering but also has minimised the MSE distortion, D1, Eqn. (3.15) (i.e. the neurons has been mapped to the optimum positions). Otherwise the map can be only called a local ordered or disordered optimum map; or ordered but non-optimal map; or non-optimal and disordered map.

Considering the example used in Section 2.5.3, if the neurons are arranged in the same dimension (2 in this case), a fully ordered map can be easily obtained as shown in Fig. 3.19 (a). Since there are only three neurons along each dimensional side in neuron space, the highest possible ordering order is 2 if an open map structure is used, or 1 if a wrapped map structure is used (in this case the input space would also be wrapped). When the output space is only 1-D (i.e. a neuron chain), only a 1st-order ordered map is possible for the SOM algorithm as shown in Fig. 3.19(b), if an open map structure is used. The maps shown in Fig. 3.19(c) and (d) are possible higher-order ordered maps for a wrapped structure (at least to 3rd- or 4th-order) and for a non-wrapped structure (up to 8th-order one side), but neurons are not at the optimal positions so they can not be called optimal maps as they will not minimise the MSE distortion.

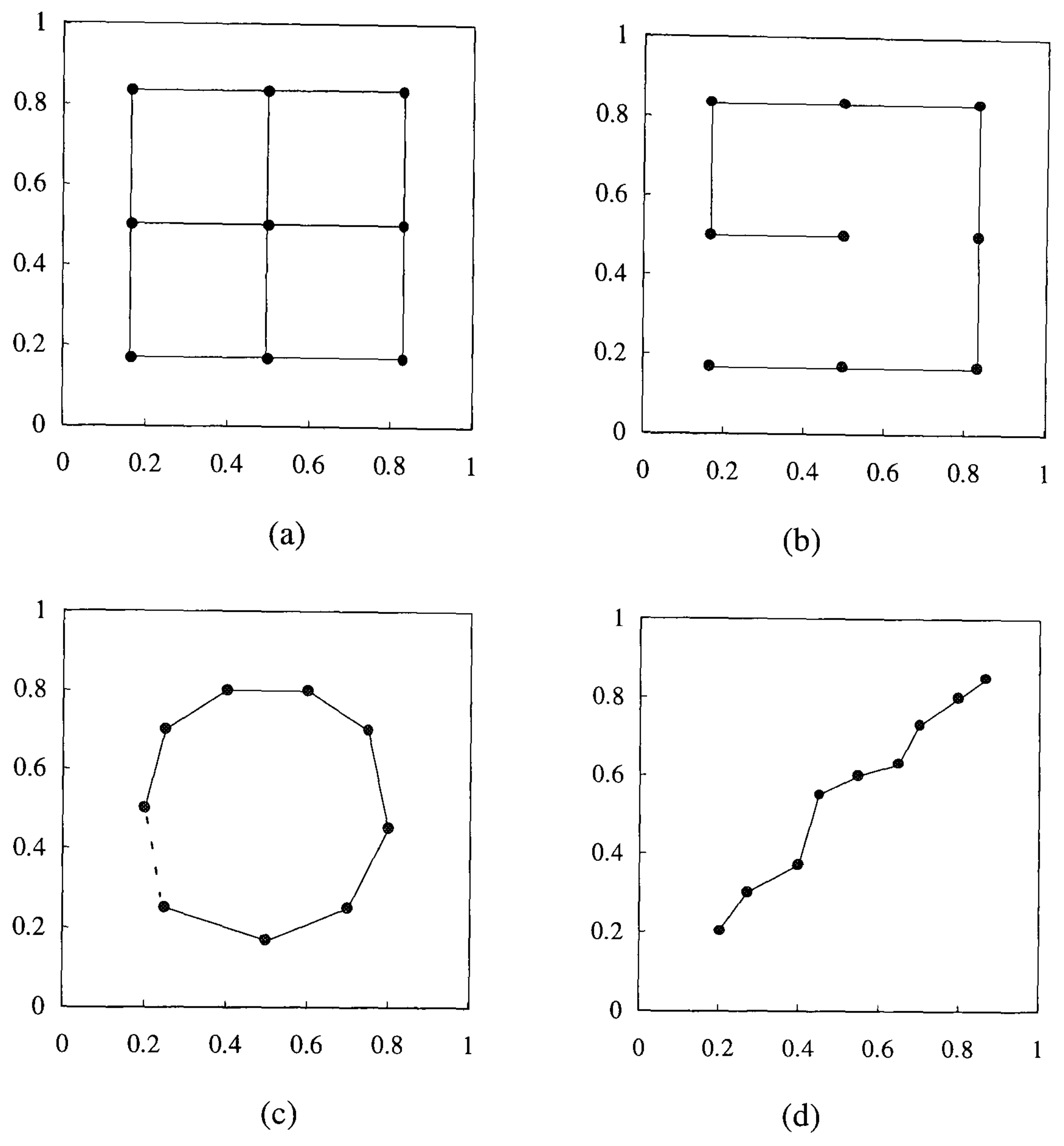


Figure 3.19: Different ordering situations.

The meaning of this kind of ordering definition is that the ordered map can tolerate the corresponding levels of *directional* channel or recalling pattern noise, and is optimal for decoding (but not for encoding) with overall minimum distortion and channel error. That is, if the reference book is fixed and the channel noise level is not above the order level, then the decoded signals have the minimum error in the corresponding directions. If a code error occurs on the k-th dimension of the code vectors and the error bias is a positive value, say, *one*, then the corresponding biased code will have less distortion than all neurons whose bias value is greater than *one*. For example, the map in Fig. 3.19(b) will produce less errors than other maps when the channel noise is limited only to 1st-order, as it is a 1st-order ordered maps.

Arranged in a lower dimension, the map may have advantages in the sense of less code errors, when channel noise exists. High dimensional codes increase channel errors. For example, using 2-D codes may double the code errors of using 1-D code. However the mapping from high dimension to low dimension may reduce the possible order-levels, i.e. the noise tolerance of the map.

The next definition of ordering is stricter in the sense of error-tolerance encoding and decoding.

(2) ***Order Definition Two, Strict Ordering Definition:***

When the channel noise has to be taken into account, the encoding must be considered in a different way. Its objective function is now the hierarchical mean squared error distortion, i.e. D_2, Eqn. (3.16), instead of D_1. Luttrell (1989a, 1989b, 1994a) has developed a hierarchical self-organising encoding network to minimise D_2. Although the source coding and error correcting channel coding are usually separately designed in many practical cases, the important properties of such combined, noise tolerant encoding cannot be ignored. It will provide benefits not only for signal encoding, but also for information compression, associative memories, and pattern recognition.

Luttrell used the variational principle to analyse the hierarchical encoding in the SOMs. His hierarchical self-organising network is similar to the SOM, but with the noise pdf (fixed) replacing the normal neighbourhood function (shrinking). In addition, the nearest neighbour winning rule must be replaced by the so-called *"local" minimum distortion winning rule,* which can be expressed as

$$\mathbf{v} = \arg\min_{\mathbf{c}\in\mathbf{Y}} \{ \int d(\mathbf{w}_\mathbf{c} - \mathbf{x})\pi(\mathbf{c})d\mathbf{c} \} \tag{3.49}$$

where the integration is over the range that code noise density covers.

This winning rule is different from the nearest neighbour rule, as the distortion curves to each dimension may not be symmetric (Luttrell 1994a). When trained using the above minimum distortion rule, for a global optimum in terms of total distortion and channel errors, the resulting map should also be ordered very strictly to fit any directional noise.

The definition of ordering and the requirement for the ordering depend on the pdf of the channel noise. There are two common noise situations: uniform and symmetric (decreasing) distributions. Without losing generality, the independent identical distributed properties of the noise in each dimension of reference vectors and Gaussian distribution for symmetric noise are assumed.

A. ***1st-order map***

if the code errors or channel noise pdf only spread to the nearest neighbouring codes.

(a) Uniform noise:

$$d(\mathbf{w}_\mathbf{c}, \mathbf{w}_{\mathbf{c}^{(1)}(k)}) \le d(\mathbf{w}_\mathbf{c}, \mathbf{w}_{\mathbf{c}'}),\ \ \forall \mathbf{c}\in\mathbf{Y};\ \forall \mathbf{c}^{(1)}(k)\in\aleph_\mathbf{c}^{(1)},\ \forall \mathbf{c}'\in\mathbf{Y}(but \notin \aleph_\mathbf{c}^{(1)}) \tag{3.50a}$$

For every neuron, any other neighbouring neuron (except the 1st-order neighbouring neurons) is further (i.e. has higher distortion) from it than any of its 1st-order neighbouring neurons are.

(b) Gaussian noise:

$$\begin{aligned}&\text{If } d(\mathbf{c},\mathbf{c}^{(1)}(k)) \le d(\mathbf{c},\mathbf{c}'), \text{ then}\\ &d(\mathbf{w}_\mathbf{c}, \mathbf{w}_{\mathbf{c}^{(1)}(k)}) \le d(\mathbf{w}_\mathbf{c}, \mathbf{w}_{\mathbf{c}'}),\ \forall \mathbf{c}\in\mathbf{Y};\ \forall \mathbf{c}^{(1)}(k)\in\aleph_\mathbf{c}^{(1)},\ \forall \mathbf{c}'\in\mathbf{Y}(but \ne \mathbf{c},\mathbf{c}^{(1)}(k))\end{aligned} \tag{3.50b}$$

*where d(***c***,* **c****) is the distance of two neurons indexed by* **c** *and* **c**** in neuron space.*

Every neuron with its 1st-order neighbouring neuron has a similar weight distance hierarchy in the input space as it has in the neuron space.

B. 2nd-order ordered map

If the code errors or channel noise pdf can happen to not only the nearest neighbouring codes, but also the second nearest neighbouring codes.

(a) Uniform noise:

$$d(\mathbf{w}_{\mathbf{c}}, \mathbf{w}_{\mathbf{c}^{(2)}(k)}) \le d(\mathbf{w}_{\mathbf{c}}, \mathbf{w}_{\mathbf{c}'}), \ \forall \mathbf{c} \in \mathbf{Y}; \ \forall \mathbf{c}^{(2)}(k) \in \aleph_{\mathbf{c}}^{(2)}, \ \forall \mathbf{c}' \in \mathbf{Y}(but \notin \aleph_{\mathbf{c}}^{(2)}) \tag{3.51a}$$

(b) Gaussian noise:

$$\begin{aligned}&\text{If } d(\mathbf{c}, \mathbf{c}^{(2)}(k)) \le d(\mathbf{c}, \mathbf{c}'), \text{ then} \\ &d(\mathbf{w}_{\mathbf{c}}, \mathbf{w}_{\mathbf{c}^{(2)}(k)}) \le d(\mathbf{w}_{\mathbf{c}}, \mathbf{w}_{\mathbf{c}'}), \ \forall \mathbf{c} \in \mathbf{Y}; \ \forall \mathbf{c}^{(2)}(k) \in \aleph_{\mathbf{c}}^{(2)}, \ \forall \mathbf{c}' \in \mathbf{Y}(but \ne \mathbf{c}, \mathbf{c}^{(2)}(k))\end{aligned} \tag{3.51b}$$

......

This recursive definition can be extended to ***m-th-order map.***

......

This kind of ordering has the greatest channel noise tolerance. For example, if the channel errors can be up to an m-th neighbourhood, an m-th-order ordered map could produce less errors than lower-order ordered maps could, i.e. within error occurring area (m-th neighbourhood) distortion is smaller than outside the area when noise is uniform distributed, or the distortion is smaller if the error probability is higher when the noise is symmetric and monotonically decreasing Gaussian distributed. This is why the ordered map will give the best decoding performance up to the order-level channel noise. With its local minimum distortion winning rule (encoding rule) instead of distance winning rule (nearest neighbour encoding), the resulting map can be the global optimal map (for both encoding and decoding).

3.6.2 Ordering measurements

Kohonen has given a measure for disorder of a map in the 1-D case, which is quite obvious (Kohonen 1984),

$$DIS_1 = [\sum_{i=1}^{M-1} d(w_i - w_{i-1})] - d(w_{M-1} - w_0) \tag{3.52}$$

Carrato (1994) extended this measure to any (assume M) dimension maps:

$$DIS_M = \sum_{i_1=1}^{C_1-1} \cdots \sum_{i_{M-1}=1}^{C_{M-1}-1} \{ [\sum_{i_M=1}^{C_M-1} d(\mathbf{w}_{i_1,i_2,\dots i_{M-1},i_M+1} - \mathbf{w}_{i_1,i_2,\dots i_{M-1},i_M})] - d(\mathbf{w}_{i_1,i_2,\dots i_{M-1},C_M-1} - \mathbf{w}_{i_1,i_2,\dots i_{M-1},0})\}$$

$$+ \sum_{i_1=1}^{C_1-1} \cdots \sum_{i_{M-2}=1}^{C_{M-2}-1} \sum_{i_M=1}^{C_M-1} \{[\sum_{i_{M-1}=1}^{C_{M-1}-1} d(\mathbf{w}_{i_1,i_2,\dots i_{M-1}+1,i_M} - \mathbf{w}_{i_1,i_2,\dots i_{M-1},i_M})] - d(\mathbf{w}_{i_1,i_2,\dots,C_{M-1}-1,i_M} - \mathbf{w}_{i_1,i_2,\dots,0,i_{M-1}})\}$$

$$+ \dots\dots$$

$$+\sum_{i_2=1}^{C_2-1}\cdots\sum_{i_{M-2}=1}^{C_{M-2}-1}\sum_{i_M=1}^{C_M-1}\{[\sum_{i_1=1}^{C_1-1}d(\mathbf{w}_{i_1+1,i_2,\dots i_{M-1},i_M}-\mathbf{w}_{i_1,i_2,\dots i_{M-1},i_M})]-d(\mathbf{w}_{C_1-1,i_2,\dots,i_{M-1},i_M}-\mathbf{w}_{0,i_2,\dots,i_{M-1},i_M})\} \tag{3.53}$$

This kind of disorder measure actually measures the excessive length or straightness (geometric regulation, or rectangularness, or squareness) of the map, which is equal to a sum of sub-sums over all dimensions of the output space; and each sub-sum is a sum of all segments connecting adjacent neurons along one dimension of neuron space subtracting the length between the first and last neurons in that dimension. For the 1-D case, DIS_1=0, if the map is ordered. While for higher dimension cases, DIS_M=0, only when the maps are perfect M-dimensional rectangular lattices. However in higher dimensions, this will often not be the case unless the input space is uniformly distributed. Carrato extended this measure in order to compare the SOM with other non-ordering VQ algorithms (e.g. LBG, CL) and showed the advantages of ordered codebooks for VQ.

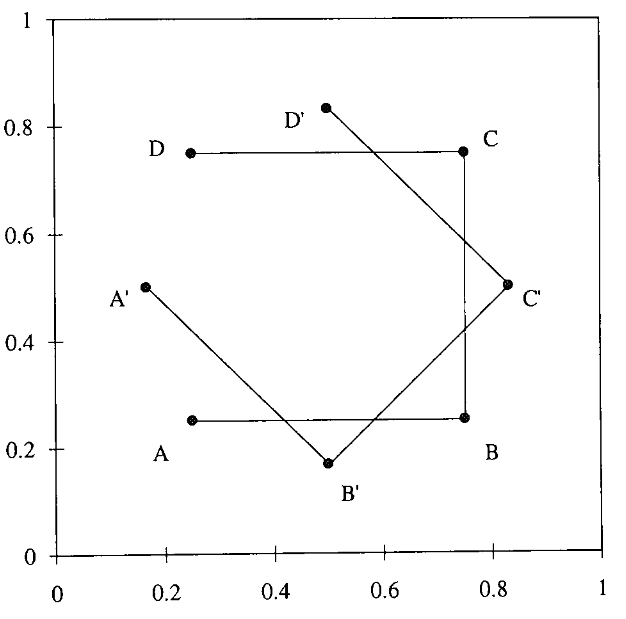


Figure 3.20: Two possible globally ordered chains.

This measure, however, cannot be used to indicate whether a map is ordered or not. Its value also cannot be used as a judgement of the goodness of the ordering of the map. This has two implications. One is that the measure (3.53) cannot be used to assist in the training, since some disordered maps could have a lower value than an ordered map! For example, a map condensed to the centre of the input space (e.g. a small random initial map) has a smaller DIS_M value than a fully expanded and ordered map (e.g. Figure 3.19 (b)). The second is that it cannot be used to judge the goodness of the final SOM results when local minima exist. In this case, some local minimum map (but ordered) could have a lower disorder index value than the global one. For example in Fig. 3.20, a four-neuron chain is mapped to a 2-D square. The global minimum reference map is shown by A-B-C-D; while a local minimum map is shown as A'-B'-C'-D', whose dis-order index is $(\sqrt{2}/3+\sqrt{2}/3+\sqrt{2}/3)-\sqrt{2}/3=2\sqrt{2}/3=0.9428$, which is smaller than (0.5+0.5+0.5)-0.5=1.0 of the global one. However, the global optimum map will produce less MSE distortion than the local one.

Bauer and Pawelizk (1992) have proposed a neighbourhood preservation measure by using topographic products. Their definition for neighbourhood is similar to ours, but they grade the neighbourhood order-levels entirely by distance. So our first-order neighbourhood would be their first- and second- order neighbourhoods; our second-order neighbourhood would be their third-, fourth-, and fifth- order neighbourhoods, and so on. They measure the distance hierarchies of the map's various neighbourhoods in the input space, but require distances from a same order neighbourhood to be the same as for a well preserved mapping. This measure is the first which can be used to quantitatively measure the ordering for more than one dimensional space. However no optimisation meaning has been provided.

In my opinion, order can only be measured by using its definition. A globally ordered map should have **at least** the first-order ordering property, which is defined by (3.47) for the general purpose SOM (e.g. the original SOM, or normal SOM-VQs), or defined by (3.50) for the error tolerance coding SOM. The ordering can be measured by the order-levels of the maps. If the order-level is below one, the map is disordered or at most a local ordered or partly ordered (1st, 2nd, ...) map. Actually in many applications of the SOM, the mapping is from a very high dimension to a much lower dimension. Only local ordering can be found in the resulting maps; the global one may not exist. For example, in the SOM-VQ and ECSOM examples in Section 3.4, many local ordered areas (but not global ones) can be seen in the resulting codebooks (see Figs. 3.9(b), 3.12(b), and 3.13(b)).

3.6.3 The impact of neighbourhoods on the ordering

The neighbourhood function has three distinct and important roles in SOMs: preventing under- and/or over- utilisation and initial effects, escaping local minima, and producing ordered maps. These roles have already been demonstrated through various analyses and examples in the previous sections.

The realisation of ordered maps (by whichever definition) is indeed dependent on the appropriate implementation of the neighbourhood functions. For the ordinarily defined (Definition one) ordered maps, the original SOM together with a proper shrinking speed for neighbourhood function, or the proposed ECSOM algorithm, can produce the required order ordered maps if the neighbourhood function remains long enough at the required order level before it goes to zero. Care also should be taken to leave the learning rate large enough to move neurons to their optimal positions.

For the second type ordering, Luttrell's self-organising networks with a fixed neighbourhood function (which is identical to the channel or decoding noise pdf) may produce the required maps. However, there are still arguments that the neighbourhood function should be wider at the beginning and then gradually shrink to the noise pdf, so that the updating is undertaken in a wider area. Otherwise the algorithm would be like a group WTA-CL algorithm, as the winner is selected according to the local distortion of a neighbourhood area; updating is also in the same area (not in a larger area). Thus it will have similar problems as the CL algorithm, such as local minima and under-utilisation.

It is worth mentioning that if a mapping within the same dimension is required, the map can be initialised in an ordered fashion. There is no point in wasting time in the ordering phase; one can just concentrate on the convergence phase.

3.7 Conclusions

In this chapter, a full treatment of the SOM algorithm for a wide range of applications has been presented. This chapter has analysed and exploited the SOM's potential in its application areas, and corresponding modifications and extensions have been given so that the best performance may be achieved. Two different modifications, the ECSOM and Bayesian SOM, to the SOM algorithm for two of its major application domains: VQ and pattern classification, have been proposed. They could be useful for many practical tasks. Kalman-filtered SOM is an example to show that traditional optimal filter theory can be applied to neural networks in order to reduce the dynamic learning noise and improve convergence performance. The SOM's convergence can also be accelerated by the proposed constrained ECSOM algorithm, which is aimed at forming a global or near-global optimum VQ.

The proposed Bayesian SOM solved the non-optimal problems of the original SOM when used as a classifier. Bayes's theorem has been incorporated into the SOM algorithm, and results in this extended SOM algorithm, which can converge to the Bayesian boundaries in classifying MD-modelled input data structures.

A good signal representation (e.g. by optimal VQ) may be helpful in pattern classification as well. For example, when the neurons are mapped (unsupervised) to a representation space with equal distortion for each neuron, the next supervised stage labels neurons corresponding to sample patterns. The network may then have minimum errors for later classification! A good classification may also be beneficial in VQs, as in some VQ designs pattern classification is also incorporated (Oehler and Gray 1995).

The definitions for the ordering of maps have been proposed for any dimension in two ways: geometric and fault tolerant. They have clear optimisation meanings. They are quantitatively measurable and can be used to judge the goodness of the ordering of maps. These definitions give us a better understanding about the advantages of an ordered map.

Chapter 4

IMAGE TEXTURES AND MARKOV RANDOM FIELDS

Texture analysis is a basic and important methodology in image processing and computer vision. Texture properties describe the spatial-relationships between image elements. Together with grey levels and colours, they define the categories and attributes of image objects, and provide these images with their distinct visual characteristics. In this chapter, the definitions of textures, visual perception of image textures, and textural analysis methods (such as statistical feature based, model based, and multichannel-filter based approaches) are reviewed. In particular, the Markov random field, Gibbs distribution, and related texture approaches, are extensively analysed and explained. Some commonly used estimation techniques for model parameters (such as least square, maximum likelihood, maximum pseudo-likelihood) are summarised and discussed in a logical order. Texture related image processing problems are also briefly addressed. All this paves the way for more directed work in the next chapter on textured image segmentation.

4.1 Introduction

Textures are image primitives upon which human visual perception and discrimination are, in part, based. Image classification, segmentation, and other processing employing the image's textural characters have played an important role in the study of images. However, a precise and rigid definition for texture does not exist. Perhaps the main difficulty is that there is a large variety of textural attributes. It is extremely difficult to produce a single precise definition to cover such a diversity.

A very general and dictionary definition of texture has been given by Ballard and Brown (1982) as "*something composed of closely interwoven elements*".

Textures can be classified into two major categories: *structural* and *statistical*. Some images may contain attributes of both. Natural textures can be either structural or statistical; or both. Structural textures are regarded as containing, or being composed of, some image primitives of various shapes and sizes, with many placements. Image primitives play a strong role in structural textures. Statistical textures are regarded as realisation samples from spatial random processes which can be described in a statistical sense by the dependence or interaction between the elements of the image. Image primitives play a weak role in the statistical textures. They are mostly irregular and the placement rules are random. The textures which are addressed in this thesis belong to this second category. Approaches to statistical textures have been studied for many years. Haralick (1979) has given a comprehensive review of most classical methods in this area. Tuceryan and Jain (1993) have presented a review on more recent progress in texture analysis.

Approaches to statistical textures have fallen into two major methodologies: feature-based analysis and model-based analysis. The feature-based analysis attempts to find a set of good statistical features which can be used to describe and classify the textures. Substantial research has been done on this basis. A comprehensive survey of the most well-known and commonly used texture features (such as auto-correlation functions, Fourier transforms, spatial grey-tone (grey-level) co-occurrence matrices, grey-tone run lengths) can be found in Haralick (1979, 1986). In the model-based texture approach, however, textures are described by a mathematical process, in which a set of parameters can be extracted as textural features for description and discrimination (Kashyap 1986; Chellappa *et al.* 1993). Furthermore, these models can be used to reproduce or regenerate textures which provide us with a physical and visual measurement on the "goodness-of-fit" of the model. Typical texture models include: time series; fractals; random mosaics; autoregression (AR); Markov random fields (MRFs); Gibbs random fields (GRFs). The practical evaluation of a model is between its generality and its complexity. A good model should be physically meaningful with a wide application range and a simple model structure. MRF models seem to be very attractive models and have received intensive attention in scene and texture processing over recent years. Gibbs distribution (GD), a MRF equivalence, has also received a great deal of attention in texture analysis. MRF and GRF both belong to the same stochastic process approach, but through different mathematical descriptions. In MRF, the characteristics of a texture are described by local conditional probabilities; while in GRF, a general form of a joint probability is employed and is related to statistical physics mechanisms.

In the next section, some basic conceptions and most popular descriptions for textures, together with some examples, are presented. Section 4.3 very briefly reviews various statistical approaches to textures, including various statistical features, statistical models, and multichannel filters. More detailed discussions on MRFs and GDs, as well as some simulations of MRFs, are given in Section 4.4. Model parameter estimation plays a very important role in model-based approaches. Section 4.5 outlines the commonly used estimation methods. A brief introduction to various texture processing tasks, especially the segmentation of textured images, is given in Section 4.6, followed by a short summary section.

4.2 Description of Textures

4.2.1 Definitions of textures

Objects and scenes have their characteristic surface textures, which in the corresponding digital images become parts of their distinguishing identities. Texture characterisation and discrimination are important to understanding human visual abilities, and have become more and more important in computer vision for analysing various images, e.g. satellite, microscopic, robotic, and medical images.

However, "*Despite its importance and ubiquity in image data, a formal approach or precise definition of texture does not exist. The texture discrimination techniques are for the most part ad hoc.*" (Haralick and Shapiro 1992). Since textures present so much variety, a precise and rigid universal definition of textures may not be possible. "*Something composed of interwoven elements*" can only be a very broad dictionary description. More detailed descriptions depend on particular applications. The following are some of definitions (compiled by Coggins) cited in Tuceryan and Jain (1993):

"We may regard textures as what constitutes a macroscopic region. Its structure is simply attributed to the repetitive patterns in which elements or primitives are arranged according to a placement rule." (Tamura et al. 1978).

"A region in an image has a constant texture if a set of local statistics or other local properties of the picture function are constants, slowly varying, or approximately periodic." (Sklansky 1978).

"An image texture is described by the number and types of its (tonal) primitives and spatial organisation or layout of its (tonal) primitives." (Haralick 1979).

"Texture is defined for our purposes as an attribute of a field having no components that appear innumerable. The phase relations between the components are thus not apparent. Nor should the field contain an obvious gradient. The intent of this definition is to direct attention of the observer to the global properties of the display — i.e. its overall 'coarseness', 'bumpiness', or 'fineness'. " (Richards and Polit 1974).

"Texture is an apparently paradoxical notion. On the one hand, it is commonly used in the early processing of visual information, especially for practical classification purposes. On the other hand, no one has succeeded in producing a commonly accepted definition of texture." (Zucker and Kant 1981).

"The notion of texture appears to depend upon three ingredients: (i) some local 'order' is repeated over a region which is large in comparison to the order's size, (ii) the order consists in the non random arrangement of elementary parts, and (iii) the parts are roughly uniform entities having approximately the same dimensions everywhere within the texture region" (Hawkins 1969).

Although there is not a universal definition for texture, a number of general and intuitive properties of texture have been summarised by Tuceryan and Jain (1993) as:

* *Texture is a property of area; the texture of a (single) point is undefined. So, texture is a contextual property and its definition must involve grey values in a spatial neighbourhood. The size of this neighbourhood depends upon the texture type, or the size of the primitives defining the texture.*

* *Texture involves the spatial distribution of grey levels. Thus, two-dimensional histograms or co-occurrence matrices are reasonable texture tools.*

* *Texture in an image can be perceived at different scales or levels of resolution. For example, consider the texture represented in a brick wall. At a coarse resolution, the texture is perceived as if formed by the individual bricks in the wall; the interior details of the brick are lost. At a higher resolution, when only few bricks are in the field of view, the perceived texture shows the details of the brick.* (The same argument applies to forests and trees).

* *A region is perceived to have texture when the number of primitive objects in the region is large. If only a few primitive objects are present, then a group of countable objects is perceived instead of a textured image. In other words, a texture is perceived when significant individual "forms" are not readily identifiable.*

In a computer vision context, textures are treated as some form of image pattern in which some regularities (image primitives) and/or irregularities (stochastic grey levels) can be found with some regular and/or irregular placement and size variation rules. They can be expressed as spatial dependence functions, which can be either explicit or implicit, either deterministic or random, functions of image elements or grey levels. Psychophysicists (e.g. Julesz 1962; Marr 1982; Julesz and Bergen 1983) explore the relationships between human visual perception and texture discrimination; while statisticians (e.g. Besag 1974; Kinderman and Snell 1980; Qian and Titterington 1991) model textures as functions of spatial positions. Researchers in signal processing and computer vision, on one side directly apply the discoveries of above two, on the other side expand their traditional methodologies (such as transformations, features, and models) from one dimensional signal processing into two dimensions (e.g. Woods 1972; Ord 1975; Cross and Jain 1983; Kashyap and Chellappa 1983; Geman and Geman 1984; Daugman 1985; Dunn *et al.* 1994).

Some image texture examples of structural, statistical, natural, and artificial are shown in Fig. 4.1 to end this section.

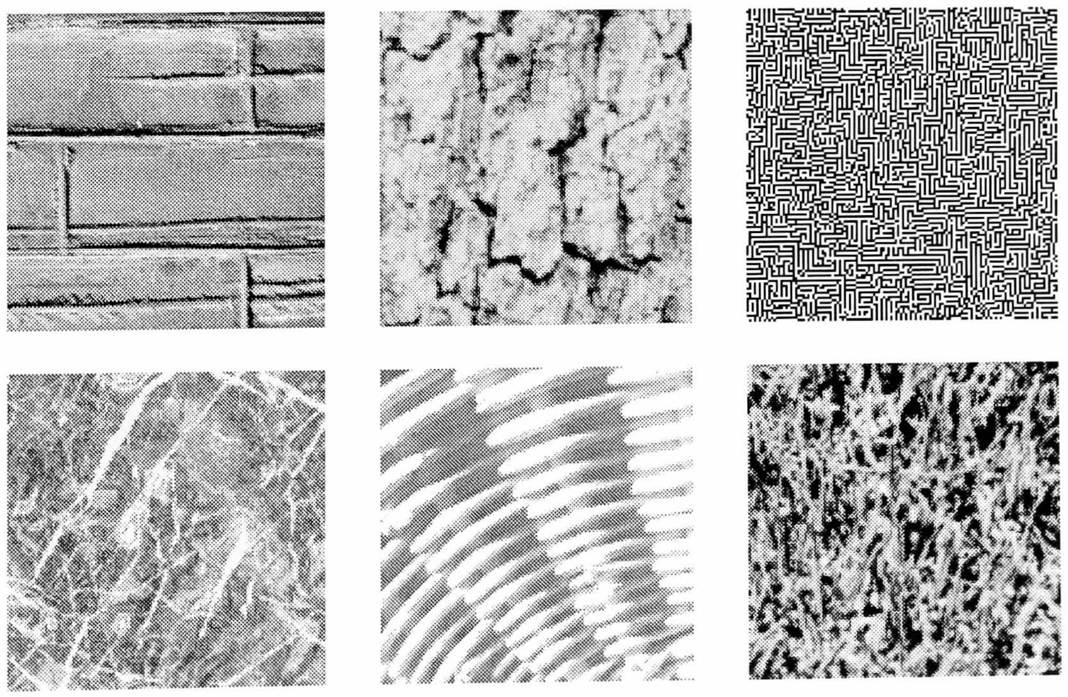


Figure 4.1: Some image textures (8-bit, 128×128).

4.2.2 Human visual perceptions of textures

It has been known that human visual perception of textures has two basic distinct but interactive operations:

(1) ***Preattentive Vision***

"Parallel, instantaneous, without scrutiny, independent of the number of patterns, covering a large visual field, as in texture discrimination" (Julesz and Bergen 1983).

(2) ***Attentive Vision***

"Serial search by focal attention in 50-ms step limited to small aperture, as in form recognition" (Julesz and Bergen 1983).

Most of Julesz's work (Julesz 1962, 1981; Julesz and Bergen 1983) is concerned with preattentive texture perception (mainly for structural textures). He discovered that a few local conspicuous features, which he named *textons*, appear to be the basic units, or image primitives, of preattentive texture perception. In his proposed "theory of textons"; textons are defined as elongated blobs, terminators, and crossings of line segments. Julesz and Bergen (1983) also found that *"preattentive vision directs attentive vision to the locations where differences in the density (number) of textons occur, but ignores the position relationships between textons."*

In most practical image processing applications, the images are likely to contain natural textures, which are statistical rather than structural. The image primitives are too small to be recognised as features or textons, too many to count, and too random in their sizes and locations. Texton definitions for these images are not appropriate. The features in statistical images are hidden, so a proper mathematical method and/or transformation is needed to find the underlying distinguishing features. In the next section, we will review some of the most popular methods regarding statistical textures.

4.3 Approaches to Statistical Textures

Approaches to textured image analysis have progressed through two different stages (Cross and Jain 1983; Haralick 1986; Haralick and Shapiro 1992): feature-based approaches and model-based approaches. Early work in texture analysis sought to discover useful features to characterise the textures and to establish specific measures for discriminating between textures. Later, model-based work seeks a deeper understanding of inter-pixel relationships within a textured image by using stochastic models. On one hand, model-related parameters can be used as useful texture features for recognition or classification. On the other hand, a generative image model has the capability to resemble textured images which provide direct visual matches with observed textures. Very recently, multiresolution, or multiscale, signal processing techniques such as Gabor filters and wavelet transformations or decompositions are receiving considerable attentions and have been applied to texture processing (Tuceryan and Jain 1993). Fig. 4.2 provides a taxonomy of approaches in texture analyses.

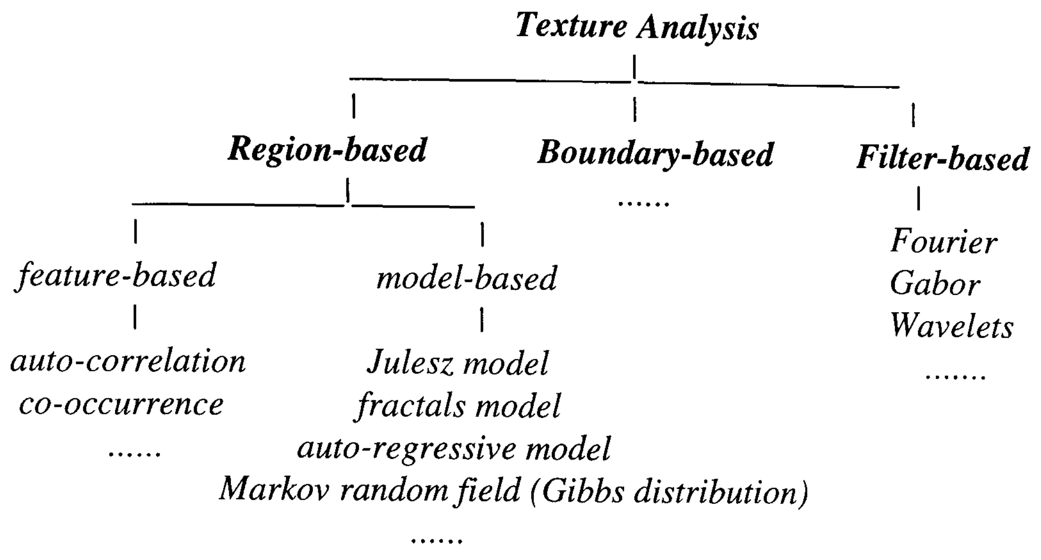


Figure 4.2: General approaches to texture analysis.

4.3.1 Statistical feature based approaches

Image textures have a number of perceived qualities or attributes which give some quantitative measures in describing them, such as: coarseness, roughness, contrast, fineness, smoothness, regularity, directionality, uniformity, and density. In feature-based approaches, the primary goal is to find a set of good features for a texture, which has less parameters but can effectively measure these qualities and thus capture the most prominent characteristics of the texture. The most commonly used texture features are briefly described below. A general review of these measures can be found in Haralick (1979, 1986) and Weszka *et al.* (1976).

(1) ***Autocorrelation***

Let Ω denote a NxN image lattice (we will usually consider a square lattice for simplicity, however results can be easily extended to any other shape of lattice). Let $x(i, j)$ denote the discrete intensity or grey level of the image at position (i, j), then the spatial autocorrelation function is defined by

$$\rho(i,j) \equiv \frac{\dfrac{1}{(N-i)(N-j)}\displaystyle\int_{-\infty}^{\infty}\!\!\int x(u,v)x(u+i,v+j)\,du\,dv}{\dfrac{1}{N^2}\displaystyle\int_{-\infty}^{\infty}\!\!\int x^2(u,v)\,du\,dv}, \quad 0\le i \le N,\ 0\le j \le N \qquad (4.1)$$

This function measures the spatial autocorrelation of image pixels and primitives. It describes the size of the tonal primitives. For an image with tonal primitives of large size, its autocorrelation will extend further; while an image with tonal primitives of a small size, its autocorrelation will fall quickly with distance. The autocorrelation function may have periodic properties if the primitives of the texture are spatially periodic.

(2) *Digital Transforms*

The most commonly used transform is the Fourier transform, defined by

$$F(u,v) = \int \int_{-\infty}^{\infty} x(i,j)\exp(-2\pi\sqrt{-1}(ui+vj))\,di\,dj, \qquad -\infty < u,\ v < \infty \tag{4.2}$$

where u, v are the *spatial frequencies.*

The Fourier transform directly measures the spatial frequency components of the textured image. Fine textures are rich in high spatial frequencies, while coarse images are restricted to the low frequencies.

Other transforms include Walsh-Hadamard transform, Slant function, Karhunen-Loeve expansion (see Haralick 1986). It is reported (cited in Haralick 1979) that there is no significant difference in classification accuracy between these transform functions.

(3) *Texture Gradient*

Another method to measure the spatial frequency of textures is to calculate the number of edges (or sharp intensity changes) per unit area, that is "*texture edginess*". Fine textures have more edges per unit area, while coarse textures have less.

An extended edginess measure is the gradient function over distance, which is defined as

$$g(d) \equiv \sum_{(i,j)\in\eta} \{|x(i,j)-x(i+d,j)| + |x(i,j)-x(i-d,j)| + |x(i,j)-x(i,j+d)| + |x(i,j)-x(i,j-d)|\} \tag{4.3}$$

where $d(\cdot)$ denotes the distance, and η denotes the texture window over which the gradient function is defined.

(4) *Spatial Grey-Tone Co-Occurrence Matrix*

Spatial grey-tone (grey-level) co-occurrence matrix is another commonly used texture measure. It measures the relative frequencies or probabilities of transition from one grey level to another at defined spatial distances. Some texture features can be extracted from this matrix. Haralick (1986) and Weszka *et al.* (1976) give a comprehensive review on this issue. Such studies have achieved reasonable results for different textures (Haralick 1979, 1986).

Usually the co-occurrence probability matrix is defined at a fixed distance d and a fixed angle ϕ,

$$P(i,j|d,\phi) = \#\{(k,l),(m,n)\in L, d[(k,l),(m,n)] = d, \phi[(k,l),(m,n)] = \phi, x(k,l) = i, x(m,n) = j\} \tag{4.4}$$

where $d(.)$ is a distance measure, and $\phi(.)$ is an angle measure, i, $j\in$ {grey levels}, # denotes the number of pixels in the set.

From this matrix, the following texture features can be calculated:

$$\text{Energy: } \sum_i \sum_j P(i,j)^2 \tag{4.5a}$$

$$\text{Entropy: } \sum_i \sum_j P(i,j) \log P(i,j) \tag{4.5b}$$

$$\text{Contrast: } \sum_i \sum_j (i-j)^2 P(i,j) \tag{4.5c}$$

$$\text{Correlation: } \sum_i \sum_j (i-u_x)(j-u_y)P(i,j) / \sigma_x \sigma_y \tag{4.5d}$$

etc.

where u_x, u_y are means of the matrix along row and column respectively, and σ_x, σ_y are the corresponding standard variances.

A generalised grey level spatial dependence has been proposed by Haralick and Shapiro (1992) to extend the measures for the relationships of pixel pairs to more generalised spatial neighbourhoods.

(5) ***Other Features***

There are many other traditional texture features such as *grey-level run-length* statistics, *grey-level difference* statistics (see Weszka *et al.* 1976). The *Gabor function* has been used for extracting texture features for discrimination (Fogel and Sagi 1989; Turner 1986). Shang and Brown (1992) have used *interframe principal component* features in texture classification.

4.3.2 Texture model based approaches

In model-based texture approaches, the primary goal is to seek a mathematical expression, or model, which can efficiently and effectively describe the inter-relationship of the pixel grey levels in an image. These models define the stochastic configuration of pixels on the image lattice. They are always reproductive, and can be used to simulate the textures under study, and thus provide a direct visual comparison between real textures and synthetic ones. The most popular models are:

(1) ***Julesz Model***

To study the visual discrimination on statistical textures, Julesz carried out a series of experiments in which a stochastic model, known as the *Julesz model*, was used to generate textures with different statistical orders (Julesz 1962, 1975, 1981). Julesz first conjectured that human visual texture discrimination is based on the second order statistics of textures. Later, he modified the hypothesis to "*the pre-attentive textural system cannot globally compute third- or higher-order statistics*" (Julesz 1981). The Julesz model is applied mostly for texture image analysis in 1-D on a row by row basis. In a $N \times N$ image, a pixel sequence $\{x(i,1), x(i,2), \ldots, x(i,k), \ldots x(i,N)\}$ is formed from right to left along each row. If the grey tone is quantised to L levels, then $0 \leq x(i,j) \leq L-1$, and X and Y are two random variables taken from 1 to L-1.

For the first-order process, each pixel is an independent random variable

$$P[X(i,j)=m]=P(m) \tag{4.6}$$

For the second-order process, in addition to the first order probability of (4.6), the second-order statistical property is formed by using the following procedure (Pratt *et al.* 1981):

Set $X(i,1)=X$, i.e. $x_{i,1}$, select $X(i,j)=(x_{i,j-1}+Y) \bmod L$, then the transition probability is

$$P(x_{i,j}|x_{i,j-1})=P[X(i,j)=(x_{i,j-1}+Y)=n \bmod L]=P(n) \tag{4.7}$$

In a similar way, the third order probability is described by

$$P(x_{i,j}|x_{i,j-1},x_{i,j-2})=P[X(i,j)=(x_{ij-1}+x_{ij-2}+Y)=q \bmod L]=P(q) \tag{4.8}$$

Two dimensional cases can be generated by repeating the above procedures on a column basis as well. Gagalowicz (1981) proposed a method to synthesise a stochastic process from its *a priori* second-order statistics.

(2) ***Simultaneous Autoregressive (SAR) Model***

The autoregressive model has been used successfully in 1-D time series analysis, where it models the underlying stochastic process of the series by a linear combination of the past states and random noise. Such a model can be extended to 2-D cases. Denoting $X(i,j)$, or X_{ij}, as a random variable, $x(i,j)$, or x_{ij}, is a realisation of $X(i,j)$. $\mathbf{x}$ is a realisation of the random process on a $N\times N$ lattice Ω, $\mathbf{x}=(x_{11}, x_{12}, \ldots x_{NN})$. For simplicity, we can assume $\mathbf{x}$ has zero mean (otherwise $\mathbf{x}$ can be subtracted from the mean value).

For an infinite lattice, the *autoregressive* (AR) model can be represented by

$$X(i,j)=\sum_{(u,v)\in\eta}\theta_{u,v}X(i+u,j+v)+\varepsilon(i,j) \tag{4.9}$$

where $\{\theta_{u,v}, (u,v)\in\eta\}$ are the *minimum mean-squared-error* (MMSE) model parameters, u and v are incremental coordinates, η is a neighbourhood set, $\{\varepsilon(i,j)\}$ is the *independent identical distributed* noise sequence with zero-mean and variance, σ^2.

Ordinary AR models are causal, which means that the neighbour set only consists of pixels before site (i,j) in the raster scan order. When the neighbour set consists of the pixels of both sides, i.e. before and after the pixel (i,j), the model is termed a *simultaneous autoregressive* model (SAR). Although the neighbourhood needs not to be symmetrical, in most cases it is assumed to be so, and parameters are also assumed symmetrical (i.e. $\theta_{u,v}=\theta_{-u,-v}$). Otherwise parameters may not be identifiable (Besag 1974; Cross and Jain 1983). A commonly used hierarchically-ordered neighbourhood structure is shown in Fig. 4.3. The numbers indicate the relative order of the neighbourhood system. For example, the first order neighbourhood set, i.e. the nearest-neighbourhood, is {(0,-1); (0,1); (-1,0); (1,0)}, the second order neighbourhood set is {(-1,-1); (0,-1); (1,-1); (-1,0); (1,0); (-1,1); (0,1); (1,1)}, referring the pixel (i,j). However, there is no restriction on the definition of the neighbourhood structure.

5	4	3	4	5
4	2	1	2	4
3	1	x	1	3
4	2	1	2	4
5	4	3	4	5

Figure 4.3: Neighbourhood structures (Numbers indicate the relative orders of the neighbourhood of x).

For a finite lattice Ω, a toroidal structure can be used to compensate the boundaries where usually only half neighbourhood pixels are available. The toroidal SAR model can be written as

$$X(i,j) = \sum_{(u,v)\in\eta} \theta_{uv} X(i \oplus u, j \oplus v) + \varepsilon(i,j) \tag{4.10}$$

where $\oplus$ is the modulo operator.

If the noise sequence is assumed to be Gaussian distributed with zero-mean and variance σ^2, then the process has a joint probability density function (Besag 1974):

$$p(\mathbf{x}) = (2\pi\sigma^2)^{-M/2} |\mathbf{B}| \exp\{-\frac{1}{2\sigma^2}\mathbf{x}^T\mathbf{B}^T\mathbf{B}\mathbf{x}\} \tag{4.11}$$

where $M=N\times N$, and $\mathbf{B}$ is a $M\times M$ matrix with diagonal elements of unity and off-diagonal element of $-\ \theta_{uv}$. A sufficient condition on $\{\theta_{uv}\}$ to ensure stationary of $X(.)$ is given by Kashyap and Chellappa (1983), as

$$1 - \sum_{(u,v)\in\eta} \theta_{uv} z_1^u z_2^v \neq 0, \qquad \forall z_1, z_2;\ |z_1| = 1,\ |z_2| = 1 \tag{4.12}$$

Given a finite lattice image, the SAR model-based parameters can be estimated by the *least square* (LS) or the *maximum likelihood* (ML) methods. LS estimation is simple, but not consistent for non-unilateral neighbour sets (Besag 1974; Kashyap and Chellappa 1983). ML estimation yields consistent and efficient estimates, but is very complicated even for the Gaussian case. Kashyap and Chellappa (1983) used an approximate expression for the log likelihood function and have proposed an iterative method which yields an estimate close to the ML estimate with a faster convergence speed. More details on model parameter estimation will be addressed in Section 4.4.

For a *rotation invariant simultaneous autoregressive* model (RISAR) (Kashyap 1986), averaging around a circle can be used to de-orientate the directionality of textures.

(3) *Markov Random Field (MRF) Model*

The SAR model is a subclass of the MRF model. For every SAR model, there exists a unique MRF model. The MRF model is an extension of Markov chain models for 1-D series to 2-D fields, and is a spatial interactive statistical model to describe interrelationships between point variables in a 2-D statistical field. The MRF model, and its equivalent Gibbs distribution (GD) model will be reviewed in details in Section 4.3

(4) ***Other Models***

There are some other texture models, e.g. mosaic model, fractal model, etc. for some specific application purposes. For more details, see Kashyap (1986), Haralick (1986), and Cross and Jain (1983).

4.3.3 Multichannel filter based approaches

In addition to MRF or GD model-based approaches, multichannel filtering methods have become popular in recent years, because of their multiscale, or multiresolution nature. Both approaches have achieved very impressive results. Using MRF models, statisticians and mathematicians attempt to seek a deeper understanding of spatial relationships between image pixels by fitting them to mathematical models. The neurobiologists and psychophysicists have begun to understand the human visual perceptive principles in texture discrimination based on human retinal multichannel localised spatial filtering receptors. Tuceryan and Jain (1993) termed this latter approach as "*signal processing methods*".

The Gabor filter (1942, cited in Daugman 1985) is a Gaussian window, and the corresponding transform is a short-time Fourier transform windowed by such a function. It was first used to optimise both time and frequency resolutions, i.e. to reach the low bound of time-frequency resolution. Daugman (1985) extended this one dimensional optimal filter to two dimensional (i.e. spatial and spatial-frequency domain) to mimic the receptive fields of the human visual system (HVS). There is considerable evidence from neurobiological and psychophysical studies showing that the HVS is performing some form of local spatial-frequency filtering of the retinal image using a bank of filters pre-tuned to different spatial-frequency bands and on different spatial scales. A typical 2-D Gabor function (Daugman 1985) can be defined as

$$\varphi(i,j) = \exp\{-[\frac{(i-i_0)^2}{2\sigma_i^2} + \frac{(j-j_0)^2}{2\sigma_i^2}]\} \times \exp\{-2\pi\sqrt{-1}[u_0(i-i_0) + v_0(j-j_0)]\} \quad (4.13)$$

where (i,j) is the spatial co-ordinate, (u,v) represents spatial-frequency co-ordinates, and the (σ_i,σ_j) represents the spatial bandwidth. The 2-D Gabor function $\varphi(x,y)$ is a product of an elliptical Gaussian with an aspect ratio $\lambda=\sigma_i/\sigma_j$ whose centroid is located at (i_0, j_0) and a complex exponential modulation with spatial-frequency $\sqrt{u_0^2 + v_0^2}$ and orientation $\theta=\arctan(v_0/u_0)$. The Fourier transform of the 2-D Gabor function can be written as

$$\Phi(u,v) = \exp\{-2\pi[\sigma_i^2(u-u_0)^2 + \sigma_j^2(v-v_0)^2]\} \times \exp\{-2\pi\sqrt{-1}[i_0(u-u_0) + j_0(v-v_0)]\} \quad (4.14)$$

The *Heisenberg uncertainty inequality* defines the time and frequency and spatial and spatial-frequency resolution limits for any filter as follows

$$\text{1-D:} \qquad \Delta t \Delta f \geq \frac{1}{4\pi} \tag{4.15a}$$

$$\text{2-D:} \qquad \Delta i \Delta u \geq \frac{1}{4\pi}; \;\; \Delta j \Delta v \geq \frac{1}{4\pi}; \;\; \Delta i \Delta j \Delta u \Delta v \geq \frac{1}{16\pi^2} \tag{4.15b}$$

where Δt and Δf are the 1-D time and frequency effective widths of a 1-D filter. The effective width is defined as the square root of the variance of the energy distribution of the filter. Δi, Δj, Δu, and Δv are the 2-D spatial and spatial-frequency effective widths for a 2-D filter. The Gabor functions can achieve the lowest resolution limit with the effective widths in both domains of

$$\Delta i = \frac{\sqrt{2}\sigma_i}{2}; \;\; \Delta j = \frac{\sqrt{2}\sigma_j}{2}; \;\; \Delta u = \frac{\sqrt{2}}{4\pi\sigma_i}; \;\; \Delta v = \frac{\sqrt{2}}{4\pi\sigma_j} \tag{4.16}$$

Many applications have adopted a bank of Gabor filters (with different locations and orientations), and used the outputs of these filters as features for texture processing (Tuner 1986; Bovik *et al.* 1990; Jain and Farrokhnia 1991; Dunn *et al.* 1994; Guerin-Dugue and Palagi 1994; Tan 1995)

Using a Gabor filter bank is an example of multiresolution signal processing, and more recently the field of wavelet theory (wavelet representation, decomposition, and transformation) has developed (Mallat 1989). By varying the scale of the window function, the wavelet transform provides varying frequency (or spatial-frequency) resolutions with varying time (or spatial) resolutions (Mallat 1989; Rioul and Vetterli 1991; Manjunath and Chellappa 1993; Chen and Kundu 1994). Although orthogonal decomposition is the most desirable, it is difficult to achieve. The non-orthogonal basis functions, such as Gabor functions, are the most popular basis functions used in practical applications. For example, Manjunath and Chellappa (1993) have used the 2-D Gabor function as a basis function (basic wavelet). The signal can be decomposed in terms of such functions with different scaling, usually α^q, (α>0, q=1, 2, 3 ...), to the different aspect ratios λ, and different orientations in [0, π]. The corresponding Gabor wavelet transform can be expressed as (Manjunath and Chellappa 1993)

$$W_q(i_0, j_o, \alpha, \lambda, \theta) = \iint x(i,j)\varphi^*[\alpha^q(i-i_0, j-j_0), \lambda, \theta]\,didj \tag{4.17}$$

4.4 Markov Random Field and Gibbs Distribution

4.4.1 Definition of Markov random fields (MRFs)

Besag (1974) and Cross and Jain (1983) have given a clear definition of MRFs. Assuming that a random field $\mathbf{X}$ is defined on a lattice Ω of NxN, then $\mathbf{X}$ is a matrix with each element is a random variable. Assuming $\mathbf{x}$ is one possible realisation of the $\mathbf{X}$ and denote Λ as the set of all possible realisations of $\mathbf{X}$ on Ω.

Definition: A MRF is a random field X on a lattice Ω with:

(1) Positivity: *Any realisation on the lattice is possible, i.e.*

$$P(\mathbf{x}) > 0, \text{ for all } \mathbf{x} \in \Lambda \tag{4.18a}$$

(2) Markovianity: *The conditional probability of any site on the others only depends on its neighbouring sites, i.e.*

$$P[x_{ij}|x_{mn}, (m,n)\neq(i,j), (m,n)\in\Omega]=P[x_{ij}|x_{mn}, (m,n)\neq(i,j), (m,n)\in\eta_{ij}] \tag{4.18b}$$

(3). Homogeneity: *The conditional probability $P(x_{ij}\,|\,\eta_{ij})$ depends only on the configuration of neighbours and is translation invariant, i.e. does not depend on the location.*

where η_{ij} denotes the neighbourhood set of pixels around (i,j), excluding (i,j) itself. Neighbourhood structures are defined as in the SAR model.

4.4.2 Expression of Markov random fields

Woods (1972) has shown that in the homogeneous Gaussian case, the MRF can be expressed by a set of difference equations, which is also called the *conditional Markov* model (CM) by Kashyap and Chellappa (1983).

In a finite lattice, a Gaussian MRF (GMRF) can be represented by

$$X(i,j) = \sum_{(u,v)\in\eta} \theta_{uv} X(i \oplus u, j \oplus v) + e(i,j) \tag{4.19}$$

where the neighbour set is symmetrical and $\theta_{uv}=\theta_{-uv}$ for all $(u,v)\in\eta$, $\{e(i,j)\}$ is a stationary Gaussian noise sequence and is characterised by

$$\mathrm{E}\{e(i,j)|\text{all } x(m,n), (m,n)\neq(i,j)\}=0 \tag{4.20a}$$

$$\mathrm{E}\{e(i,j)\}=0 \tag{4.20b}$$

$$\mathrm{E}\{e^2(i,j)\}=\rho \tag{4.20c}$$

So the noise sequence is correlated as shown below

$$\mathrm{E}\{e(i,j)e(i+u,j+v)\} = \begin{cases} \rho & u,v=0 \\ -\theta_{uv}\rho & (u,v)\in\eta \\ 0 & \text{otherwise} \end{cases} \tag{4.20d}$$

Parameters $\{\theta_{uv}\}$ are the MMSE coefficients for forming $X(i,j)$ by using its neighbours. The noise $\{e(i,j)\}$ is the error and is not white. However if $\{e(i,j)\}$ is homogeneous and Gaussian, the $\{X(i,j), (i,j)\in\Omega\}$ is also homogeneous, Gaussian, and Markovian (Wood 1972, and Kashyap and Chellappa 1983 (Geman and Geman 1984).

The conditional probability density function of the GMRF is

$$p[x(i,j)|x(i+u,j+v),(u,v)\in\eta]=\frac{1}{(2\pi\rho)^{1/2}}\exp\{-\frac{1}{2\rho}[x(i,j)-\sum_{(u,v)\in\eta}\theta_{uv}x(i,j)]^2\} \quad (4.21)$$

The sufficient condition for stationary of $\{X(i,j)\}$ is

$$\mu_{ij}\equiv(1-2\Theta^T\Phi_{ij})>0, \qquad \text{for all } (i,j)\in\Omega \quad (4.22)$$

where $\Theta=col[\theta_{uv},\ (u,v)\in\eta']$, $\Phi_{ij}=col\{\cos[(2\pi/M)(iu+jv)],\ (u,v)\in\eta'\}$, η' is the asymmetric half of η.

The joint distribution for a zero-mean GMRF on the lattice Ω is (Besag 1974)

$$p(\mathbf{x})=(2\pi\rho)^{-M/2}|\mathbf{B}|^{1/2}\exp\{-\frac{1}{2\rho}\mathbf{x}^T\mathbf{B}\mathbf{x}\} \quad (4.23)$$

where the $\mathbf{B}$ is defined as for the SAR model.

4.4.3 Simulations of Markov random fields

There are several algorithms to generate MRFs (e.g. Fourier transform methods: Kashyap and Chellappa 1983, Khotanzad and Kashyap 1987; Monte Carlo methods: Cross and Jain 1983). Here we describe the algorithm proposed by Cross and Jain (1983):

(0) Choose a initial realisation $\mathbf{x}$ with equal grey level probability at each site of the lattice Ω,

(1) Randomly choose two pixels, $x(i,j)$ and $x(m,n)$, and swap them to obtain a new realisation $\mathbf{y}$. Re-index $\mathbf{x}$ and $\mathbf{y}$ to vectors: $\mathbf{x}=[x(1), x(2), \ldots x(M)]$ and $\mathbf{y}=[y(1), y(2), \ldots y(M)]$, where M is the total number of sites in Ω.

(2) Calculate the probability ratio $r=p(\mathbf{y})/p(\mathbf{x})$ from the conditional distribution (Besag 1974, Cross and Jain 1983),

$$\frac{p(\mathbf{y})}{p(\mathbf{x})}=\prod_{k=1}^{M}\frac{p[x(k)|x(1),x(2),\ldots x(k-1),y(k+1),\ldots y(M)]}{p[y(k)|x(1),x(2),\ldots x(k-1),y(k+1),\ldots y(M)]} \quad (4.24)$$

where $p[x(k)|\eta]=\dfrac{\exp[x(k)T(\eta)]}{1+\exp[x(k)T(\eta)]}$, or $p[x(k)|\eta]=\dfrac{\exp[x(k)T(\eta)]}{\sum_{s\in G}\exp[sT(\eta)]}$, for binary or grey-level images respectively, and $T(\eta)$ is a function of the neighbourhood and G is the grey-level range.

(3) If $r\geq 1$, accept the new realisation, if $r<1$, accept the new realisation with probability r.

The following figure shows some simulation results of second order Markov random fields by using this algorithm.

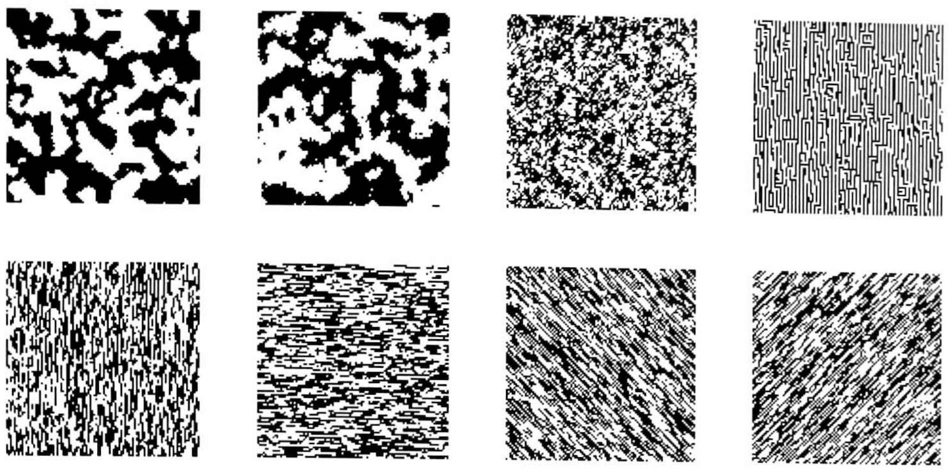


Figure 4.4: Markov random field simulations: Binary case (100×100 in size).

4.4.4 Gibbs distributions (GDs)

MRFs were not widely used until the Hammerslay-Clifford theorem (see Besag 1974), Spitzer' work (1971), and the work of Hassner and Sklansky (cited in Cross and Jain 1983; Geman and Geman 1984) became better known. They have proved the equivalence between the MRF and Gibbs distribution (GD). That is, every MRF can be measured by the Gibbs distribution, and the Gibbs distribution is Markovian. An important property of the MRF is its local characteristic, while an important property of the GD its explicit probabilistic measure. The GD provides the MRF with an explicit form of joint distribution (not only for the Gaussian case).

The random field concept came from some of Ising's work on statistical and physical characteristics of ferromagnetic materials, is known as the Ising model, and was used to measure the probabilistic configurations (Kinderman and Snell 1980). The measure which Ising defined is the Gibbs measure or Gibbs distribution

$$p(\mathbf{x}) = \frac{1}{Z}\exp\{-\frac{1}{k_B T}U(\mathbf{x})\} \tag{4.25a}$$

where $U(\mathbf{x})$ is the energy function of configuration $\mathbf{x}$, T is the temperature, k_B is a universal constant (e.g. Boltzmann constant), and Z is a normalising constant defined by

$$Z = \sum_{\mathbf{x}\in\Lambda}\exp\{-\frac{1}{k_B T}U(\mathbf{x})\} \tag{4.25b}$$

which is also called the partition function.

To complete the Gibbs distribution definition, we need to define the clique system. In a lattice with a neighbourhood system, (Ω, η), the cliques are defined as follows:

Definition of cliques: A clique of image lattice (Ω, η), denoted by c, is a subset of Ω such that:

(1) c consists of a single pixel;

(2) c consists of a set of pixels, which are neighbours of each other.

The collection of all cliques of (Ω, η) is denoted by $\mathcal{C}$.

For the first order and second order neighbourhood systems, corresponding clique types are shown in Fig. 4.5.

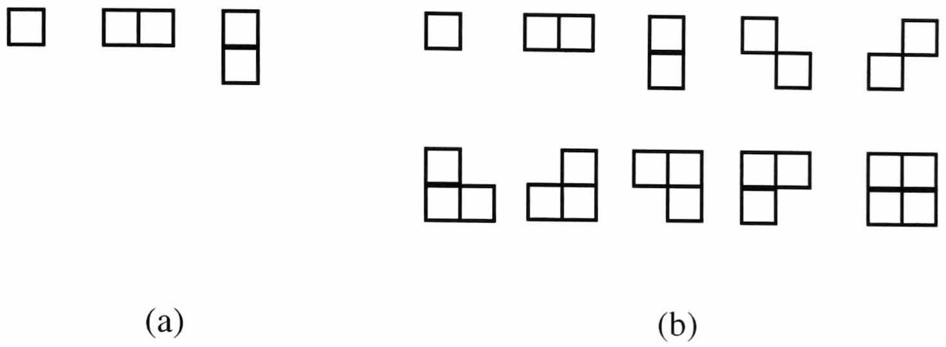


Figure 4.5: Clique systems. (a) First order; (b) Second Order.

Definition of Gibbs distribution: For a neighbourhood system, η, defined over a finite lattice Ω, a random field $\mathbf{X}=\{X(i,j), (i,j)\in\Omega\}$, has the Gibbs distribution, i.e. it is a Gibbs random field if and only if its joint distribution has the form of (4.25a) with the partition function of (4.25b) and the energy function defined as

$$U(\mathbf{x})=\sum_{c\in\mathcal{C}} V_c(\mathbf{x}). \tag{4.26}$$

where $V_c(\mathbf{x})$ is a potential associated on (L, η) and depends only on those pixels $x(i,j)$ of $\mathbf{x}$ for which $(i,j)\in c$.

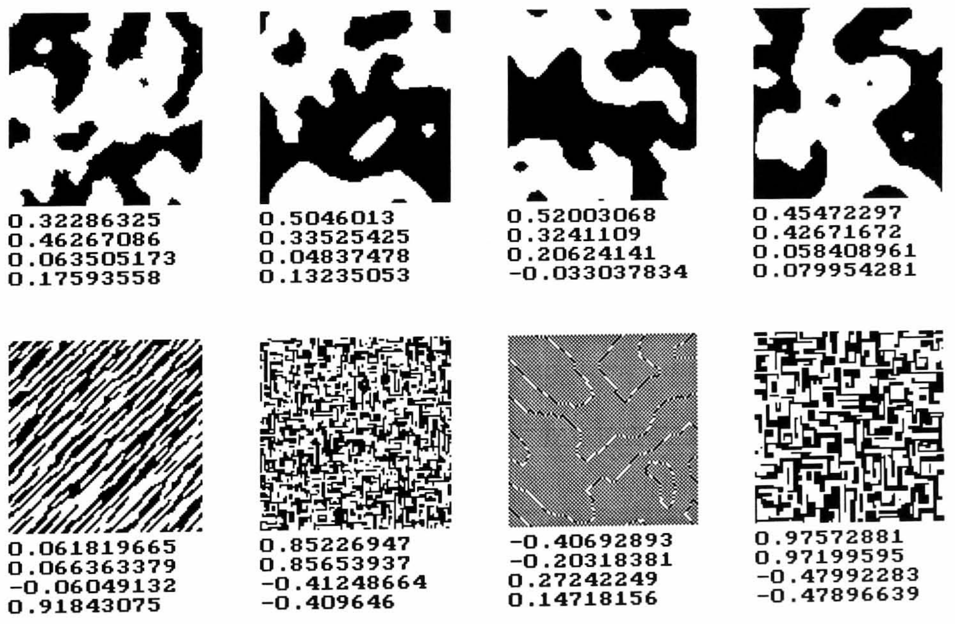


Figure 4.6: Simulations of Gibbs random fields (Binary case, 100×100 in size).

GDs have been intensively employed in MRF model-based image processing. Geman and Geman (1984) developed a stochastic relaxation algorithm for image restoration. They also proposed a method, called the *Gibbs sampler*, for generating Gibbs random fields. Derin and Elliott (1987) used GDs to model and segment noisy images and textured images. Fig. 4.6 shows some simulations of binary GRFs by using the Gibbs sampler algorithm.

4.5 Model Parameter Estimation

In model-based texture approaches, model parameters are used as features for classifying different textures and to construct discriminable energy functions. Model parameter estimation plays an very important role in model-based texture analysis whether the analysis is for classification or segmentation, either in supervised or in unsupervised cases. The accuracy, effectiveness, consistence, and computational efficiency of the estimate are the important practical criteria. The followings are commonly used estimation methods:

4.5.1 Least square (LS) estimation

(1) ***SAR Model***

Given an image on a finite lattice Ω, its SAR model parameters can be estimated by applying the LS method to the model equation (4.9) over the entire lattice, which results in

$$\hat{\Theta} = [\sum_{(i,j)\in\Omega} \mathbf{z}(i,j)\mathbf{z}^T(i,j)]^{-1}[\sum_{(i,j)\in\Omega} \mathbf{z}(i,j)x(i,j)], \tag{4.27a}$$

$$\hat{\sigma}^2 = \frac{1}{M}\sum_{(i,j)\in\Omega}[x(i,j) - \hat{\Theta}^T\mathbf{z}(i,j)]^2, \tag{4.27b}$$

$$\mathbf{z}(i,j) = col[x(i+u, j+v), (u,v)\in\eta], \tag{4.27c}$$

where *col* means column.

The LS estimate is very simple method, but is not consistent for non-unilateral neighbour sets even when M tends to infinity (Kashyap and Chellappa 1983). However, because of its low computational cost, LS estimates are still widely used either as an initial estimate for other iterative estimate methods, or as an estimate for segmenting purposes.

(2) ***CM Model***

For the CM model, or GMRF model, it is also possible to use the LS method to estimate its model parameters in Eqn. (4.19). Kashyap and Chellappa (1983) have modified the above LS estimate by using only interior pixels, and have proved that it yields an asymptotically consistent estimation:

$$\hat{\Theta}=[\sum_{(i,j)\in\Omega_I}\mathbf{q}(i,j)\mathbf{q}^T(i,j)]^{-1}(\sum_{(i,j)\in\Omega_I}\mathbf{q}(i,j)x(i,j)) \tag{4.28a}$$

$$\hat{\rho}=\frac{1}{M}\sum_{(i,j)\in\Omega_I}(x(i,j)-\hat{\Theta}^T\mathbf{q}(i,j))^2 \tag{4.28b}$$

where Ω_I is called the interior set, and Ω_I=Ω–Ω_B, where Ω_B is called the boundary set, $\Omega_B=\{(i,j)\in\Omega$, and $(i+u,j+v)\notin\Omega$, for at least one $(i,j)\in\eta\}$, and

$$\mathbf{q}(i,j)=col[x(i+u,j+v)+x(i-u,j-v),(u,v)\in\eta] \tag{4.28c}$$

(3) *GD (MRF) Model*

The difficulty in estimating GD parameters is in the calculation of the partition function, which is an integration over *all* possible realisations of the random field under one set of parameters. Exact derivation of this function is almost impossible. Derin and Elliott (1987) proposed a parameter estimate method, termed the *histogramming method* by Gurelli and Onural (1994), which avoids the calculation of the partition function. We regard this method as a LS method. The method models each pixel's energy function as a joint co-occurrence probability of the pixel and its neighbours, which can be estimated using histograms. This process can form a series of difference equations. Then an estimate for the model parameter can be made by using LS methods to these equations. The method can be stated as follows:

Denote $\{x(i+u,j+v);(u,v)\in\eta\}$ as pixels in the neighbourhood, η, of the pixel $x(i,j)$. Then we have

$$\frac{p[x(i,j),x(i+u,j+v);(u,v)\in\eta]}{p[x(i+u,j+v);(u,v)\in\eta]}=p[x(i,j)|\eta]=\frac{e^{-U(x(i,j)|\eta,\Theta)}}{Z[x(i,j)|\eta,\Theta]} \tag{4.29a}$$

where

$$Z[x(i,j)|\eta,\Theta]=\sum_{x(i,j)\in Q}e^{-U[x(i,j)|\eta,\Theta]} \tag{4.29b}$$

where Q is the state space for each pixel, i.e. the grey levels of the image.

Rearranging these two equations yield

$$\frac{e^{-U[x(i,j)|\eta,\Theta)}}{p[x(i,j),x(i+u,j+v);(u,v)\in\eta]}=\frac{Z[x(i,j)|\eta,\Theta]}{p[x(i+u,j+v);(u,v)\in\eta]} \tag{4.30}$$

Note that the right hand side (RHS) of this equation does not depend on the pixel $x(i,j)$, so should the left hand side (LHS). Therefore, for any two distinct pixels, e.g. $x(i_1,j_1)$, $x(i_2,j_2)$, we have

$$e^{-U[x(i_1,j_1)|\eta,\Theta)]+U[x(i_2,j_2)|\eta,\Theta]}=\frac{p[x(i_1,j_1),x(i_1+u,j_1+v);(u,v)\in\eta]}{p[x(i_2,j_2),x(i_2+u,j_2+v);(u,v)\in\eta]}\equiv\frac{p_1}{p_2} \tag{4.31}$$

where p_1 and p_2 are the joint probabilities for $x(i_1,j_1)$ and $x(i_2,j_2)$ with their own neighbourhoods respectively. The energy function is usually defined as

$$U[x(i,j)|\eta,\Theta] = -\sum_{(u,v)\in\eta}\theta_{uv}\phi[x(i,j),x(i+u,j+v)] \tag{4.32}$$

Taking the logarithm of Eqn. (4.31), we have a difference equation for the pixel pair $x(i_1,j_1)$ and $x(i_2,j_2)$

$$\sum_{(u,v)\in\eta}\theta_{uv}\{\phi[x(i_2,j_2),x(i_2+u,j_2+v)]-\phi[x(i_1,j_1),x(i_1+u,j_1+v)]\} = \ln(\frac{p_1}{p_2}) \tag{4.33}$$

The value of RHS of (4.33) can be estimated using a histogram method. Taking as many as possible of distinct pairs from the image, the parameters Θ can be estimated using the LS method.

This method works well provided that a large amount of image data is available, however for limited training data or unsupervised case, the histogram estimates for joint probabilities will be far from accurate. Gurelli and Onural (1994) have improved the performance of this method for the case of a small amount of image data.

4.5.2 Maximum likelihood (ML) estimation

The ML estimate is an optimal Baysian estimate for model parameters. It can yield a consistent and efficient estimate. However in many cases, even the log likelihood function can only be derived under some assumptions about distribution of the process, such as Gaussian. Maximising the likelihood function is not an easy task. Numerical optimisation methods, such as the Newton-Raphson approach, have to be used to obtain ML estimates. Generally, ML estimates are computationally very costly.

(1) ***SAR Model***

For the SAR model, the joint distribution for an infinite lattice, i.e. (4.9), is extremely difficult to derive, but is available, i.e. (4.11), for a finite toroidal lattice SAR model. Kashyap and Chellappa (1983) have proposed an approximation to the ML estimate for the finite toroidal lattice SAR model in Gaussian noise case. Their method is an iterative scheme

$$\Theta_{t+1} = (R-\frac{1}{\rho_t}S)^{-1}(V-\frac{1}{\rho_t}U), \quad t=0,1,2,... \tag{4.34a}$$

$$\sigma_t^2 = \frac{1}{M}\sum_{(i,j)\in\Omega}(x(i,j)-\Theta_t^T\mathbf{z}(s))^2, \qquad t=0,1,2,... \tag{4.34b}$$

where

$$S=\sum_{(i,j)\in\Omega}\mathbf{z}(i,j)\mathbf{z}^T(i,j); \quad U=\sum_{(i,j)\in\Omega}\mathbf{z}(i,j)x(i,j), \quad V=\sum_{(i,j)\in\Omega}C_{ij}, \quad R=\sum_{(i,j)\in\Omega}(S_{ij}S_{ij}^T-C_{ij}C_{ij}^T)$$

$$C_{ij}=col[\cos\frac{2\pi}{N}(iu+jv),(u,v)\in\eta], \; S_{ij}=col[\sin\frac{2\pi}{N}(iu+jv),(u,v)\in h]$$

The initial value, Θ_0, is chosen as $\Theta_0=S^{-1}U$, i.e. the LS estimate. Their experimental work has shown that this scheme yields an estimate close to the ML estimate and also yields a good estimate for the non-Gaussian variable case.

(2) *CM Model*

The joint distribution for CM model is expressed by (4.23), then the likelihood function is

$$\ln p(\mathbf{x} / \Theta, \rho) = \ln |\mathbf{B}|^{1/2} - \frac{M}{2} \ln 2\pi\rho - \frac{1}{2\rho} \mathbf{x}^T \mathbf{B} \mathbf{x} \tag{4.35a}$$

where

$$|\mathbf{B}| = \prod_{(i,j)\in\Omega} (1 - \Theta^T \Psi_{ij}) \tag{4.35b}$$

$$\Psi_{ij} = col\{\exp[\sqrt{-1}\frac{2\pi}{M}(iu + jv)], (u,v) \in \eta\} \tag{4.35c}$$

For the Markov model, it appears that the ML estimate of model parameters can only be obtained by applying numerical optimising methods in maximising the above likelihood function (Cross and Jain 1983; Kashyap and Chellappa 1983).

(3) *Coding Method*

Coding methods for model parameter estimation were introduced by Besag (1972, cited in Besag 1974), initially for binary images and later for other cases.

In the Markov model, the noise term has a correlative nature over a neighbourhood. Every site's conditional likelihood can not be simply summed to form the joint likelihood function for the whole lattice because of the dependence of neighbouring pixels. Only with the assumption of Gaussian structure for the noise sequence and toroidal lattice structure, does the joint log likelihood function have an explicit form (Besag 1974; Kashyap and Chellappa 1983), which is still difficult to solve.

In general, the lattice can be partitioned into disjoint sets (independent sets) of points called *codings*. For example, the first order process has two sets of codings, and the second order process has four sets of codings, etc. For every set of codings, the log likelihood of each point can be summed because of independence of pixels in every set. Therefore a ML estimate of parameters can be obtained by maximising this sub-joint likelihood function. However, there is lack of a reasonable and rigorous ways for combining all coding estimates to form the final estimate because these estimates are dependent. Cross and Jain (1983) use a simple averaging over all the coding estimates, and find little variation in their examples. While in Kashyap and Chellappa's experiment (1983), each coding estimate differs considerably and the simple averaging over these estimates is not satisfactory. Geman and Geman (1984) also doubted the credibility and consistence of the coding method.

4.5.3 Maximum pseudo-likelihood (MPL) estimation

Besag (1975, 1986) has proposed an alternative to the joint likelihood, known as *pseudo-likelihood* (PL), which is defined by

$$\prod_{(i,j)\in\Omega} p[x(i,j)|x(m,n),(m,n)\neq(i,j),(m,n)\in\Omega;\Theta] \tag{4.36}$$

The method considers that the random variables at each pixel are conditionally independent. Although the PL is not genuine and is not always valid, it indeed makes the parameter estimation much easier and can provide reasonably good estimates.

(1) *GMRF Model*

For the GMRF with conditional distribution of (4.21), the joint PL is

$$PL = p'(\mathbf{x}) = (2\pi\rho)^{-M/2}\exp\{-\frac{1}{2\rho}\mathbf{x}^{\mathrm{T}}\mathbf{B}\mathbf{x}\} \tag{4.37}$$

Maximising the above PL is relatively easy. The results are the same as for the LS estimate.

(2) *GD Model*

For the GD distribution, although the joint distribution is explicitly expressed, i.e. (4.25a), the partition function, (4.25b), will involve an expectation over all possible realisations under the same parameter set, and it is often intractable. Therefore, the ML estimation is practically impossible to achieve. In PL methods, the independence of each pixels is assumed, so that the joint distribution is a multiple of each single site's conditional distribution, which is also a Gibbs distribution, and defined by

$$p(x(i,j)|x(i+u,j+v),(u,v)\in\eta_s;\Theta) = \frac{1}{Z_{ij}}\exp\{-\frac{1}{k_BT}U[x(i,j)|\eta,\Theta]\} \tag{4.38a}$$

where

$$Z_{ij} = \sum_{x(i,j)} \exp\{-\frac{1}{k_BT}U[x(i,j)|\eta,\Theta]\} \tag{4.38b}$$

In each site's distribution, its partition function is simply a sum of conditional distributions over all possible pixel values at one site. Geman and Graffigne (1986) have proved the consistency of the MPL estimation. This method has been widely adopted in image texture analysis (e.g. Geman and Graffigne 1986; Cohen and Cooper 1987; Qian and Titterington 1991; Manjunath *et al.* 1990; Manjunath and Chellappa 1991).

Parameter estimation is still a very importance issue in MRF or GD model-based image analysis. How to make an consistence and efficient estimate for non-lattice images, as in many segmentation problems, remains a challenging task.

4.6 Textured Image Processing

In this section, some basic texture processing tasks are briefly reviewed. In the next chapter, we will focus on unsupervised segmentation of textured images.

4.6.1 Region segmentation

Segmenting an image into different regions is a process which assigns or groups pixels in the image into different sets, each of which has similar properties. Texture segmentation in a broad sense is segmenting an image according to its textural nature or characteristics. Texture within the same region is always assumed to have some kind of homogeneity.

There is no single standard approach to segmentation. The perceptual process involved in segmentation of a scene by the human visual system are not well understood. Segmenting methods are very dependent on the images to be analysed. We will concentrate on the segmentation based upon the textural properties of the image. Early segmentation work relied on texture feature extraction, feature distance measures and clustering algorithms. Recent segmentation work involves more accurate description of the textured images through some forms of statistical model and relies more on probabilistic measures.

Referring to the prior knowledge about the texture image to be segmented, there are basically two categories of segmentation: supervised and unsupervised. In supervised segmentation, the features or model parameters of each region, are known, or can be obtained through the analysis of the sample images provided from known categories. While in unsupervised segmentation, these properties, sometimes even including the number of the regions, are not known or only partially known. They need to be learnt during the segmentation. Simultaneous parameter estimation and segmentation are often very difficult. It needs many possible interactive steps to update parameter estimation based on new segmenting results, and to re-segment the image based on these new parameters.

In model-based texture analysis, there are two key difficulties in segmentation. One is to select an appropriate neighbourhood size over which pixels are regarded as dependent. Large neighbourhood size means more features are available, and may lead to more accurate description by the model. However, this creates a greater computational demand. Some approaches use a fixed neighbourhood size which is determined empirically. Others use a second- or third-order neighbourhood system which appears to be adequate in many cases.

The other difficulty is to select an appropriate window size from which the local model parameters are extracted. Usually different window sizes are needed during the segmentation. Sometimes the scale of the image is varied in accordance with a multiresolution scheme. Some approaches use large windows at the beginning to obtain a coarse segmentation, then use smaller windows to obtain a finer segmentation. This may avoid being trapped in a local minimum when only using small initial windows. However there is no guarantee that it will lead to a globally optimal segmentation.

In recent years, maximum *a posteriori* probability (MAP) has dominated the MRF model-based texture segmentation. There are two major reasons. One is the proof of the equivalence of MRF and GD. The other is that simulated annealing or stochastic relaxation becomes available for obtaining a MAP segmentation even though theoretically the annealing process needs a very long time to reach the optimum. In practice most applications use a fixed temperature or a fast annealing scheme. There are some alternatives, such as using mean field theory in the expectation-maximisation method for searching for optimum (Zhang 1992) and using maximum posterior marginal probability estimation (Marroquin 1989; Comer and Delp 1994). In these segmentation approaches, the texture image is modelled as a hierarchical (or doubly) MRF-Gibbs model – texture region field (higher level) and texture grey image field (lower level) (Geman and Geman 1984; Cohen and Cooper 1987; Derin and Elliott 1987; Lakshmanan and Derin 1989; Manjunath *et al.* 1990; Manjunath and Chellappa 1991; Zhang *et al.* 1994). Usually they model the texture grey image field as a second- or higher-order Gaussian MRF model, while the texture region field a first-order Gibbs model.

Given a texture image **x** on a finite lattice Ω, let the random field, $\mathbf{Y}=\{Y(i,j),\ (i,j)\in\Omega\}$, denote the underlying region field, where $Y(i,j)$ takes values from the region label set $R=\{1, 2, ...K\}$, and K is the number of region types. $Y(i,j)=y(i,j)=l$ means that the pixel (i,j) belongs to region type l. In the texture grey image level, $\mathbf{X}=\{X(i,j),\ (i,j)\in\Omega\}$, is a multi-random field. Among each region l, there is a random field which takes values from l-th region grey level set, $G^{(l)}=[g_1^{(l)}, g_2^{(l)}, \dots g_{q_l}^{(l)}]$, where $q_l^{(l)}$ denotes the number of grey levels in region l.

The objective of MAP segmentation is to assign each pixel to a proper region label, i.e. form a region realisation **Y**=**y** respect to the observed image **X**=**x**, such that the posterior probability $P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x})$ has the maximum value. Using the Bayes rule, we can write

$$P(Y=\mathrm{y}|X=\mathrm{x})=\frac{P(X=\mathrm{x}|Y=\mathrm{y})P(Y=\mathrm{y})}{P(X=\mathrm{x})} \tag{4.39}$$

Since $P(\mathbf{X}=\mathbf{x})$ is a constant referring to the region field **Y**, so maximising the above form is equivalent to maximising the numerator of the RHS of (4.39), i.e. the joint probability $P(\mathbf{Y}=\mathbf{y},\mathbf{X}=\mathbf{x}) = P(\mathbf{Y}=\mathbf{y})P(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y})$. Most existing algorithms are aimed at finding a resolution which is as close as possible to the optimal MAP estimate.

4.6.2 Texture classification

Texture classification is to identify one texture from others. Usually one image contains only one class of texture. The knowledge about each texture class is learnt in a supervised fashion, i.e. from known class samples.

Texture classification techniques have been widely used in natural image recognition (e.g. Kashyap *et al.* 1982; Vickers and Modestino 1982), aerial image classification (Weszka *et al.* 1976), and industrial product inspection (Siew *et al.* 1988). With texture features, the classification rate can be much better than simply using spectral densities of the image (Kirvida 1976). In most circumstances, the classification is required to be rotation invariant. Kashyap and Khotanzad (1986) have proposed a rotation invariant SAR model, which uses a circular symmetric AR model. The model averages the pixels of different angles from a circular neighbourhood. Mao and Jain (1992) have extended this model to multiresolution levels, and improved the classification rates. Cohen *et al.* (1991) proposed a MRF model to classify the rotated and scaled image textures.

4.6.3 Image restoration

The ultimate goal of restoration techniques is to recover the real image from a blurred, distorted, noisy or degraded one. Such blur or noise to the image may be caused by camera defects, digitalisation, transmission errors, etc. Restoration is oriented towards modelling the noise, blurring, or degradation processes, to applying the inverse process in order to remove the noise or deblur, and recover the original image. Gonzalez and Woods (1992) provide a review on this topic.

Using texture models, e.g. MRFs and GDs, to describe the image attributes and noise nature has become increasingly popular. Geman and Geman (1984) first proved that the conditional probability distribution of the real image given an observed degraded image, i.e. the posterior distribution for restoration, is a Gibbs distribution. The restoration process is the MAP estimation. The conventional *maximum entropy* restoration is a special case of MAP estimation. They proposed a stochastic relaxation algorithm to search for the global optimal reconstruction. They have proved that under certain annealing schedules (though too slow to use in practice) the algorithm will converge to the minimum energy configurations of the field. Besag (1986) has proposed an iterative method for reconstruction known as *iterated conditional modes.* Woods *et al.* (1987) have proposed a MRF model to recover noised images. Bouman and Sauer (1993) have used a Gaussian MRF model to reconstruct images from noisy data with high edge-preserving ability.

4.7 Conclusions

In this chapter, we have reviewed the most commonly used descriptions and approaches for texture processing. In particular we have reviewed the SAR, MRF, and GD models for the representation of statistical textures and their parameter estimation methods. Although the multiresolution filtering methods, such as Gabor filter bank and wavelet decomposition, are becoming increasingly popular in the past few years, the MRF model-based approaches still have a very strong influence on texture analysis, and can also yield the greatest performance when used in a multiresolution scheme. There are also some methods combining the MRF model with multiresolution or multiscale wavelet decomposition methods (e.g. Bouman and Liu 1991; Liu and Yang 1994; Chen and Kundu 1994). In the next chapter we adopt the MRF model, particularly the GMRF, for textured image segmentation. A self-organising neural network structure is proposed and incorporated in the segmentation process. The network simulates, in part, the human visual operations, but has a simple computational form, and can perform Bayesian classification or MAP segmentation.

Chapter 5

SELF-ORGANISED SEGMENTATION OF TEXTURED IMAGES

In this Chapter, a hierarchical self-organising learning structure is proposed for the unsupervised segmentation of textured images. The system combines model-based texture description especially Markov random field, a local model parameter estimator, a self-organising chain, and a local voting network. It learns to progressively estimate the model parameters for each texture region in an image and hence classifies the various region categories. The model parameters and the segmentation are updated iteratively, and their final estimates are obtained at the end of the process. The computational structure of the algorithm is relatively simple and efficient. Theoretical analysis of the algorithm shows that the algorithm will converge to the maximum likelihood or maximum *a posteriori* (if the Bayesian SOM is used) segmentation. A number of experimental results on various images are provided. A simple parallel stochastic boundary relaxation algorithm is also proposed for improving the segmentation quality at boundaries. The algorithm reconfigures a boundary in a local area encompassing a boundary according to the mean-square-error energies. It can be used after the segmentation as a post-processor, or it can be incorporated within the segmentation process as a on-line validation scheme. Based on this idea, a simple on-line minimum mean-square-error validation scheme is proposed for the validation of the number of regions, when this is not known *a priori*. Experiments have demonstrated the usefulness of this approach.

5.1 Introduction

Most recent work on model-based image segmentation employs MRF model parameters, clustering algorithms, and/or deterministic or stochastic relaxation, in order to obtain a maximum a *posteriori* (MAP) segmentation. One difficulty in the unsupervised segmentation is the lack of *a priori* knowledge. The model parameters of the different regions, on which the segmentation process depends, are unknown, or need to be estimated using the emerging segmentation results formed during the process. Hence, these estimates are always inaccurate or incomplete. The segmentation has to be achieved by an iterative procedure (e.g. Lakshmanan and Derin 1989; Manjunath and Chellappa 1991; Zhang *et al.* 1994). Another difficulty is the accurate estimation for the model parameters. *Maximum likelihood* (ML) estimation is essentially impossible in most cases. Alternatives involve using pseudo-likelihood function to yield a *maximum pseudo-likelihood* (MPL) estimate, in which the independence of pixels has to be assumed. Even so the MPL estimate is still computationally intensive.

The least-square (LS) estimate provides an easy and fast parameter estimation for the Gaussian MRF (GMRF). Although it can only achieve asymptotically consistency under restricted conditions, its implementation efficiency and low computational demanding make it a satisfactory estimator. In the first part of the proposed segmentation algorithm, an LS estimator gives rapid but coarse parameter estimates over the windowed pixel data. Then, a two-level SOM structure learns to classify and cluster these noisy, coarse parameters and to re-estimate them in order to reduce the noise.

The SOM network is used in the proposed segmentation algorithm as the parameter re-estimator (after the local LS estimator) and region classifier. Lampinen and Oja (1989) proposed a self-organising auto-regressive (AR) model for segmenting textured images. In their algorithm, each texture was modelled as an AR series. SOM's competitive matching law and Widrow-Hoff least-mean-square learning rule were combined to obtain the model parameters. However, for two-dimensional images, causal AR series are not a valid assumption. In addition, their learning is a pixel-based adaptive least-mean-square-error method, though the estimation errors were averaged over the past. The textures are characterised by groups or blocks of pixels, rather than by individual pixels, therefore the inhomogeneity of each single pixel could heavily affect their parameter estimation and so the resulting segmentation.

This chapter is organised as follows. In the next section, a detailed examination of the LS estimation and its performance over windows with different sizes are provided. In Section 5.3, the proposed segmentation structure is gradually introduced, together with various experimental results. Section 5.4 presents an analysis of the optimality of the algorithm. It indicates that the Bayesian SOM, proposed in Chapter 3, can be used in the segmentation algorithm for an improved interpretation of the data structures. A parallel boundary relaxation algorithm is proposed in Section 5.5. A validation method given in Section 5.6 can be useful for a fully unsupervised situation, where the number of the region clusters is also unknown.

5.2 On-Line Model Parameter Estimation

Model parameter estimation is very important in model based texture image processing. In many applications, there is a requirement for on-line estimation of model parameters. Therefore the estimation method has to be computationally efficient, simple, and accurate. Among several existing estimation methods discussed in Chapter 4, the LS method proposed by Kashyap and Chellappa (1993) is a simple but asymptotically consistent estimation. This LS method is also equivalent to the MPL estimation for GMRF as previously noted. Thus it is a good choice for practical problems. In the on-line parameter estimation, the whole region for each texture categories may be not available. The region areas are also variables. The estimation has to be operated in an *incomplete data* environment. According to the texture's block and local properties (i.e. texture properties are contained in a small block of the entire texture), we can use a small window (small compared to the image size, but large enough to recover the texture local properties) to estimate the model parameter in this window, which is considered homogeneous. In this section we shall first examine the LS estimation for the whole texture, and from various smaller windows. This is to pave the way for the next section, in which on-line estimation and segmentation are carried out simultaneously.

5.2.1 Asymptotically consistent least-square estimation

In zero-mean cases, the LS estimate for the parameter set of the MRF model is Eqn. (4.28). The calculations are straightforward. $\mathbf{q}(i,j)$ is a neighbouring pixel pair vector, $\mathbf{q}(i,j)\mathbf{q}^T(i,j)$ is a k×k matrix (k=2, 4, 6, ... for the first-, second-, third-, ... order models respectively). The summation in Eqn. (4.28a) includes all pixels in the interior set (i.e. excluding the boundaries). Care should be taken when using multiple-grey-level images and high-order models, since the determination of the matrix $\Sigma\mathbf{q}(i,j)\mathbf{q}^T(i,j)$ might result in overflow of variable bytes if the image size is very large. A solution is to normalise the data, so each pixel will be within [0,1) range. When the mean of the image is not zero, the pixel values in Eqn. (4.28) should be subtracted by the mean value. The mean value itself should be also considered as a parameter in this case. A proper normalisation to the variance terms is also needed in practice. We propose to use the sample's variance to normalise to the model's variance (for binary textures, such normalisation is not necessary). The normalised variance is

$$\hat{\sigma}' = \sqrt{\frac{\hat{\sigma}^2 = \dfrac{1}{M'} \sum_{(i,j)\in\Omega_I} [x(i,j) - m - \hat{\Theta}^T \mathbf{q}(i,j)]^2}{\dfrac{1}{M} \sum_{(i,j)\in\Omega} [x(i,j) - m]^2}} \tag{5.1}$$

where m is the sample mean over the entire texture on Ω, $\hat{\Theta}$ is the estimate of the parameter set, $x(i, j)$ is the pixel value at point (i, j), M is the total number of pixels in Ω, and M' is the total number of pixels in the interior lattice.

Consider first some synthetic textures and their corresponding MRF parameter estimates using the LS method. Several textures were simulated using the method described in Section 4.3.3 with the parameter settings given in Table 5.1. The two grey levels, black and white, were set to the equal probability. The results shown in Fig. 5.1 are after 100,000 iterations, and are 128×128 pixels squares.

setting	(a)	(b)	(c)	(d)	(e)	(f)
a	-3.0	-4.0	6.0	0	-3.0	-2.0
***b*11**	3.0	2.0	-3.0	3.0	3.0	0.0
***b*12**	0.0	2.0	-3.0	-1.0	3.0	0.0
***b*21**	3.0	2.0	-3.0	-1.0	-2.0	0.0
***b*22**	0.0	2.0	-3.0	-1.0	-2.0	3.0

Table 5.1: Simulation parameter settings.

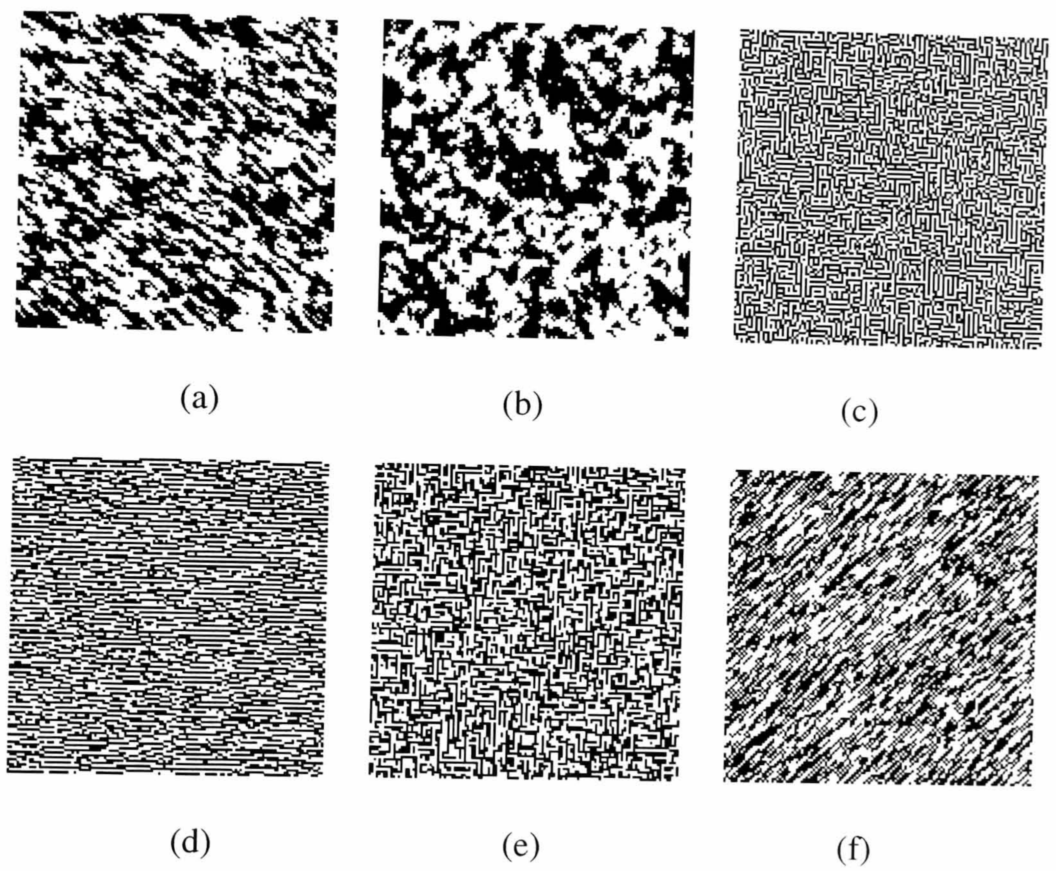


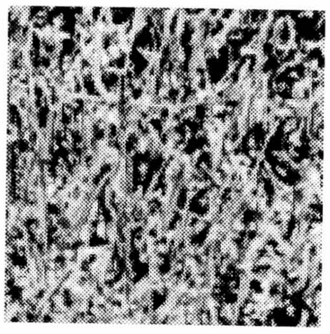
Figure 5.1: Some synthetic textures (128×128).

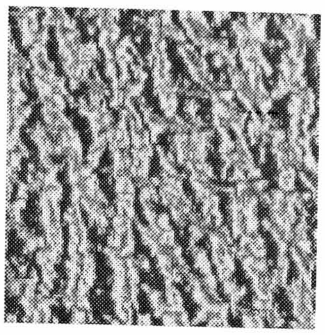
The corresponding estimation results of their second-order MRF model are listed in Table 5.2. It can be seen that these parameters are distinctive, and follow the simulation parameter settings.

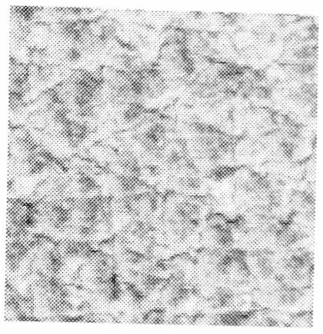
para-meters	Fig. 5.1 (a)	Fig. 5.1 (b)	Fig. 5.1 (c)	Fig. 5.1 (d)	Fig. 5.1 (e)	Fig. 5.1 (f)
θ_{11}	0.2887	0.1595	-0.2460	0.3209	0.3573	-0.0036
θ_{12}	-0.0353	0.1648	-0.2455	-0.1006	0.3498	0.0032
θ_{21}	0.2741	0.1077	-0.2631	-0.0586	-0.2385	0.0036
θ_{22}	-0.0071	0.1052	-0.2563	-0.0482	-0.2320	0.4459
σ	0.3144	0.3031	0.3219	0.3120	0.3485	0.3436

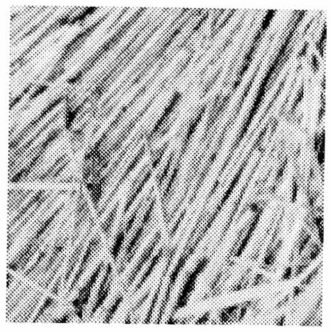
Table 5.2: Parameter estimates for the textures in Fig. 5.1.

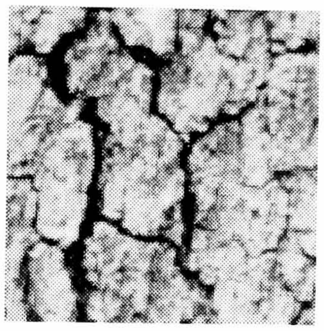
Examples of 8-bit natural textures (128×128 in size) are shown in Fig. 5.2. They are Brodatz (1966) textures. Their corresponding third-order GMRF model parameters were estimated as shown in Table 5.3.


(a) Grass (D9)


(b) Leather (D24)


(c) Pigskin (D92)


(d) Straw (D15)


(e) Treebark (D12)

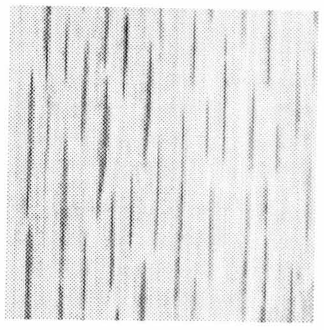

(f) Woodgrain (D68)

Figure 5.2: Natural textures examples (128×128).

parameters	Grass	Pigskin	Leather	Straw	Tree-bark	Wood-grain
θ_{11}	0.3594	0.4812	0.2008	-0.0775	0.4706	0.2292
θ_{12}	0.6140	0.3456	0.5754	0.5414	0.5012	0.5420
θ_{21}	-0.1707	-0.1111	-0.0329	0.0443	-0.1871	-0.1184
θ_{22}	-0.1520	-0.0986	-0.0886	0.2082	-0.1708	-0.1097
θ_{31}	-0.0277	-0.0761	-0.0198	-0.0027	-0.0535	-0.0001
θ_{32}	-0.1505	-0.0517	-0.1493	-0.2167	-0.0696	-0.0433
m'	0.4687	0.6423	0.5091	0.6067	0.5653	0.7289
σ'	0.2506	0.3442	0.3710	0.2334	0.1637	0.1288

Table 5.3: Third-order GMRF parameters estimated from the textures shown in Fig. 5.2. Here m' and σ' are the normalised mean standard variance.

5.2.2 Window-based model parameter estimation

The above estimation examples are applications of the LS estimate method to an entire texture region (one texture category). The parameter set reflects the average or overall properties of the texture. For texture classification tasks, such a parameter set can be employed as the representative features for each texture. Comparison among the parameter sets using Euclidean distance metric (or others) should give good classification performance. In model-based segmentation problems, model parameters are also the major representing figures in distinguishing different texture regions. However, the overall estimation of one texture region may not be available at least initially as the configuration of the regions still need to estimate. To obtain an accurate estimate of parameter set for each region so as to obtain the correct region segmentation under unsupervised situation, is one of the most challenging tasks in image processing. In the next section, a hierarchical structure for achieving such an objective is proposed, in which the parameter estimation is from various local windows. Therefore, we now examine the case where estimates are from only part of an entire region, and compare the difference between these estimates and those from the whole region.

Since texture is a property of a group of neighbouring pixels rather than isolated pixels, basic texture characters are preserved in local areas. So recovering parameters from a smaller window is feasible; and the resulting estimate will be an approximation to the estimate for the whole region. The accuracy of this approximation depends on the size of the window (i.e. the larger the better). However large windows make good segmentation less likely, since they are more likely to contain more than one texture region so the homogeneity assumption of the window will not be valid. The correct window size is one of the key problems in the model-based segmentation. The window has to be small enough for accurate segmentation, but large enough to represent the texture characteristics. The low limit of the window size depends on the nature of the texture and recognition requirement. For example, in a brick-wall texture, a small window can only tell the properties of the brick, a window large enough to cover several bricks will show the characteristics of the wall. Usually homogeneous textures, e.g. synthetic textures, can preserve their properties in a very small area, but most natural textures need a larger window. We shall discuss this aspect later.

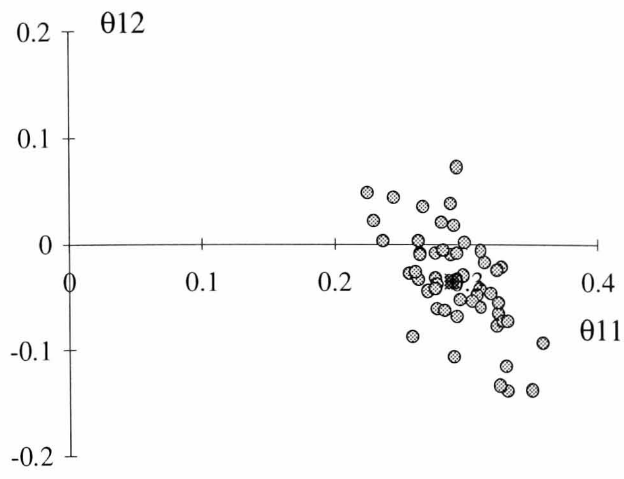
As window-based estimation provides an approximation for the entire region parameters, we can model such an estimate as a noisy estimate, i.e.

$$\hat{\theta}(t) = \theta^* + n(t) \tag{5.2}$$

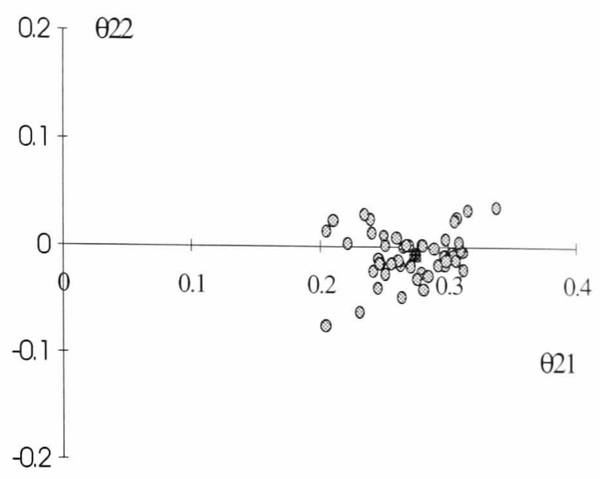
where θ^* is the correct parameter set (including θ_{11}, θ_{12}, θ_{21}, ..., m (or m'), σ (or σ')); $n(t)$ is the noise term, which can be regard as normally distributed; and t is the time step.

This hypothesis describes the estimate from a window as the real parameter plus a random noise. As the iteration progresses, the actual parameters can be recovered by applying an appropriate statistical technique to the window estimates. We now give some examples of window-based parameter estimates for the synthetic and natural textures shown in Figs. 5.1 and 5.2.

Fig. 5.3 shows 50 estimates of the second order GMRF parameter estimates from a 30x30 window which is randomly located on the image Fig. 5.1 (a) at each time. We split the parameters in pairs in order to visualise them. As we can see, window estimates are well distributed around the estimate from the whole region (given in Table 5.2, and marked in dark squares). The distribution in the variance parameters has the same appearance as these two pairs; and examination on the other images in Fig. 5.1 gives similar distribution results.



(a) θ_{11} and θ_{12}



(b) θ_{21} and θ_{22}

Figure 5.3: Window-based parameter estimates from the synthetic texture Fig. 5.1(a) with window size of 30×30. Dark-square represents the estimate from the entire texture.

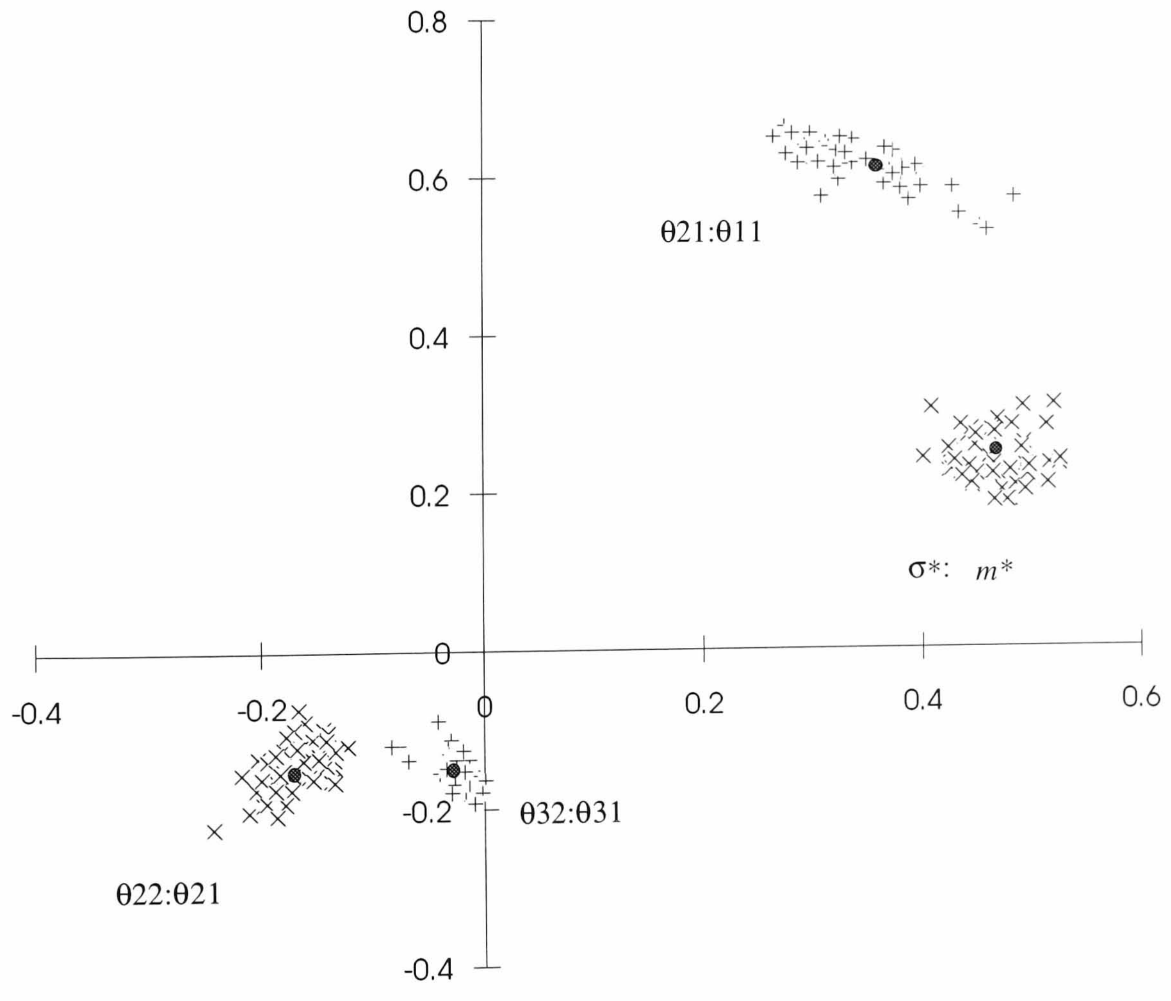


Figure 5.4: Window-based parameter estimates from the natural texture Fig. 5.2(a) with window size of 30×30. Dark-dots represent the estimates from the whole texture.

For natural texture images, the hypothesis is also acceptable. A 30×30 window has been used on the images shown in Fig. 5.2, their MRF parameters estimated from the windowed regions support Eqn. (5.2). Here we just present the results of one texture – grass. Its third-order GMRF parameters for 50 estimates are shown in Fig. 5.4. The parameters: θ_{11} and θ_{12} (first-order); θ_{21} and θ_{22} (second-order); θ_{31} and θ_{32} (third-order); m' and σ' (mean and variance), have been grouped into pairs so that they can be shown in one figure for convenience.

As we can see the estimate from the whole image is approximately the statistical centre of the window-based estimates. In other words, if the mean calculation is applied to the window-based estimates, then the real parameter can be recovered.

The window size certainly controls the accuracy or dispersion of the estimates. The smaller the window the larger variance in the estimates. We use the total variance, which is a sum of the variances so 50 such estimates, to describe the influence of the window size. Fig. 5.5 shows the results for two examples, synthetic texture Fig. 5.1 (a) and natural texture Fig 5.2 (a). The window size is varied from 10× 10 to 50×50. At each size, 50 estimates from the window were obtained. Such figures can provide guidance for selecting window size. As we can see when the window size is smaller than 10×10, the estimate from the window is generally very unreliable for both synthetic and natural textures. If the window size is 20×20 or over, the window-based estimate will be sufficient for synthetic or very homogeneous natural textures. For other natural textures, however, the minimum window size needs to be about 30×30. After post-processing of these raw estimates, a near-optimal estimate of the model parameters can be obtained.

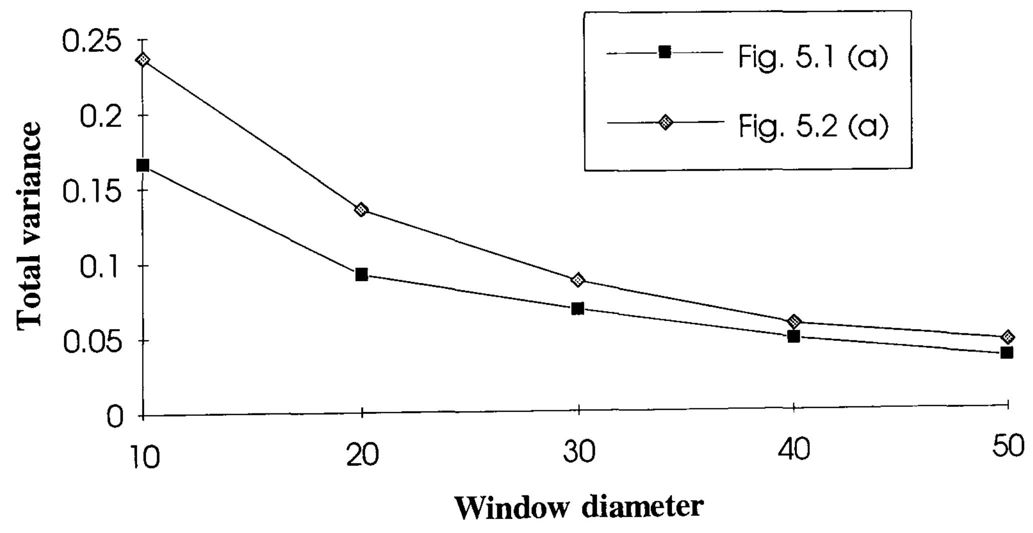


Figure 5.5: Variances of window-base estimates vs window sizes.

In the next section, these estimates, instead of raw pixel data, will be used as input data. An SOM network is used to further remove noise and so to converge to the real parameter set for each region. At the same time, the SOM network also acts as a classifier, which assigns the current parameter estimate to one region type, and produces a winning signal to the upper segmenting layer. The segmenting layer then updates the region type of the area that the current window covers.

5.3 Hierarchical Self-Organised Segmentation Structure

5.3.1 A shrinking window to emulate preattentive and attentive visions

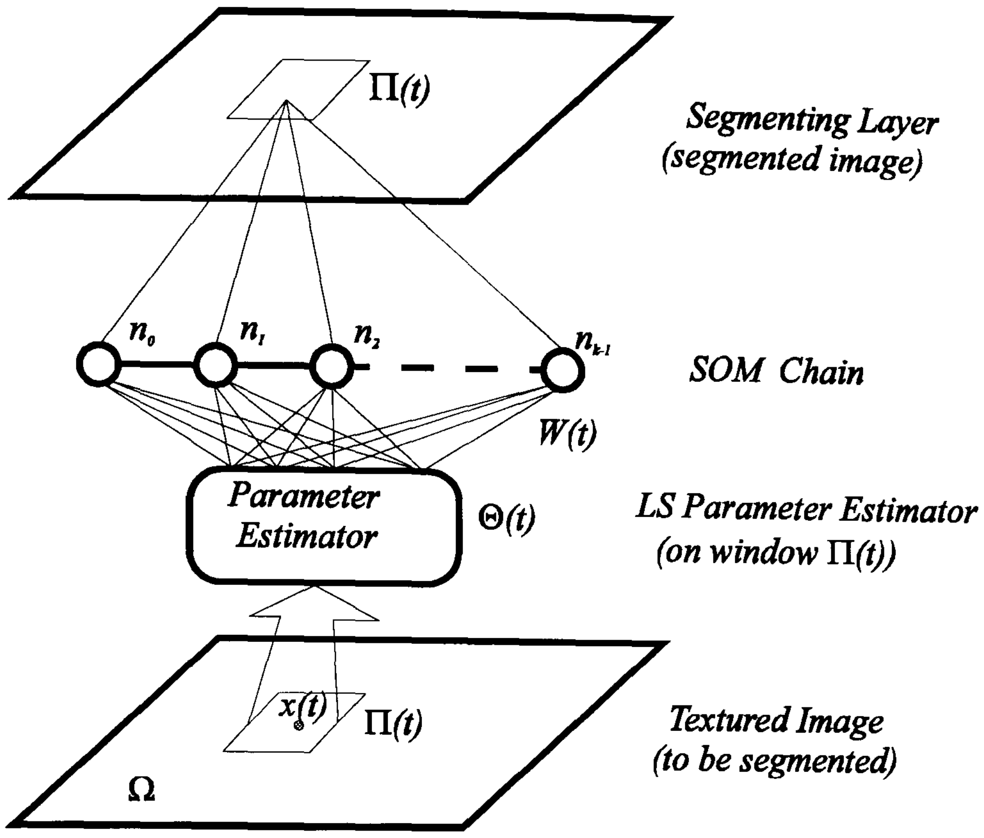
When we look at an image, first we glance it, i.e. we take a general look on the whole image or large parts of it. Then we gradually focus on to local details in the image. These are two well-known human visual operations: *preattentive* and *attentive* vision, which were briefly described in Chapter 4. To apply these principles to model-based segmentation and estimation problems, we can use a shrinking window, whose size is very large at the beginning and then gradually shrinks to small sizes. Large windows will give an overall description about the image and its region distribution, while small windows can give more accurate descriptions especially at the region's boundaries.

The preattentive and attentive vision principle can also be applied in other ways. For example, one can first use large windows to look for large homogeneous blocks, and use them as region bases. This process corresponds to the preattentive phase. Then small windows can be used to refine the area for each region by adding or subtracting small window areas round the region boundaries. This process corresponds to the attentive vision phase.

5.3.2 A primary self-organised segmentation (SOS) network

From the analysis of window-based parameter estimation in previous sections and analysis of SOM performance, we can now develop a basic network for textured image segmentation using an MRF model and SOM network. The general structure is depicted in Fig. 5.6.

$\Pi(t)$ represents the shrinking random window whose size shrinks with time and whose location is chosen randomly at each time step. The *parameter estimator* performs a crude estimate for the area that the current window covers. Then the *estimating* SOM chain, will remove noise and converge to the mean or true parameter set for each region. At the same time, the SOM chain also acts as a classifier, which assigns the current parameter estimate to a region type, and produces a winning signal for the top layer, namely the *segmenting layer*. This segmenting layer then updates the region type of the area that the current window covers, according to the region type of the parameter estimate. The function of this layer is a simplified "winner-take-all", or more accurately, "winner-change-all" (WCA). Although in the original idea, the segmenting layer was to be a full SOM network (in 2-D with as many neurons as pixels), so that the neurons in the segmenting layer would converge to the means of the pixel labels. However, the simplified structure can produce very good results.



Π(t)
Segmenting Layer (segmented image)
n_0 n_1 n_2 n_{k-1}
SOM Chain
W(t)
Parameter Estimator
Θ(t)
LS Parameter Estimator (on window Π(t))
x(t)
Π(t)
Ω
Textured Image (to be segmented)

Figure 5.6: Self-organised segmentation structure.

The algorithm is given below:

Self-Organised Segmentation (SOS) Algorithm:

For a textured image $\{x(i,j),(i,j)\in\Omega\}$:

(0) *Initialisation*: The initial weights of the SOM chain are chosen randomly. The number of neurons in the chain is set by the number of texture regions in the image, *K*, which is assumed known, or the number of texture regions that we wish to segment into (otherwise class number validation is needed, see Section 5.6 for more details). Set the initial window size to a large value, say, half of the image size. Set a small threshold $\varepsilon>0$.

(1) *Window-based parameter estimation*: randomly place the window, $\Pi(t)$, centred on pixel $x(i, j)$ (ensuring the window is within the image lattice); extract the model parameters by a LS estimator (see Section 4.5.1 and Section 5.2.2) from the area that the current window covers.

(2) *Winner selection*:

$$v(t)=\arg\min_{k=0,1,..K-1} d[W_k(t)-\hat{\Theta}(t)] \tag{5.3}$$

where $\{W_k(t), k=0, 1, ...K\text{-}1\}$ is the weight vector at time t, whose dimension is equal to that of the parameter vector, Θ, and $\hat{\Theta}(t)$ is the parameter estimate of $\Pi(t)$.

(3) *Update the weights* of the winning neuron and its neighbouring neurons according to the SOM algorithm, i.e.

$$W_k(t+1)=W_k(t)+\alpha(t)h(t)[\hat{\Theta}(t)-W_k(t)], \qquad \forall k\in\eta_v \tag{5.4}$$

where η_v is the neighbourhood of the winner v (including v).

(4) *Update the segmenting layer*:

$$label[x(i,j),(i,j) \in \Pi(t)] = v \qquad (5.5)$$

i.e. "winner-change-all".

(5) If $|[W_v(t+1) - W_v(t)] / W_v(t)| \le \varepsilon$, then halt the process with the final segmented image at the segmenting layer and the regions parameters represented by the neurons weights. Otherwise go back to step (1) and continue.

Experiments using the SOS algorithm have been undertaken on both synthetic and natural (Brodatz) composite textured images (128×128 in size). Some typical results are given in Fig. 5.7. As we can see, the algorithm can correctly segregate the composite images. Although there are some errors at the region boundaries (and occasionally elsewhere), the main part of each region has been correctly assigned.

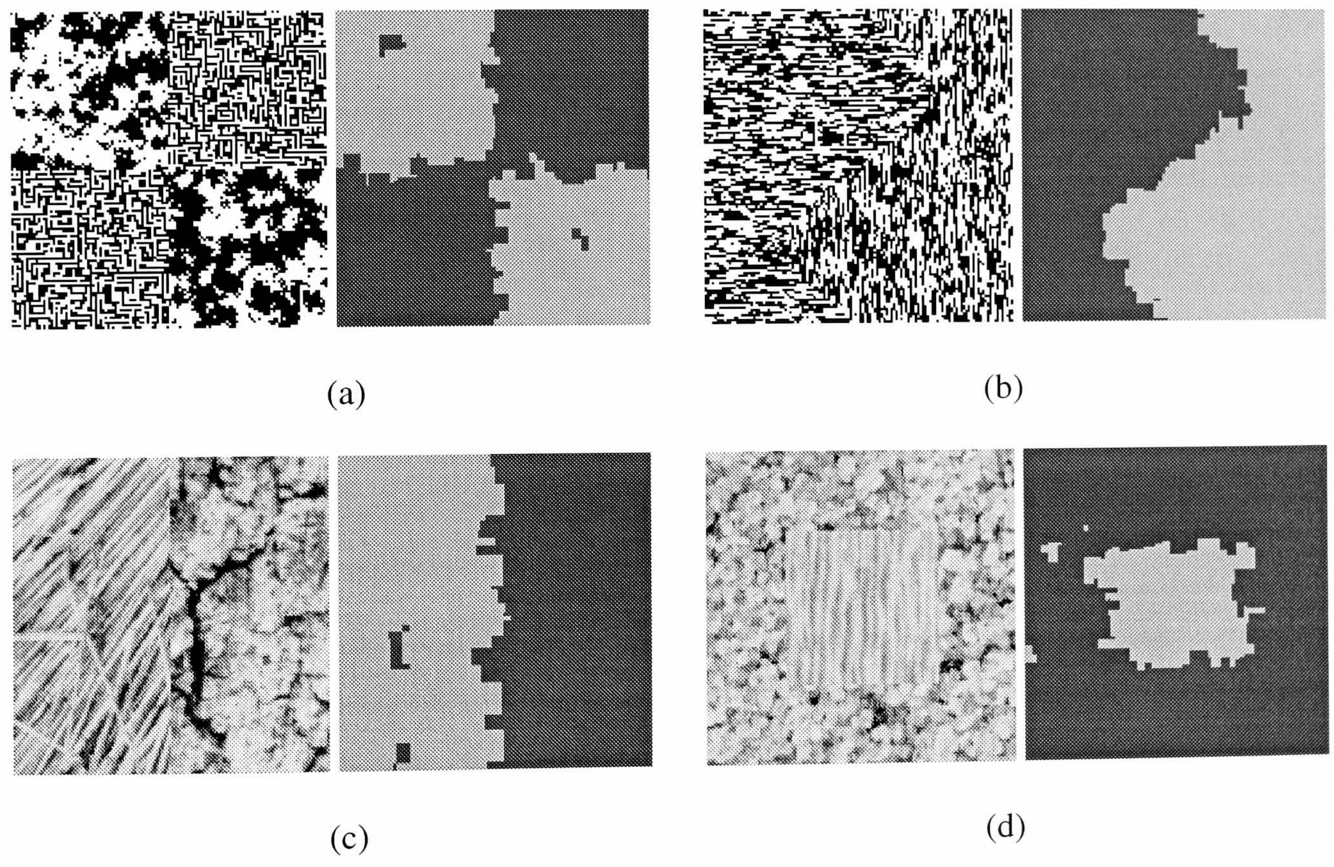
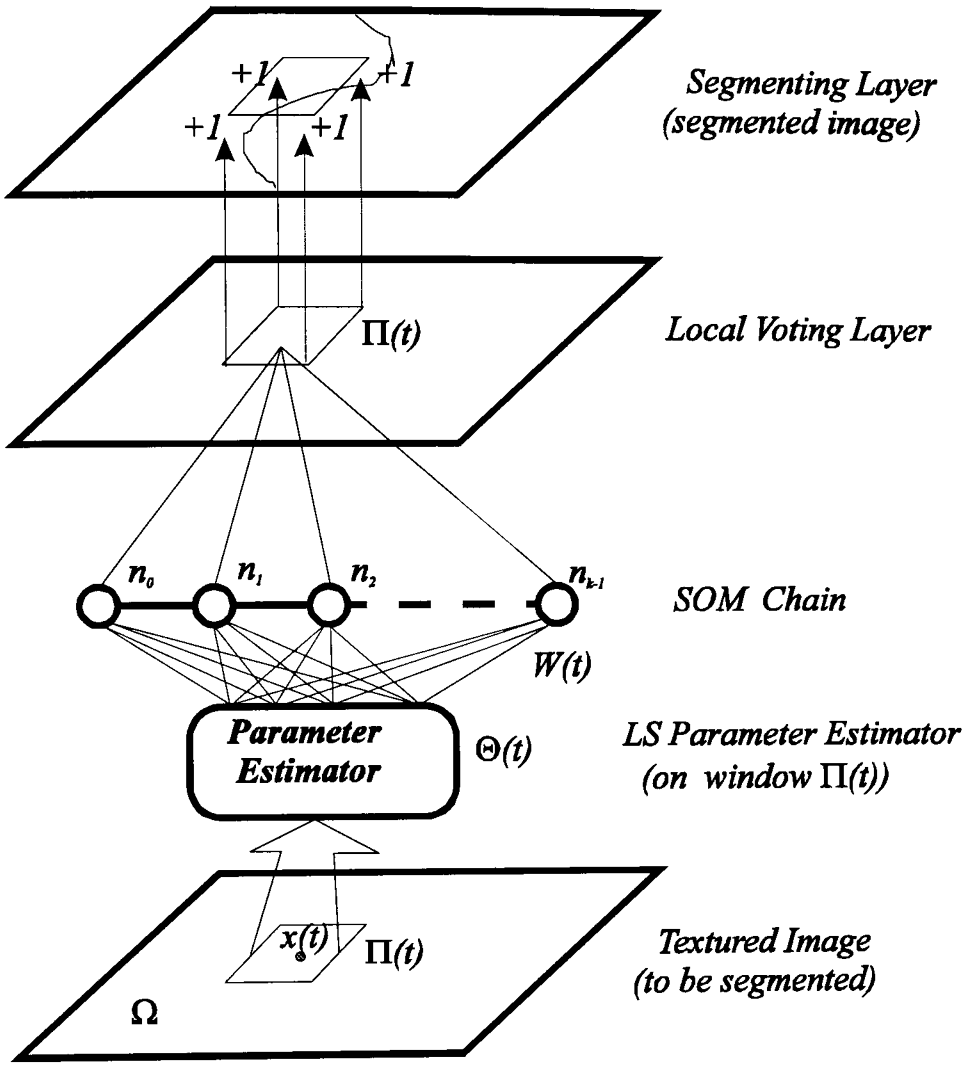


Figure 5.7: Segmentation results of the SOS algorithm.

5.3.3 A hierarchical self-organised segmentation (HSOS) network

The reason for the above errors is the inhomogeneity even within a single texture region. This can be overcome by using larger windows so that smoothed or averaged texture parameters can be derived from the window. However larger windows give poor resolution at texture boundaries. The segmenting layer's WCA algorithm does not take account the previous segmentation results. Going back to the original idea (i.e. fully implementing a 2-D SOM for the segmenting layer) would give an ideal solution. However, it would be computationally very costly. Instead, we can use an alternative scheme called the *local voting* (LV) scheme to simplify computation. The LV layer takes into account the results (region labels) of each iteration, and gives the true label of a pixel according to its highest label votes. The structure is given in Fig. 5.8. The algorithm is stated below (steps 1-3 are similar to the SOS algorithm):



+1
+1
+1
+1
Segmenting Layer (segmented image)
Π(t)
Local Voting Layer
n_0
n_1
n_2
n_{k-1}
SOM Chain
W(t)
Parameter Estimator
Θ(t)
LS Parameter Estimator (on window Π(t))
x(t)
Π(t)
Ω
Textured Image (to be segmented)

Figure 5.8: Hierarchical self-organised segmentation network.

Hierarchical Self-Organised Segmentation (HSOS) Algorithm:

For a textured image $\{x(i,j),(i,j)\in\Omega\}$:

(0) *Initialisation*: The initial weights of the SOM chain are chosen randomly. The number of the neurons in the chain is chosen as the number of texture regions in the image, K, which is assumed known. Set the initial window size to a large value. Set *an integer threshold* ε>0. Set each vote of the segmenting layer to zero: $\{L_{ij}(0)=\mathbf{0}, (i, j)\in\Omega\}$, where L_{ij} is the voting vector for the pixel at (i, j), which has K elements $\{l_{ij}^0, l_{ij}^1, \dots l_{ij}^{K-1}\}$ corresponding to the votes for each region class member.

(1) *Window-based parameter estimation*: randomly place the window, $\Pi(t)$, on the image, and assume its centre is at pixel $x(i, j)$; extract the model parameters using a LS estimator from the current window.

(2) *SOM winner selection* by

$$v(t) = \arg \min_{k=0,1,..K-1} d[W_k(t) - \hat{\Theta}(t)] \tag{5.6}$$

(3) *Update the weights* of the winning neuron and its neighbouring neurons by,

$$W_k(t+1) = W_k(t) + \alpha(t)h(t)[\hat{\Theta}(t) - W_k(t)], \qquad \forall k \in \eta_v \tag{5.7}$$

(4) *Update the voting vectors* by

$$\begin{matrix} l_{ij}^v(t+1) = l_{ij}^v(t) + 1 \\ l_{ij}^k(t+1) = l_{ij}^k(t) \end{matrix}, \quad (i,j) \in \Pi(t), \forall k \neq v \tag{5.8}$$

(5) *Winner in the segmenting layer:* The largest voting element of each voting vector indicates the region class for this pixel, i.e.

$$label[x(i,j),(i,j) \in \Omega, t+1] = \arg\max_k \{l_{ij}^k(t+1)\} \tag{5.9}$$

(6) If $\sum_{(i,j)\in\Pi} \delta\{label[(i,j),t+1] - label[(i,j),t]\} \leq \varepsilon$, (where $\delta(x) = 1$, if $x = 0$; or 0 otherwise), then halt the process with segmented image at the segmenting layer and regions parameters represented by neurons' weights. Otherwise go back to step (1) and continue.

5.3.4 Experimental results

Extensive experiments using the HSOS algorithm for segmenting both synthetic and natural textured images have been carried out. Typical results are given in Fig. 5.9 through Fig. 5.12. Only a second-order MRF model (and a single resolution) was used in these tests. A higher-order model and/or multiresolution would improve the accuracy of the segmentation. What we are demonstrating here is the operational ability of the HSOS algorithm in segmenting textured images. A theoretical analysis of the optimality of the SOS and HSOS algorithms will be given in the next section. As can be seen from the figures, the results are promising and very close to the actual boundaries. The results of the HSOS network are smoother than those of the SOS algorithm, since it uses one more layer (i.e. LV) for performing a spatial median filter function.

(1) ***Synthetic Textures***

The images are 128 × 128 in size and are composed of two binary synthetic textures (of several shapes). Five sets of results after 2,000 iterations are shown in Fig. 5.9, where left-hand-side (LHS) images are the composite images; middle images are the first layer's output, corresponding to the SOS segmentation result; and right-hand-side (RHS) images are the final layer's output (i.e. the results of the HSOS). As we can see the final segmentations are much smoother and cleaner than the middle layer outputs. The algorithm has consistently produced very good segmentation.

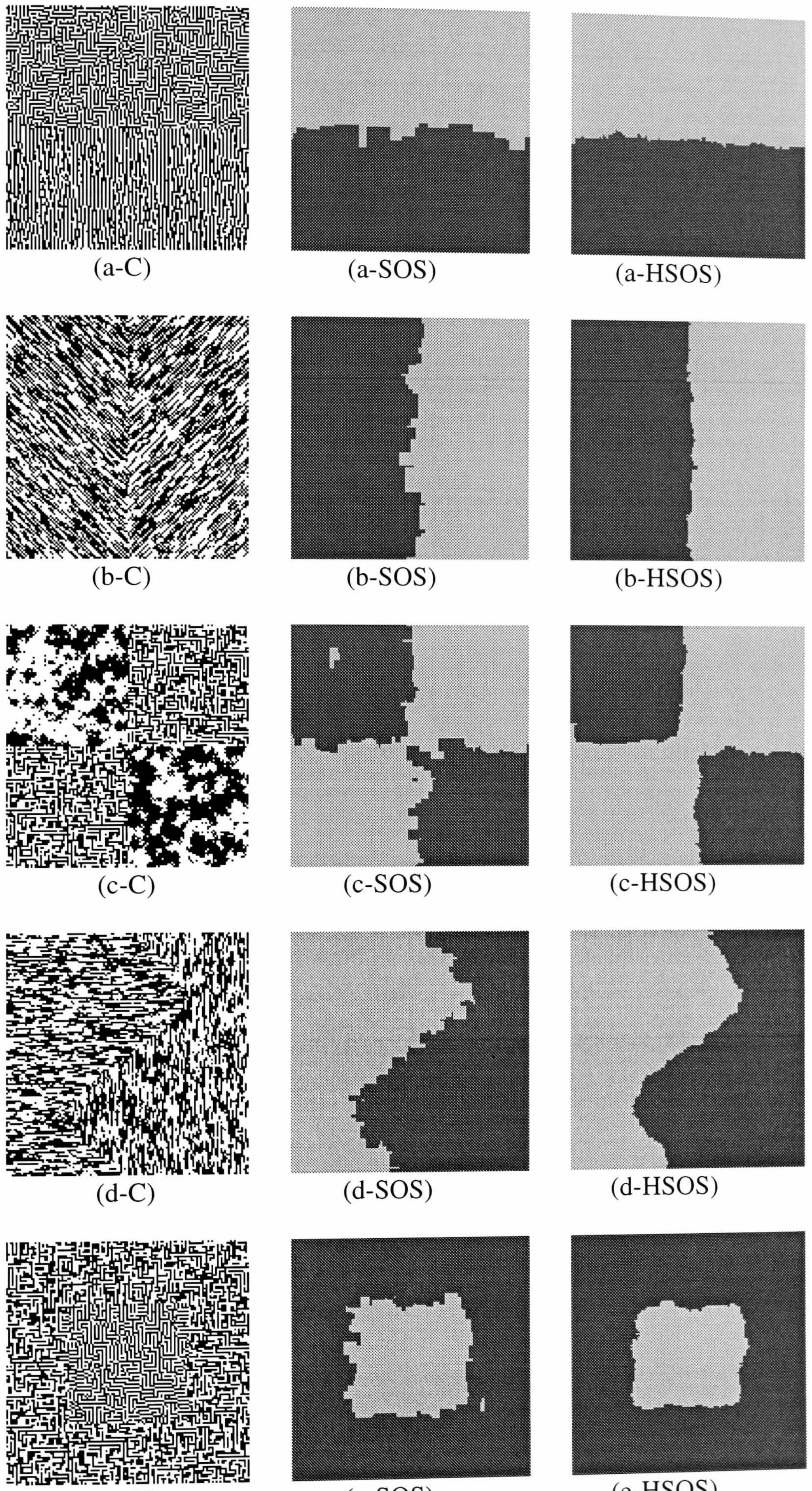


Figure 5.9: The HSOS algorithm on synthetic textured images (128×128). (x-C): Composite images; (x-SOS): Outputs of the first layer; (x-HSOS): Final segmenting outputs.

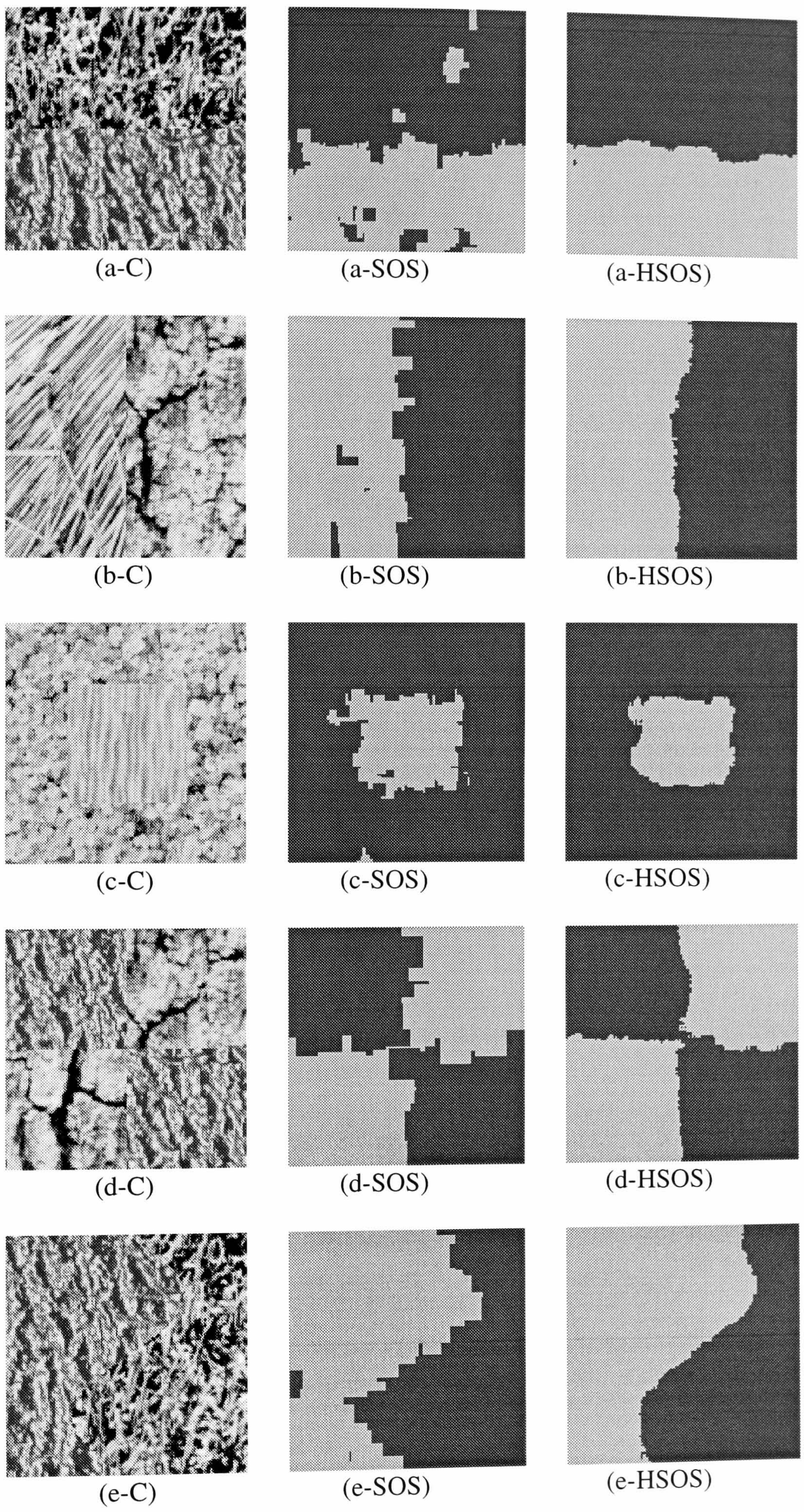


Figure 5.10: The HSOS algorithm on natural textured images (128×128). (x-C): Composite images; (x-SOS): Outputs of first layer; (x-HSOS): Final segmenting outputs.

(2) ***Natural Textures***

The images shown in Fig. 5.10 are 128×128 in size. The composite images are taken from the Brodatz album (Brodatz 1966). They are in 8-bit grey levels. The textures used are grass (D9), tree bark (D12), straw (D15), calf leather (D24), beach sand (D29), water (D38), wood grain (D68), and pig skin (D92). The final segmentations (RHS images) are also much smoother and contain less noise than the first layer's outputs (middle images). Compared to the synthetic textures, natural textures are less homogeneous, so slightly more noise is expected especially at boundaries. However the results are still very good. For example, the tree bark image in Fig. 5.10 (b) and (d) is very inhomogeneous and has large block attributes, however, it has been correctly segregated.

(3) ***Multi-Region Textures***

In the previous two-region texture examples, the neighbourhood function of the SOM is not important, as long as it can bring both neurons into activity (this can be achieved by allowing the non-winning neuron to undergo partial learning during the first few iterations, say 10). The two neurons will always globally segregate the image. However, for multi-region textures, allowing all the neurons to be active is not sufficient, since one neuron may cover two regions, while two other neurons may share one region (i.e. split one texture region into two, especially when this region is not very homogeneous), or one neuron may converge to some boundary areas, where the parameter estimates are always noisy and poor. These effects will lead to an incorrect segmentation. In such cases, the neighbourhood function plays an important role in allowing each neuron response to just one texture region so forming a globally correct segmentation. We have found that maintaining a large neighbourhood for a longer time is helpful in forming globally correct segmentation. During this stage (global ordering stage), we also found that a low-noise parameter estimate is also helpful. Thus a small (but not too small) estimating window is used during this stage, otherwise a larger window will have a greater probability of overlaying a number of regions. A proper validation scheme may be needed to assist in the segmentation process (see Section 5.6). Here we will just consider the role of the neighbourhood function.

In our experiments, the neighbourhood of the SOM chain was kept active for the first 1000 iterations. The amplitude of the neighbourhood function for the nearest neighbouring neurons is initially set to 0.6, decreasing linearly with time to 0 at the 1,000th iteration, and then remains at 0. The amplitude of the neighbourhood function for the second nearest neighbouring neurons is initially set to 0.2, decaying to 0 at the 1,000th iteration, and remains at 0. The segmenting process can be halted by monitoring the changing rate in the final layer, or simply by setting a fixed iteration numbers. Figures 5.11 and 5.12 show some typical results of the HSOS algorithm for some 4-region synthetic and natural textured images after 5,000 iterations. In general, more noise exists in natural textured images since they are less homogeneous than synthetic textures. For this reason, the minimum size of the estimating window for the natural textures should be larger than that for the synthetic textures. In our experiments, it was set to 20×20 and 15×15 for the natural and synthetic textures respectively. Good learning and ordering of the SOM depend on the appropriate shrinking rate of the neighbourhood function (which is set empirically in most cases) and may also depend on the input sequence (which is usually a purely random sequence).

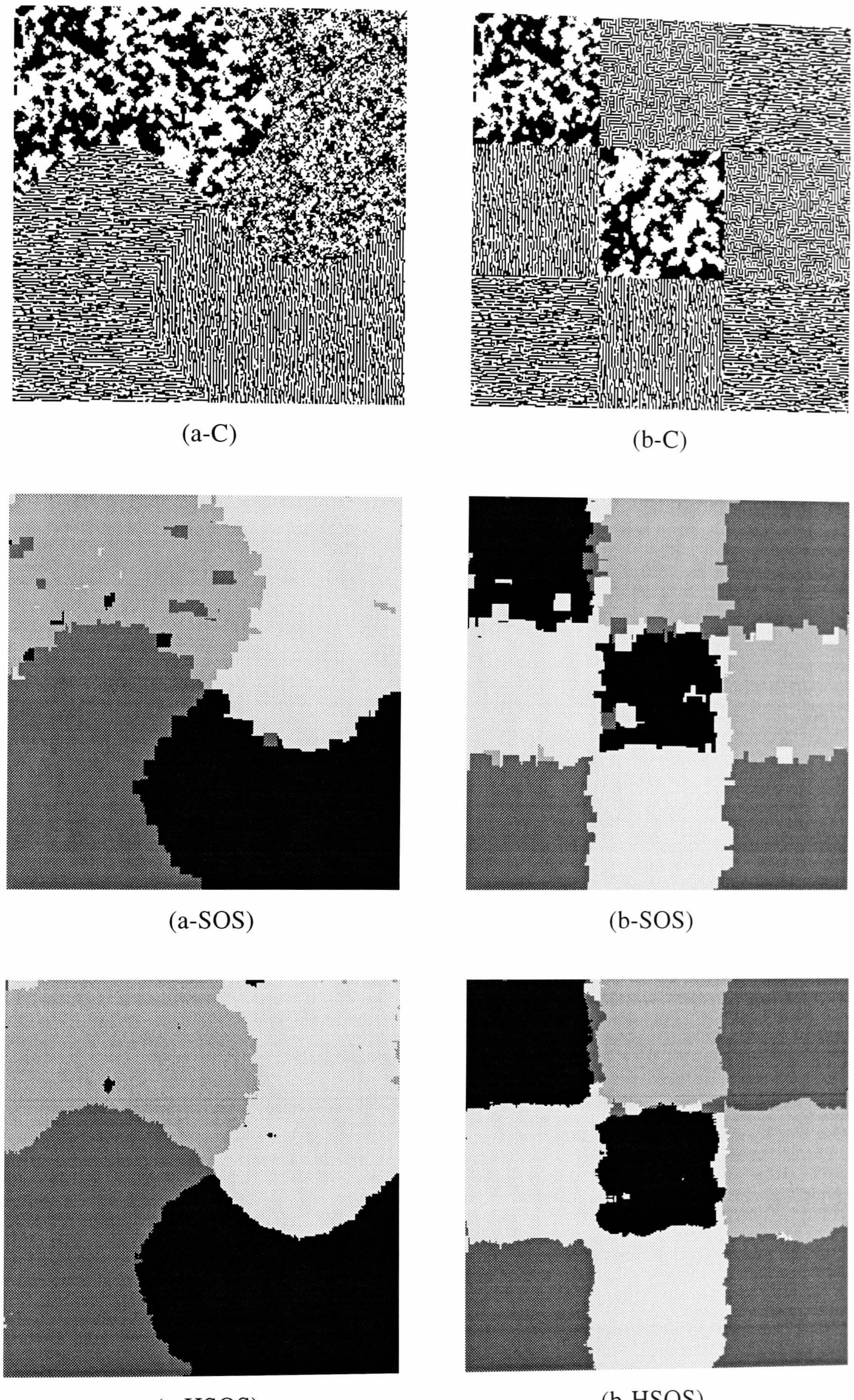


Figure 5.11: The HSOS algorithm on multi-region synthetic images (256×256). (x-C): Composite images; (x-SOS): First layer outputs; (x-HSOS): Final segmentation outputs.

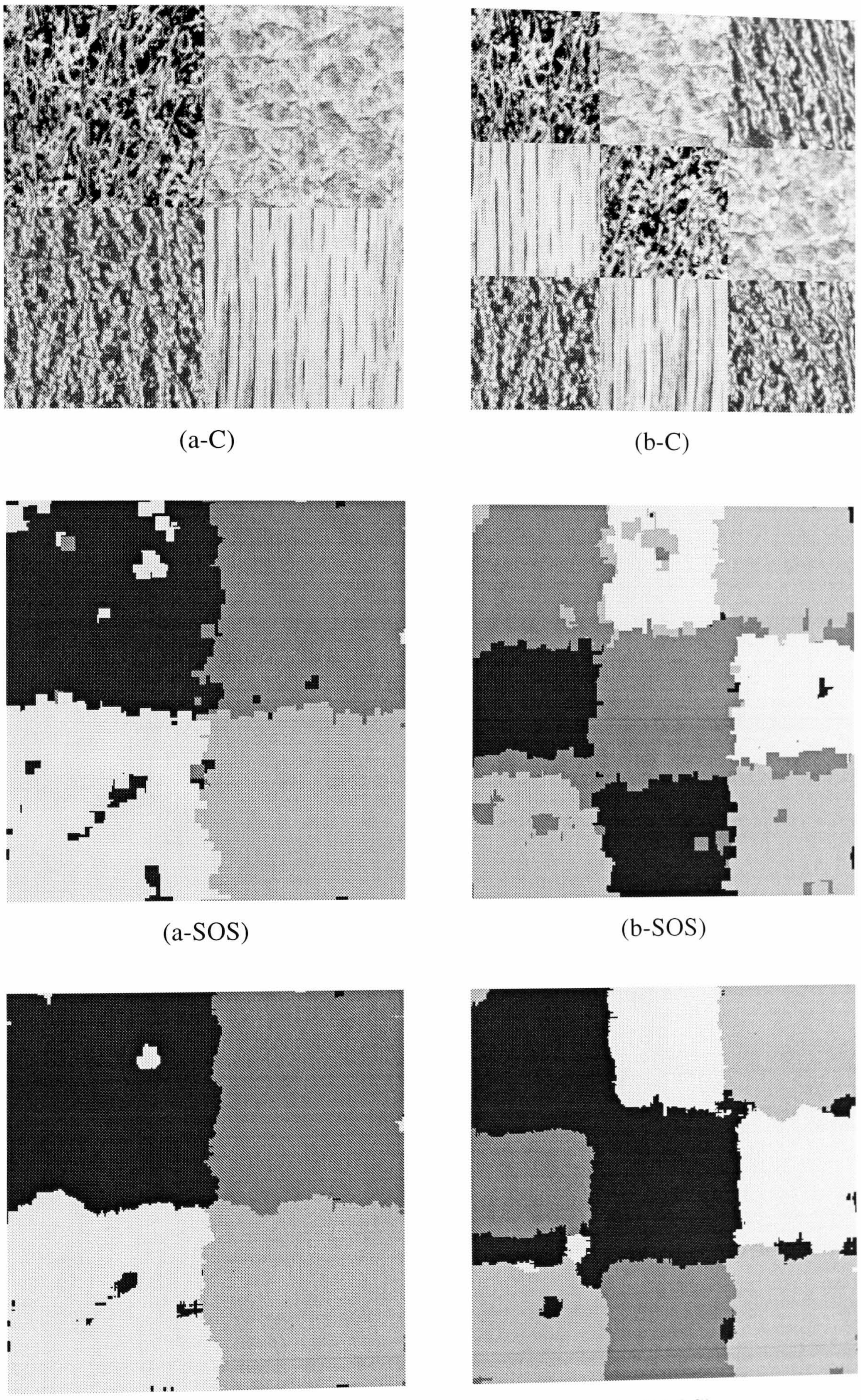


Figure 5.12: The HSOS algorithm on multi-region Brodatz images (256×256). (x-C): Composite images; (x-SOS): First layer outputs; (x-HSOS): Final outputs.

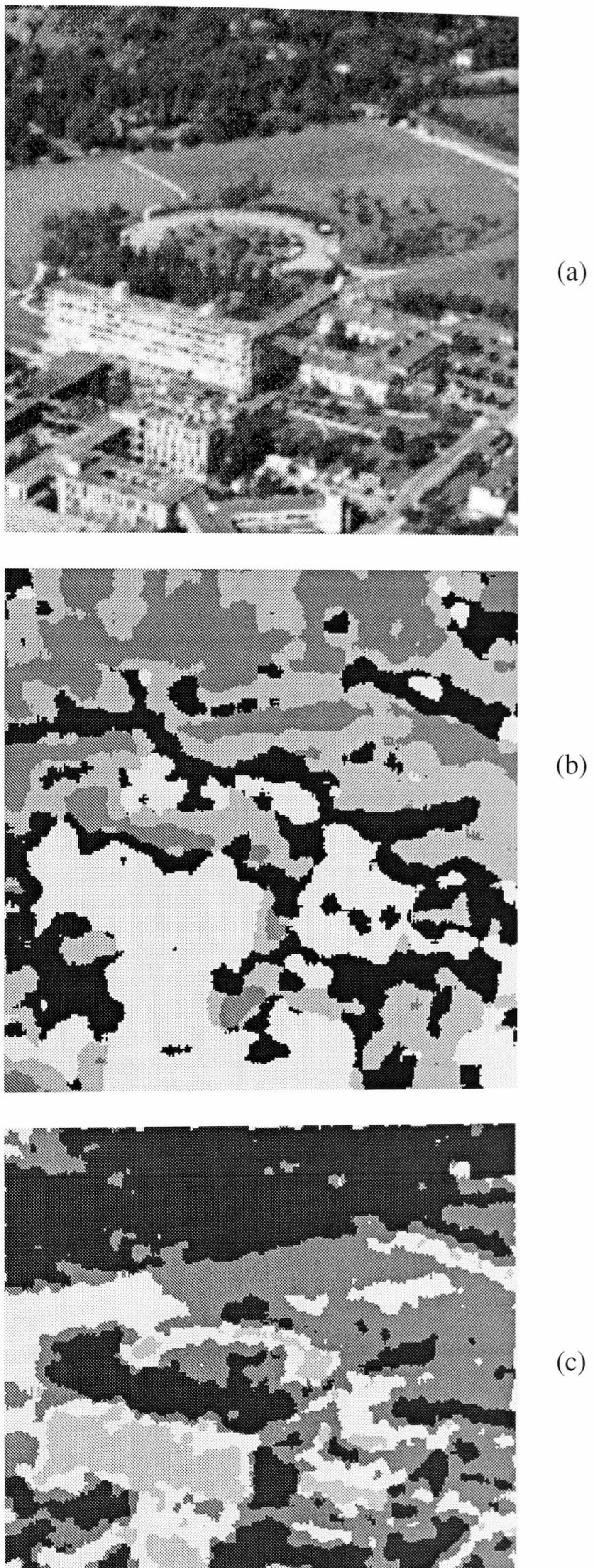


Figure 5.13: The HSOS algorithm on a real image (256×256).
(a) A landscape picture; (b) Final output of the HSOS; (c) HSOS segmentation but with pixel-value-based inputs.

(4) ***Aerial Images***

The HSOS algorithm has also been tested on a typical aerial photograph. Fig. 5.13 (b) shows the (final) segmentation result for the image of Fig. 5.13 (a), which is a picture of a university landscape. Four neurons were used in the SOM chain of the HSOS structure, together with a second-order MRF model. The result shows that the segmentation has converged to four basic meaningful region types: trees, grass, buildings, and roads. This is quite good segmentation for a totally unsupervised system. There are some errors in the segmentation, especially in the tree region. However the building areas have been clearly segregated, as have the roads.

In image processing, there are basically two kinds of inputs: pixel-based representation and feature-based representation. In the previous algorithms, the inputs to the network are the MRF model parameters or features that describe the spatial relationships between neighbouring pixels. Segmentation of such images can also been undertaken directly by using raw pixel values in a window (e.g. 6x6) as the input to the HSOS structure (i.e. without the LS estimator, see Fig. 5.8). The inputs are directly fed to the SOM chain layer, the remainder of the system remains unchanged. Different areas will converge to their corresponding neurons. Since there is no LS estimator, the segmenting speed is much faster. However, in this case, the weight dimensions are increased to 6x6, instead of just six (for a second-order MRF model). Fig. 5.13 (c) shows the result of such an approach. The segmentation has also meaningfully converged to four basic regions. It shows a very good segregation of the tree and grass areas. The road area is also very clear in most places, except for a block area on the left. In general the result is better than that of Fig. 5.13 (b), because in this case regions are not "pure" textures or less texture-based, so image data can be used directly as inputs.

5.4 Towards the Optimal Segmentation

5.4.1 The optimality of the SOS and HSOS algorithms

The SOS algorithm proposed in the last section uses the local properties of textures and optimal estimation and classification properties of neural networks. It considers the segmentation as a classification or clustering problem of various windows placed over the image to be segmented. It groups local windows with similar texture attributes, and labels the blocks with region types. It regards the label of each image pixel as a unknown constant (instead of a random variable). The advantages and disadvantages of this algorithm can be listed below:

Advantages:

* Preattentive and attentive vision based search strategy.
* Simple computational structure.
* Fast convergence (only feedforward process).
* No need to compute the image joint distribution.

Disadvantages:

* Local minimum problem (which also exists for most other algorithms).
* Not globally optimal.
* Some errors at boundaries.

The HSOS algorithm is similar to SOS, but with an extra local voting layer. The principle behind this is that the LV layer can remove noise in the label field, which is considered as unknown constant values. As the estimating windows are often regularly shaped, which will not usually match the actual boundary shapes, so windows may lie across boundaries, and noise is unavoidable in the SOS's labelling stage. The voting vectors in the LV layer of the HSOS structure act like fuzzy membership rules, and produce a median average of a pixel's region label from the results of previous various windows around it. When the noise distribution is symmetric around the true label value, the median is equal to the mean. So the HSOS algorithm will give a better performance especially at boundaries. We have seen this effect in the last section. Further improvement to the boundary classification can achieved by applying a relaxation algorithm, which will be introduced in Section 5.5.

As we have shown in Section 5.2, window-based parameter estimates will be approximately Gaussian distributed around the actual parameter for the whole region. We have modelled this relationship by Eqn. (5.2), i.e. the real parameter with a noise term. In SOS structure, the SOM chain will eventually filter out the noise and converge to the actual parameter set of the texture regions. Therefore, for each texture region, from the algorithm's distance matching law, its region-joint-distribution, or region-joint likelihood function, will be maximised, since

$$\ln p[\mathbf{x}|\Theta_i(\infty)] \approx \ln p(\mathbf{x}|\Theta_i{}^*) > \ln p[\mathbf{x}|\Theta, \forall\Theta \neq \Theta_i(\infty)], \quad \mathbf{x} \in \Omega_i \tag{5.10}$$

under the assumption of well-separated classes, or equal variance and probability for each class. This will lead the SOS and HSOS algorithms to an approximate ML segmentation. However this regional optimum may not lead to a global optimum, as the underlying assumptions may not be satisfied in real problems. So the joint-likelihood function of the whole image may not be maximised. However in this case, even the ideal optimal MAP segmentation will contain a certain amount of segmentation error.

5.4.2 Bayesian SOM for the SOS and HSOS algorithms

For a globally optimal segmentation, the assignment of the estimating window to a label class should be according to its posterior probability instead of a simple Euclidean distance. This is the so-called Bayes law, aimed at minimising classification errors. In window-based segmentation, the problem becomes a mixture distribution (MD) problem. Window estimates are approximately normally distributed in each region class. We have already proposed a Bayesian SOM (BSOM) in Chapter 3 for such problems. When replacing the SOM algorithm by the BSOM, the SOS and HSOS algorithm will become globally optimal algorithms and result in a form of MAP segmentation in the sense of window-based unsupervised texture classifications instead of the whole image sense (from Eqns. (3.41) and (3.19)).

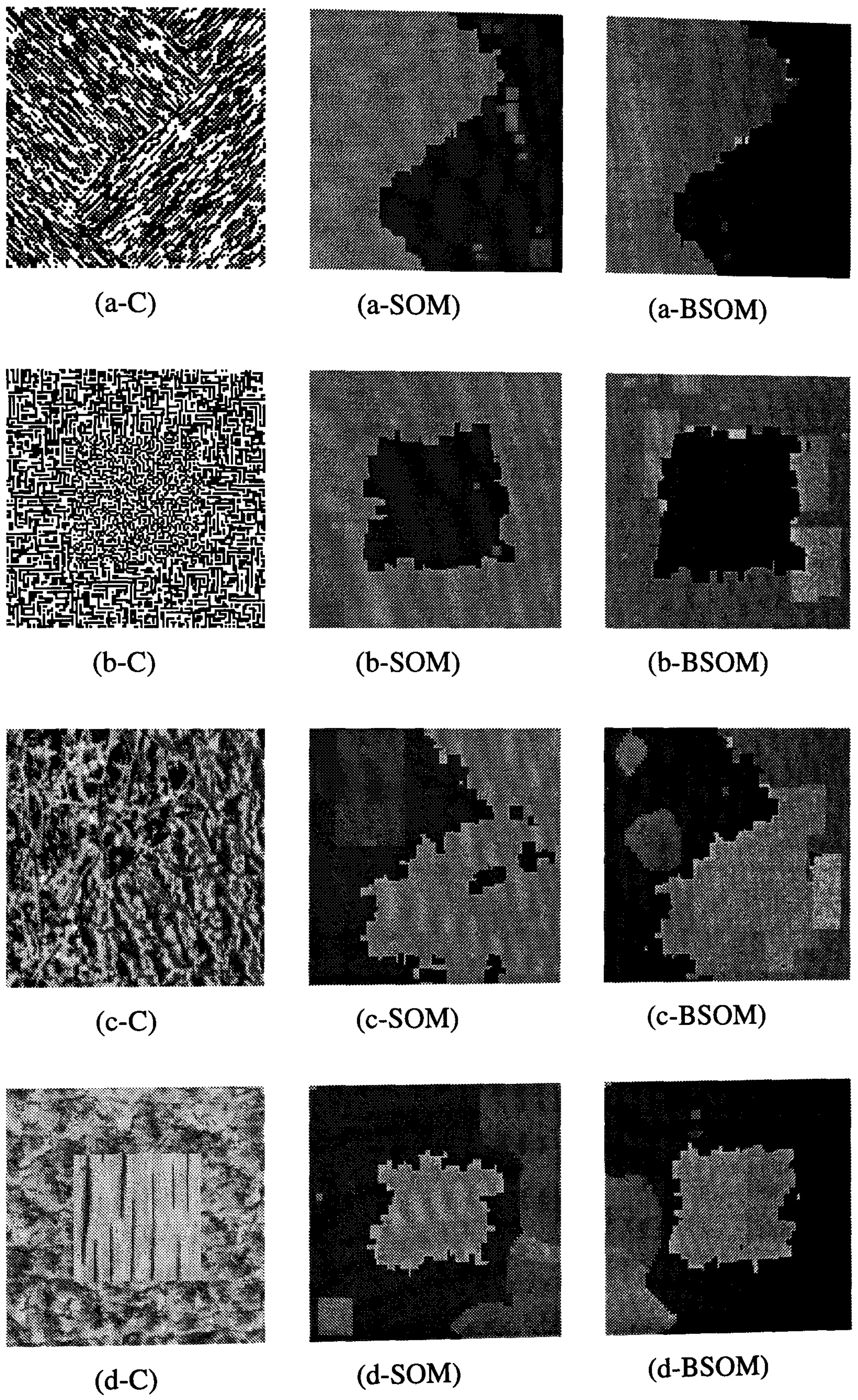


Figure 5.14: Comparison of the SOM and Bayesian SOM in the SOS or HSOS system.
(x-C): Composite textured images (128×128); (x-SOM): Results using SOM; (x-BSOM): Results using Bayesian SOM.

Fig. 5.14 shows some typical examples of applying the BSOM in the SOS network. They are the direct outputs of the SOM chain (i.e. the first layer of the HSOS network). As we can see the results of using the BSOM are better than those using standard SOM, especially when the variances or the sizes of different regions are not the same. An interesting example is shown in Fig. 5.14(a). The results from the SOM and BSOM are both very good, and are almost identical. It would be expected. This is because that the two regions have the same variance and size. In such a case, the BSOM is equivalent to the standard SOM. Although in the HSOS algorithm, the LV layer can clean up the noise and provide a smoother boundary, a better interpretation of sample distributions could be useful.

5.4.3 Towards the common ML and MAP segmentation

In statistical model-based segmentation of textured images, pixel grey levels are modelled as random fields, usually GMRF, while labels of pixels are also considered as a random field. Such doubly random field structures for describing noisy image and textured images were first adopted by Derin and Elliott (1987) and Cohen and Cooper (1987), and have since become very popular. The idea is that neighbouring pixels tend, or are more likely to have, the same region label. The label field is often modelled as a simple first-order GD or GRF.

In such a doubly random field model, the entire random field can be expressed as a hierarchical random field: $Y=[X_{ij}, L_{ij}, (i, j)\in\Omega]$, where $[X_{ij}, (i, j)\in\Omega]$ is the pixel grey level field (intensities) at the lower level, while $[L_{ij}, (i, j)\in\Omega]$ represents the label field at the higher level. Usually at the lower level, the intensity field is described through a conditional GMRF, whose distribution can be expressed as (see Chapter 4)

$$p[X_{ij}=x_{ij} | L_{ij}=l,(i,j)\in\Omega]=\frac{|\mathbf{B}(l)|^{1/2}}{[2\pi\sigma^2(l)]^{M/2}}\exp\{-\frac{1}{2\sigma^2(l)}\mathbf{x}^T\mathbf{B}^T(l)\mathbf{x}\} \qquad (5.11)$$

where $\mathbf{x}$ is a raster scan vector of all pixels whose label belongs to class l.

At the higher level, the label field is often described by a GRF, as the neighbouring pixels tend to have the same label or region class, and the region areas are more likely to have patchy-like blocks rather than spreading pixels. It seems that in natural images the distribution of texture regions can be well described by such a random field. However, some work has shown that man-made scenes (e.g. regular shapes) can also be well represented by such models (Cohen and Cooper 1987, Geman and Graffigne 1986, Lakshmanan and Derin 1989). Such a model can be stated as

$$p[L_{ij}=l,(i,j)\in\Omega]=\frac{1}{Z}\exp\{-U(l)\} \qquad (5.12)$$

where Z is similar to (4.25b), $U(l)$ is the energy function, which is often a function of pair cliques, i.e.

$$U(l)=\sum_{c_i\in C}\sum_{i=1}^{k}V_{c_i}(l) \qquad (5.13a)$$

$$V_{c_i}(l) = -\beta_i \delta[(L_{ij} - L_{mn}) | (i,j),(m,n) \in c_i] \tag{5.13b}$$

where i denote the clique type, and β_i is its corresponding parameter. The number of the clique types, k, depends on the order of the model.

The aim of the segmentation of such doubly random field models is to estimate the label field given a realisation of the multi-texture fields, i.e. choose the **L** which can maximise *a posteriori* (MAP) probability: $P(\mathbf{L}|\mathbf{X}=\mathbf{x})$. The posterior distribution can be expressed from the Bayes rule:

$$p(\mathbf{L}|\mathbf{X}=\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{L})p(\mathbf{L})}{p(\mathbf{x})} \tag{5.14}$$

One difficulty in such segmentation problems is the estimation of the region model parameters. In most algorithms (e.g. Derin and Elliott 1987, Cohen and Cooper 1987, Geman and Graffigne 1986), these parameters, together with the number of regions, are assumed to be known *a priori* and can be pre-estimated from known samples. So the conditional distributions in the numerator in (5.14) can be explicitly expressed by Eqn. (5.11). The configuration which has the largest value for the numerator in (5.14) is the best segmentation. That is, only the numerator term in (5.14), i.e. the joint distribution $p(\mathbf{L},\mathbf{X})$, needs to be compared with different configurations, as the denominator $p(\mathbf{x})$ is fortunately a constant when the input is fixed.

However, there is another difficulty in searching for a MAP configuration, since the search space increases dramatically with image size. For example, for a 64x64 lattice and two texture regions, there are totally $2^{4096}(\approx 10^{1216})$ different configurations. It is impossible to make an exhaustive search. Many methods have been developed for such problems. Usually they assume a random initial configuration or according to some fast, coarse segmentation, find a substitute segmentation either randomly or according to some rules, and then take this substitution of the segmentation if the joint distribution is increased (*deterministic*) or probabilistically according to the ratio of the joint-likelihood of this substitution to the original configuration (*stochastic*). Deterministic methods are generally fast and easy to implement. However they can only guarantee a local minimum. For example, Besag (1986) has proposed an efficient iterated conditional model method for a deterministic, pixel by pixel searching for a MAP segmentation. While stochastic relaxation searching may have a greater probability in finding the global minimum, if the temperature is decreased sufficiently slowly.

For all methods, the search space has to be limited. This can be done by applying relaxation methods to a smaller area rather than the entire image. For example, Derin and Elliott (1987) have applied the MAP segmentation on a strip basis and given a dynamic programming scheme to calculate the joint-likelihood in the strip for optimal MAP segmentation. The actual relaxation is limited to a narrow strip (strip width is from 2 to 4 pixels), so the computation for the search is manageable. For a complete segmentation, strip by strip relaxation is needed. Their method uses pre-estimated model parameters for all regions, and a fixed temperature for relaxation. This also can only achieve local (i.e. strip) MAP segmentation. Cohen and Cooper (1987) developed a parallel hierarchical relaxation for ML segmentation of MRF textured images. The method "splits" region windows into four quadrants, or "merges" region windows of different sizes. So the segmentation is searched in a relatively regular manner and at less computational cost. Since the method does not undertake an exhaustive search and stochastic relaxation, it provides an ML instead of MAP segmentation. This method also uses pre-estimated model parameters. Geman and Graffigne (1986) have used a window-based relaxation method for texture segmentation. They used a window (5x5 pixels), which slides over the image to limit the search space. They also used the pseudo-likelihood in the segmentation and the model parameter estimation, so that computation is even faster. Such simple and straightforward techniques, i.e. sliding window and/or pseudo-likelihood, have been widely used in texture segmentation algorithms (Lakshmanan and Derin 1989; Manjunath *et al.* 1990; Chang and Chatterjee 1992).

Unsupervised segmentation is even more difficult and computationally intensive. Both segmentation and parameter estimates have to be updated according to current parameter estimate and segmentation. Little work has been undertaken on these problems. Lakshmanan and Derin (1989) proposed an adaptive segmentation algorithm for unsupervised segmentation of noisy images which are corrupted by additive independent Gaussian noise. Manjunath and Chellappa (1991) proposed a window and pseudo-likelihood based interactive method for a MAP and maximum posterior marginal (MPM) unsupervised segmentation of textured images. In such problems, the optimal solution should achieve both ML parameter estimation and MAP (or MPM) segmentation.

Recently, the expectation-maximisation (EM) algorithm, proposed by Dempster *et al.* (1977) for solving ML estimation problems with incomplete data, has been applied in MRF model parameter estimation and unsupervised segmentation (Zhang 1992, Zhang *et al.* 1994). Initial model parameters for the regions and segmentation are obtained using a moving window based clustering method, then the parameters are estimated by an iterative EM algorithm. At least a local optimum can be achieved. They proposed several algorithms for independent noise models, MRF models, and doubly MRF models. However, in deriving a practical solvable procedure, many assumptions and simplification (e.g. independence of pixels; Gaussian distributions) had to be made in their methods. Under such assumptions, simpler methods, such as the proposed HSOS (with a boundary relaxation algorithm), will achieve similar performance.

We have shown that the SOS and HSOS algorithms will produce an approximate ML segmentation in the sense that the label field is a constant but unknown field. While with the BSOM, the SOS and HSOS can provide a MAP segmentation (actually an MAP *classification* over window-based inputs). In the sense of a random label field, these algorithms are approximation to a ML or MAP local optimum. Since they treat the window-based inputs as homogeneous, they simply limit the search space to *1 of M* problem. Since the size of the window cannot be reduced without limit, such methods will have some segmentation errors around region boundaries, where more complex configurations for dividing regions are needed. For a doubly random field model, a further relaxation phase is needed to achieve a full ML and MAP segmentation. As the region parameters have already obtained in the SOS or HSOS structure, and the segmentation in the centres of the regions are often noiseless, such a relaxation can easily be implemented and undertaken near region boundaries. In the next section, such a boundary relaxation scheme is proposed.

5.5 Stochastic Relaxation for Improvement on Boundary Classification

5.5.1 Optimal searching by stochastic relaxation schemes

Over the last decade, stochastic relaxation has been applied for combinatorial optimisation problems, where the objective functions have various non-linear relations with very many variables, and a directly deviation is infeasible. The earliest idea was the Metropolis algorithm (Metropolis *et al.* 1953), which simulates the behaviour of a multivariate system in equilibrium at a given temperature. The aim is to develop efficient techniques for finding minimum or maximum values of such a define objective function. Statistical mechanics, the central discipline of condensed matter physics, concerns the behaviour of a system with many degrees of freedom in thermal equilibrium at a finite temperature, and what happens to the system in the limit at low temperatures. Statistical mechanics is characterised by the Boltzmann probability factor, exp($-E/K_BT$), where E denotes the energy of the system, K_B is Boltzmann constant, and T is the absolute temperature. For many physical systems, the low energy states are the most stable and ordered, and so the most desirable states. The Metropolis algorithm applies a small random turbulence to the current state, compares the energies of the original state with the disturbed state. The new state is accepted or rejected according to the energy change, ΔE. If $\Delta E<0$, i.e. the change will bring the system to a lower energy state, this move is accepted. If $\Delta E\geq 0$, the new state is accepted with the probability of exp($-\Delta E/K_BT$).

In finding a good low energy state, the temperature plays an important role. Physicists achieve a low energy state of a substance through *annealing*, i.e. heating then slowly cooling the substance. Introducing temperature and simulated annealing process into combinatorial optimisation is due to Kirkpatrick, who has given a good review of the connection between statistical mechanics and combinatorial optimisation (Kirkpatrick *et al.* 1983). Simulated annealing, also called stochastic relaxation, has been introduced by Geman and Geman (1984) to image processing to achieve an MAP restoration of noisy or corrupted images, and an MAP segmentation of textured images. They developed the Metropolis algorithm and proposed a *Gibbs sampler* stochastic relaxation algorithm. They also derived a theoretical limit to the annealing schedule and proved that under this schedule the relaxation process will converge *in distribution* to the minimal energy configuration. The algorithm includes two steps, one is the *sampling* that generates a new sample at a site according to the local conditional Gibbs distribution, i.e. (see Geman and Geman 1984 for details)

$$p[Y_{ij}\,|\,y_{mn},(m,n)\in\eta_{ij}]=\frac{1}{Z_{ij}}\exp\{-\frac{1}{T}\sum_{(i,j)\in c}V_c(\mathbf{y})\} \tag{5.15}$$

where the partition function Z_{ij} is similar to (4.25b).

The other step is the *annealing* which gradually lowers the temperature from a high point (melting point) to a low degree according to a specified decreasing scheme:

$$i).\ T(t)\xrightarrow{t\to\infty}0,\quad ii).\ T(t)\geq M\Delta/\log t,\ \text{for all } t\geq t_0 \text{ and } t_0\geq 2 \tag{5.16}$$

where M is the size of the image, $\Delta = \max_{\mathbf{y}} U(\mathbf{y}) - \min_{\mathbf{y}} U(\mathbf{y})$

In the above the multi-variable, **y**, can be the joint configuration of a noisy image or a textured image as described in Section 5.4.3. For example in MAP segmentation problem, i.e. (5.14), we only need to compare the ratios for the numerator, i.e. the joint distribution of the observation and label field estimate. Even so the task is not easy, as the search space for the label field is too large. Geman and Geman's Metropolis-like Gibbs sampler algorithm can be used to achieve a MAP resolution. Such algorithms have been applied to image restoration and segmentation problems (Geman and Geman 1984, Geman and Graffigne 1986). However such annealing schedules are too slow to follow in practice. Some applications, as mentioned in Section 5.4.3, use an approximation to the temperature decreasing scheme, or a fast annealing scheme, while some others simply use a clamped temperature. In these cases, a local optimum is guaranteed. For example, Derin and Elliott (1987) have used a fixed temperature in their "dynamic programming algorithm" for obtaining a MAP segmentation of a doubly MRF modelled noisy or textured image. Lakshmanan and Derin (1989) and Manjunath *et al.* (1990) have used a low initial temperature ($T_0 = M\Delta$) in the relaxation scheme in their *adaptive segmentation algorithm* and stochastic network respectively for achieving the MAP segmentation.

5.5.2 A boundary relaxation algorithm

An important property of the HSOS algorithm is that it provides not only the segmentation results but also the parameter estimation results. The segmentation results are the final layer's outputs; while the parameter estimates for each texture region are contained in each neuron's weights. These parameters are very useful in unsupervised segmentation. They can be used to further clear up or filter the noise, and smooth the region boundaries. In this section, a boundary relaxation (BR) algorithm, which makes use of these results, is proposed for incorporating with the HSOS algorithm as a second processing phase.

In the BR algorithm, a relaxation window slides along the region boundaries of the segmentation result from the HSOS algorithm. The size of the relaxation window needs not to be too large, but should contain sufficient texture information. Typical sizes can be from 20×20 pixels (for homogeneous synthetic textures) to 30×30 pixels (for inhomogeneous natural textures). The centre of the window is located at the pixel which is on a current boundary. The placement rule of the relaxation window can be either random or deterministic. In the random fashion, a pixel is randomly picked at a boundary, then the window frames its neighbouring area. In the deterministic fashion, the window slides along every current boundary.

In such a window, Π, shown in Fig. 5.15, the search for a substitute configuration or segmentation is limited to simple linear segments of the window into two regions (extension to more-region linear segments is not difficult). Such a segment can be horizontal, vertical, or any sloping segment crossing the window. The search space also includes the two simplest configurations, that is, the whole window belongs to region type I or II. In our test program, a horizontal or vertical line with a random deviation from the pixel (i, j) is generated as a possible substitute. For example, as shown in Fig. 5.15, ζ^0 represents the current segmentation in the window, ζ^s

represents such a substitute, dotted lines denote other possible substitutes. The joint distribution for the current segmentation can be expressed as

$$p(\Pi,\zeta^o)=p(\Pi|\zeta^o)p(\zeta^o)=p[x_{ij}|\hat{\Theta}_I,(i,j)\in I]p[x_{ij}|\hat{\Theta}_{II},(i,j)\in II]p(\zeta^o) \qquad (5.17)$$

where the I represents one region of the current segmentation in the window, while II represents the other, $\hat{\Theta}_I$ and $\hat{\Theta}_{II}$ are the estimated parameter sets for region I and II respectively.

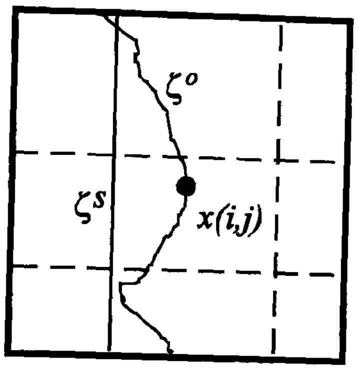


Figure 5.15: The relaxation window.

The joint distribution for the substitute segmentation can be either

$$p(\Pi,\zeta^{s_1})=p(\Pi|\zeta^{s_1})p(\zeta^{s_1})=p[x_{ij}|\hat{\Theta}_I,(i,j)\in\omega_I]p[x_{uv}|\hat{\Theta}_{II},(u,v)\in\omega_{II}]p(\zeta^{s_1}) \qquad (5.18)$$

or

$$p(\Pi,\zeta^{s_2})=p(\Pi|\zeta^{s_2})p(\zeta^{s_2})=p[x_{ij}|\hat{\Theta}_{II},(i,j)\in\omega_I]p[x_{uv}|\hat{\Theta}_I,(u,v)\in\omega_{II}]p(\zeta^{s_2}) \qquad (5.19)$$

where ω_I is one part that the substitute divides the window, e.g. the left part divided by the line ζ^S, while the ω_{II} is the other part, and ζ^{S_1} and ζ^{S_2} represent the two possible configurations (ζ^{S_1}: $\{\hat{\Theta}_I\to\omega_I, \hat{\Theta}_{II}\to\omega_{II}\}$; while ζ^{S_2}: $\{\hat{\Theta}_I\to\omega_{II}, \hat{\Theta}_{II}\to\omega_I\}$) that the substitute may make.

The label field distributions, $p(\zeta^O)$, $p(\zeta^{S_1})$, and $p(\zeta^{S_2})$, can be expressed as Gibbs distributions like Eqns. (5.12) and (5.13). However, we have noticed that the number of pixels on the boundary, both current or substitute, is much less than the total number of pixels in the window. Therefore, these three distributions are approximately the same. This further simplifies the calculation, as the label field distribution forms can be omitted from (5.17)-(5.19). The remainders are just conditional distributions which can be calculated from (5.11), (4.36), the pseudo-likelihood. The algorithm then accepts or rejects the substitute (ζ^{S_1} or ζ^{S_2})according the ratio of the joint distributions, i.e.

$$\text{Prob}\{\text{substitute: }\zeta^{Si}\}=\begin{cases}1, & \text{if } [\max\limits_{i=1,2} p(\Pi,\zeta^{s_i})]/p(\Pi,\zeta^o)\geq 1,\\ [\max\limits_{i=1,2} p(\Pi,\zeta^{s_i})]/p(\Pi,\zeta^o), & \text{otherwise}\end{cases} \qquad (5.20$$

The substitute is then fed into the LV layer of the HSOS structure and makes contributions to the voting vectors in the window area. After a number of iterations, the relaxation algorithm together with the HSOS algorithm (its LV layer) will yield accurate and smooth boundaries.

This is a Metropolis version of stochastic relaxation. The temperature can be introduced, and the relaxation can be fully implemented as described in the previous section. However we have found that this simple relaxation algorithm works very well with the HSOS structure.

5.5.3 Experimental results

We have implemented this boundary relaxation method with the HSOS structure. As we have seen from the above analysis, the calculation of the joint distribution in a window is simple when the GMRF model is used. So the speed of the relaxation phase is fast. When we further examine the HSOS structure, we have found that the calculation for this relaxation phase can be further simplified. The HSOS uses the local LS estimation for the parameters, so the HSOS will finally obtain the MMSE model parameters. This implicitly employs the independent GMRF model for textures. Such a model is the most commonly used model because of its simplicity. Many existing unsupervised segmentation algorithms (e.g. Besag 1986; Lakshmanan and Derin 1989; Manjunath and Chellappa 1991; and Zhang *et al.* 1994), although start with very general cases, but settle with this simplest model.

So why not directly use the MMSE principle in distribution or energy calculations? This can simplify the energy function to a mean-square-error function. The LS estimates consider the parameters that can minimise the total errors. In other words, when the pixels in an image are fitted to the correct parameters, the total description errors will be the smallest (This can make a short cut to avoid the full computations in calculating the joint likelihood function, and also leads to the MMSE validation method in the next section).

Here we express the local energy function in a window, Π, as the total mean-square-error of description:

$$U(\Pi) = \sum_{(i,j)\in\Pi, lab(i,j)=l} [x_{ij} - m^{(l)} - \sum_{(u,v)\in\eta_{ij}} \theta_{uv}^{(l)} (x_{uv} - m^{(l)})]^2 \qquad (5.21)$$

where $m^{(l)}$ and $\theta^{(l)}$ are the model's mean and parameters for region l which the pixel belongs to.

In the case where the window may contain more than one region, as in the boundary relaxation algorithm, different segmentation and different region models need to be fitted for a minimum error search. The BR algorithm simplifies this procedure by generating a limited number of possible segmentations of the window into just two regions, then the segment which can produce the minimum energy (errors) will be the new segmentation within the window (otherwise, the substitution is according to the energy difference). The BR algorithm can be implemented in parallel as shown in Fig. 5.16.

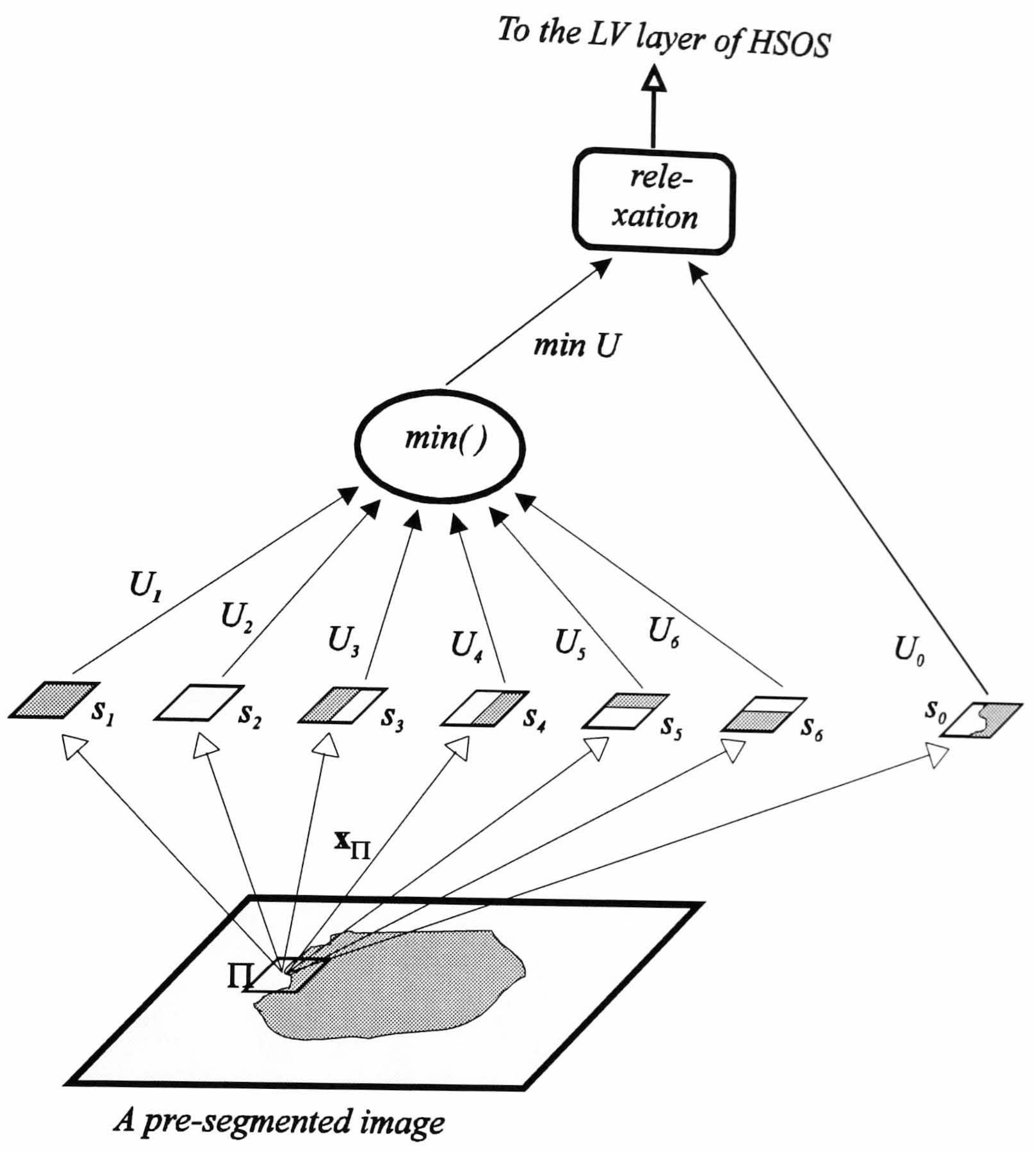


Figure 5.16: The boundary relaxation algorithm.

When the window is located at a boundary, the algorithm generates a possible vertical and horizontal segment, each of which has two possible region labelling: s_3 and s_4 are the two possible combinations for the vertical segment; while s_5 and s_6 are the two possible results for the horizontal segment. The s_1 and s_2 are the possible results for labelling the entire window to region I and II respectively. The s_0 is the original segmentation in the window. The algorithm then calculates the corresponding energy function for these possible segmentations: U_1, U_2,... U_6, and U_0, and compares the six substitute energies, and chooses the substitute with the lowest energy. The chosen substitute is then compared with the original segmentation's energy. The substitution is accepted if its energy is lower than the original one, or according to the ratio of these two energies. The result is fed into the LV layer of the HSOS structure. The computation for the energy is relatively straightforward. The speed of the BR algorithm is very fast.

Some typical results of applying this simple version of the BR algorithm for 10 iterations to the HSOS results are shown in Figs. 5.17 and 5.18. The HSOS results used here are after only 1,000 iterations of the HSOS algorithm, much less than the previous results shown in Figs. 5.9 and 5.9. However the BR algorithm has successfully removed the noise around the boundaries. After applying the BR algorithm, the segmentation is almost identical to the real boundaries. Remember that only vertical and horizontal substitutes are generated for these examples. More

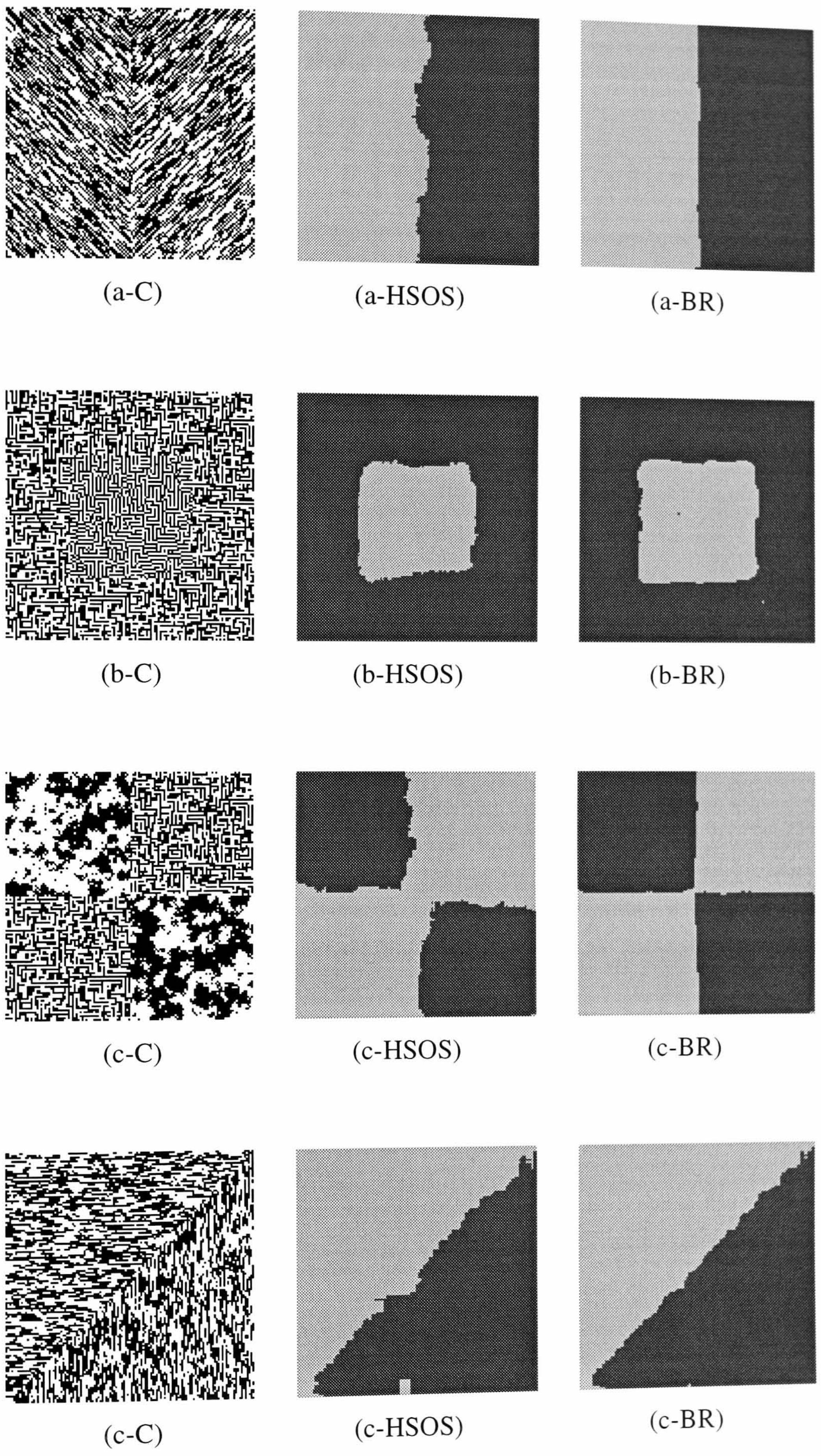


Figure 5.17: The segmentation results of the HSOS and BR algorithms for synthetic textures.
(x-C): Composite images (128×128); (x-HSOS): Results of HSOS; (x-BR): Results after applying the boundary relaxation.

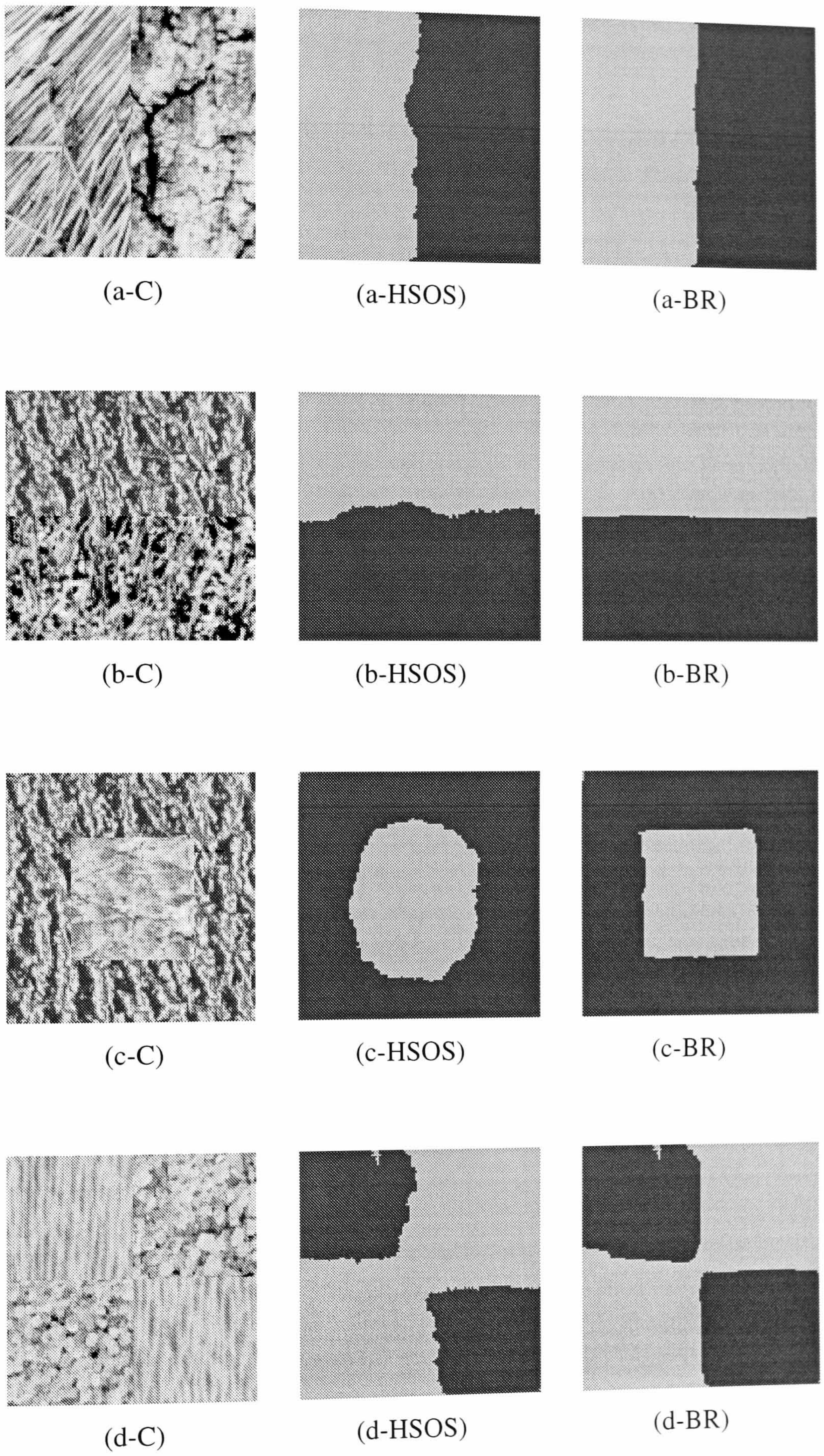


Figure 5.18: The segmentation results of the HSOS and BR algorithms for Brodatz textures.
(x-C): Composite images (128×128); (x-HSOS): Results of the HSOS; and (x-BR): Results after applying the boundary relaxation.

complicated substitutes such as, angled-line segments, curve segments, can be implemented, and the speed of the BR will not be affected too much because of its parallel nature.

5.6 MMSE Validation for Unsupervised Segmentation

5.6.1 Clustering validation

In unsupervised textured image segmentation, known class samples are not available and the model parameters for each texture have to be estimated on-line during the segmentation process. However, the number of region classes in the image is often assumed to be known in most applications. Such an assumption is often reasonable, because the number of clusters is rather a *subjective* than a *objective* matter, except for those cases where the clusters are well separated and within-class distributions are well behaved. In many circumstances, however, there is a need to determine the number of region classes, that is, the number of clusters itself is a variable. This requirement can be understood as a question of how many *meaningful* regions (for some purpose) can the image be grouped into, or how many *homogeneous* texture regions does the image contain. It is the fundamental problem of *cluster validity*, and is essentially (or explicitly) unsolved. In the general *data clustering* domain, validation refers to the objective assessment of a clustering structure to determine whether a structure is meaningful, useful, or can be well interpreted (Jain and Dubes 1988). Validation is accomplished by carefully applying statistical methods and testing hypotheses.

In the unsupervised segmentation of textured images, a common approach to the validation problem is to derive a criterion function and to repeat the clustering procedure for various numbers of clusters, e.g. K=1, 2, 3, ..., and to examine the criterion function. It is very difficult to design a criterion function that can give a maximum or minimum answer to the correct number, K^*, of clusters. However a meaningful function will normally monotonically increase or decrease with K and give a noticeable change at the correct number. For example, when the total squared-error is chosen as the criterion function, it will decrease rapidly with K until $K=K^*$, then it will decrease much more slowly until it reaches zero at $K=M$, the number of data points (pixels in this case) (Duda and Hart 1973). When the log joint-likelihood is used as the criterion function, it exhibits a rising exponential behaviour as a function of K, and the best class number can be selected at 90% of the rise (rounded to the nearest integer) according to Langan *et al.* (1994). The log joint-likelihood function is the Q-function in their EM algorithm for image segmentation (Zhang *et al.* 1994).

As the joint-likelihood is a monotonic function with K, most validation criteria amend the function by adding a penalty term to limit the likelihood function's continuous growth above the correct number. Some commonly used criteria are briefly described below.

(1) *Akaike's Information Criterion (AIC)*

The criterion is used to select the model order for an autoregressive (AR) process, and is defined as (Akaike 1974):

$$\hat{K} = \arg \min_{1 \le K \le K_{\max}} \{-\log p(\mathbf{x}|\Theta) + k\} \tag{5.22}$$

where k is defined as the number of *free* parameters in the K-class model.

As we can see the inverse joint-likelihood function decreases monotonically with K (rapidly before the correct class number, and more slowly after that number), while k always increases monotonically with K. A minimum can be expected.

(2) *Minimum Description Length Criterion (MDL)*

The minimum description length (MDL) criterion was also developed to find the best model order for AR processes, and to overcome the asymptotically biased problem of the AIC criterion. It is expressed as below (Rissanen 1984):

$$\hat{K} = \arg \min_{1 \le K \le K_{\max}} \{-\log p(\mathbf{x}|\Theta) + \frac{1}{2} k \log M\} \tag{5.23}$$

where k is defined as the number of free parameters assuming K clusters, and M is the number of data points.

The MDL criterion takes into consideration the size of the data by using it as a weight in the penalty term.

(3) *Merhav, Gutman, Ziv Criterion (MGZ)*

The MGZ criterion (Merhav, Gutman, and Ziv, 1989) is based on the MDL criterion, but limits its attention to Markov sources and exponential distributions. Given that the true number exists, it balances the overestimation and underestimation probability through a parameter, λ, in an extended Neyman-Pearson sense. It is stated as:

$$\hat{K} = \arg \min_{1 \le K \le K_{\max}} \{K | \frac{1}{M}[MDL(K) - MDL(K_{\max})] \le \lambda\} \tag{5.24}$$

where $\lambda > 0$.

5.6.2 MMSE validation for GMRF model-based segmentation

As we stated in the last section, in most segmentation algorithms as well as our HSOS algorithm, the model parameters are obtained by the LS (MMSE) or MPL method, applied to a homogeneous texture. Two methods are equivalent and all assume that the pixels are independent. Within this domain, the calculation for validation criterion can be made much easier. There will be no need to compute the joint likelihood function, which in most application is computationally extensive. Instead, the total mean-squared-error (MSE), similar to (5.21) but extended to whole image Ω and all categories (1 to K), is calculated, i.e.

$$MSE(K) = \sum_{(i,j)\in\Omega, l=lab(i,j)} [x_{ij} - m^{(l)} - \sum_{(u,v)\in\eta_{ij}} \theta_{uv}^{(l)}(x_{ij} - m^{(l)})]^2 \tag{5.25}$$

where $l=label(i, j)$ is the output of the LV layer of the HSOS network, $\{m^{(l)}, \theta^{(l)}\}$ are corresponding model parameters for region class l.

Ideally if each texture region is *very homogeneous*, segmentation with the correct number of clusters will have the minimum MSE (MMSE), because each region will give less errors for the prediction difference if fitted to its LS parameters rather than any other parameters. We have seen that this idea was successfully used in the previous section's boundary relaxation algorithm. This is why we term it the MMSE validation. Ideally the MSE function will decrease with K and then reach a (local) minimum at K^*. The MSE function will change (increase or decrease) little for $K>K^*$. The MSE function will be slightly increased if the over-numbered clusters disperse on region boundaries, or will be slightly decreased if the over-numbered regions split some regions.

However, when the texture regions are not *very homogeneous* as it is usually the case, the problem may return to the general case as we have mentioned earlier, i.e. the MSE will decrease rapidly with K until $K=K^*$, then it will decrease much more slowly until it reaches zero at $K=M$. There should be still a significant change in slope, or even a local minimum, at $K=K^*$. Even when a over-numbered cluster splits one assumed inhomogeneous texture region, the change of MSE function at $K=K^*$ should be noticeable, otherwise this split may be correct (i.e. the inhomogeneous region may be interpreted better as two regions instead of one). One advantage of using MSE is its simple computational form. The computation can be incorporated into the segmentation process. There are two ways to calculate the total MSE. One method is to use the definition of (5.25) with the model parameters obtained by HSOS network, when the segmentation process halts. The other is to use each region's variance parameter obtained on-line in HSOS algorithm, together with region size values (the number of pixels of each region), which can be easily obtained in segmentation), and the total MSE can be easily approximated to

$$MSE'(K) = \sum_{1 \le l \le K} \hat{\sigma}^2(l) M(l) \tag{5.26}$$

where is the estimated variance parameter for region l, and $M(l)$ is the total number of pixels in region l.

This MMSE validation idea may also be useful for validation of the segmentation result of the HSOS network for a globally correct segmentation even when the number of clusters is provided. If the segmentation process converges to an incorrect segmentation (e.g. one class merges two textures, or some two split one texture region), the MSE function will always be higher than that for the correct segmentation.

5.6.3 Experimental results

The MMSE validation scheme has been applied to the HSOS network by using different SOM sizes to the same composite image, and Eqn. (5.25) has been used to calculate the corresponding total MSE values.

The results are shown in Figs. 5.19-5.22, which are 256x256 pixels. The composite images consist of four different texture regions. Results shown in the figures are the LV layer outputs of the HSOS network for the cluster number K=2, 3, 4, 5 (sometimes

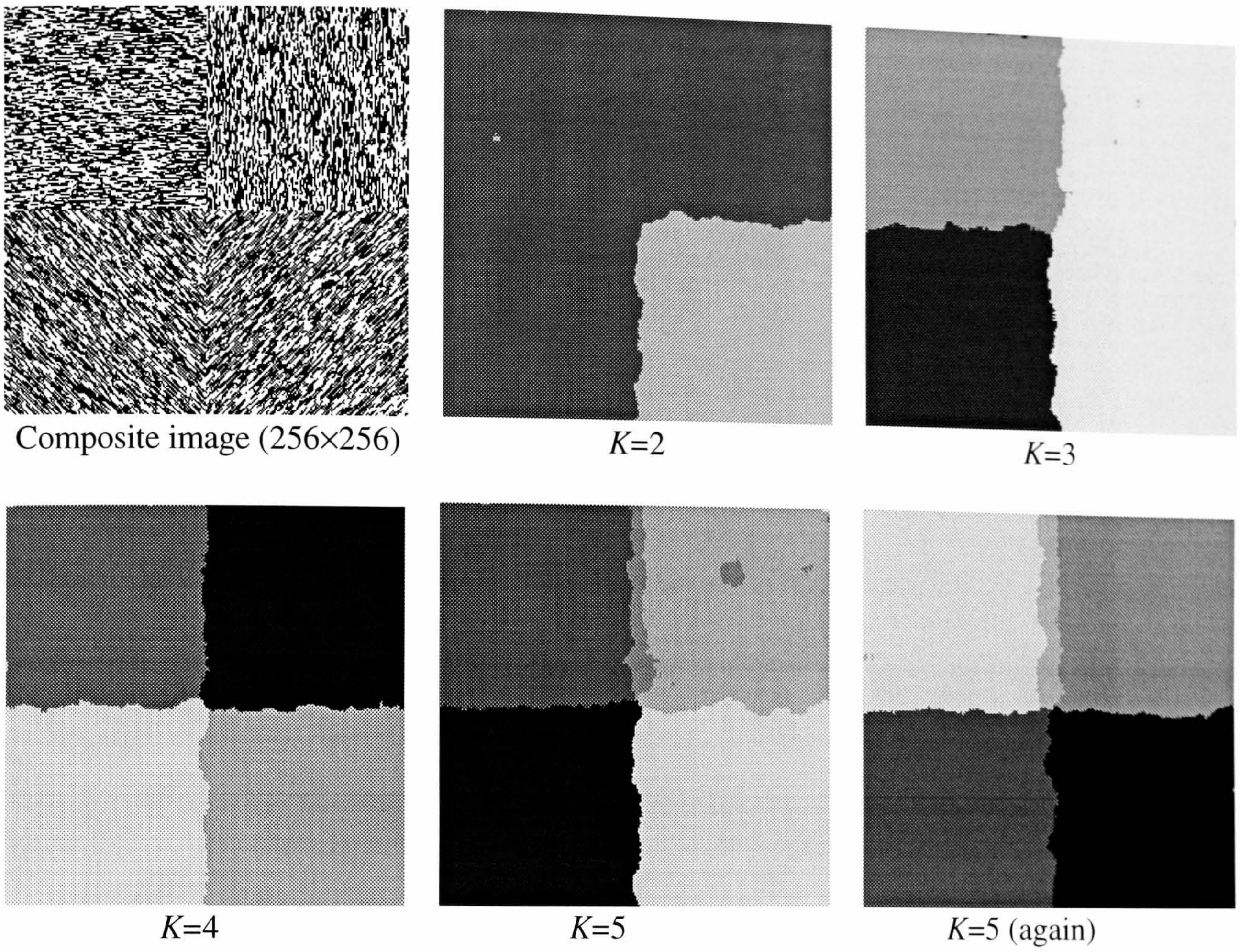


Figure 5.19: MMSE validation of unsupervised segmentation of synthetic textures.

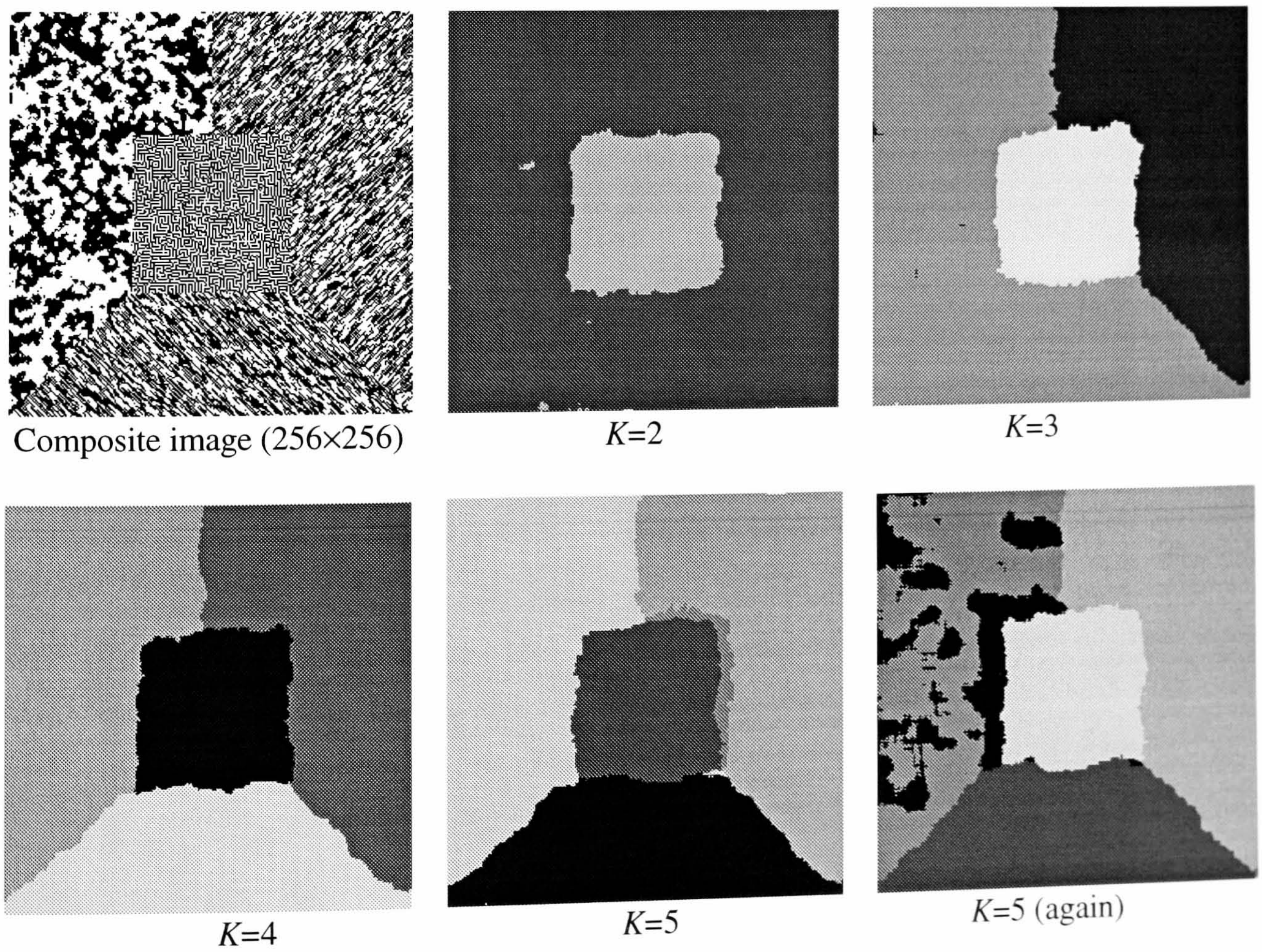


Figure 5.20: MMSE validation of unsupervised segmentation of synthetic textures.

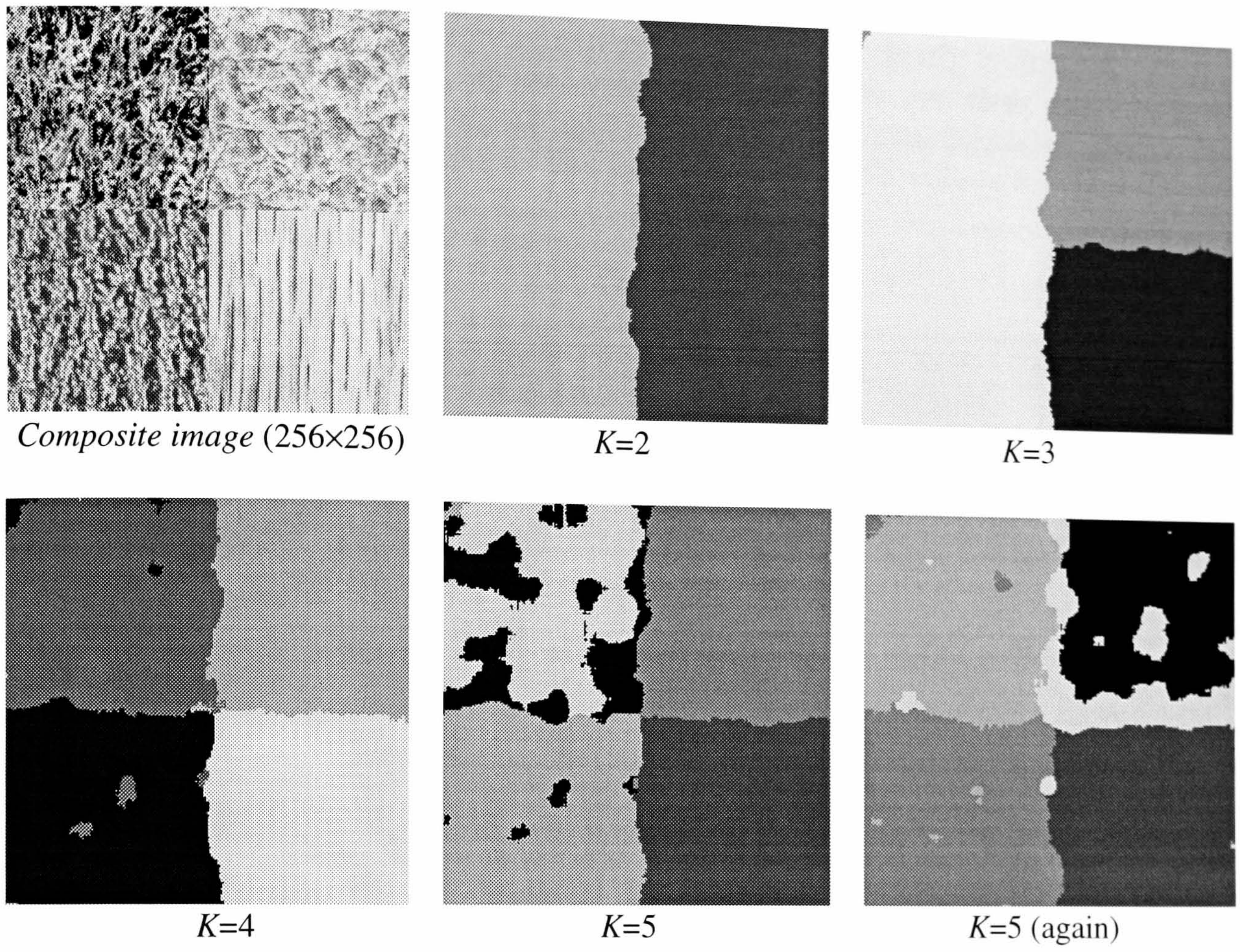


Figure 5.21: MMSE validation of unsupervised segmentation of natural textures.

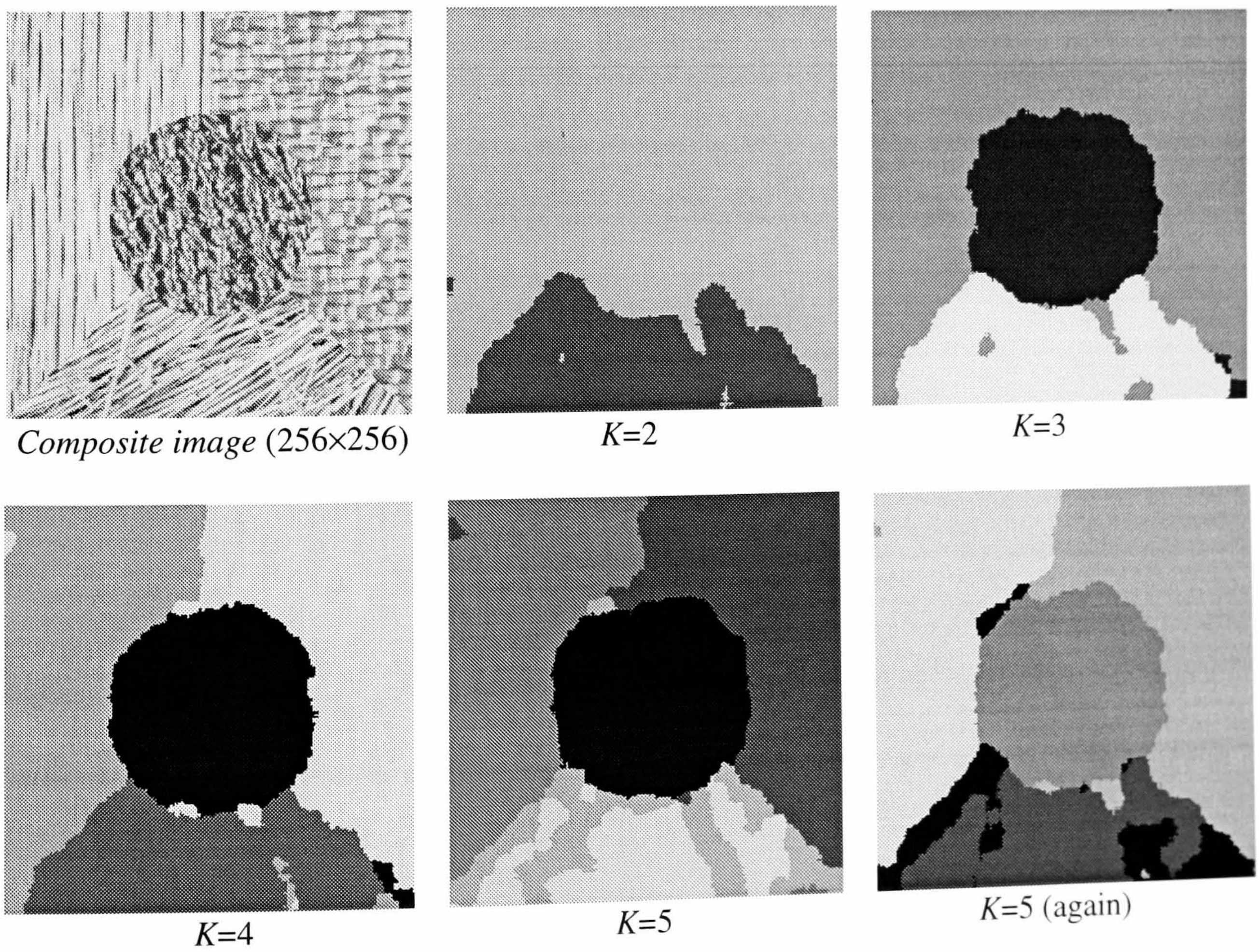


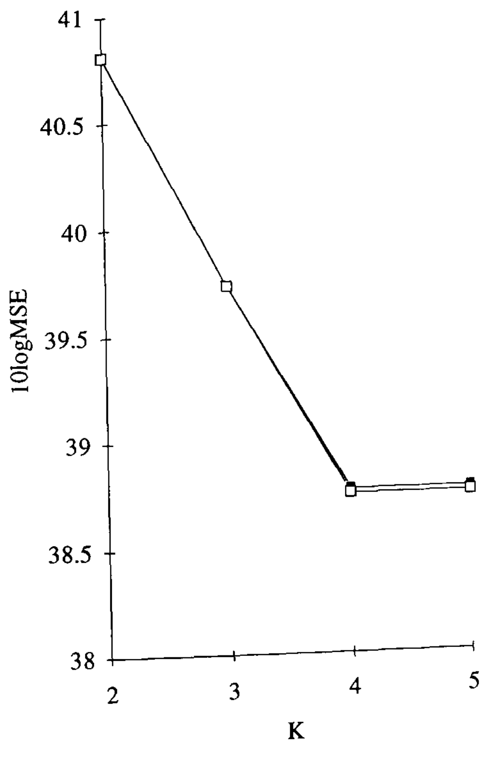
Figure 5.22: MMSE validation of unsupervised segmentation of natural textures.

two results of *K*=5 are given for comparison). The figures clearly indicate the correct segmentation. When *K*=2, 3, 4, the segmentation results are clear and compact. However, when K=5, the additional neuron can only take some boundary pixels, and may be in several separate places. The corresponding MSE values for these figures are given in Tables 5.4.

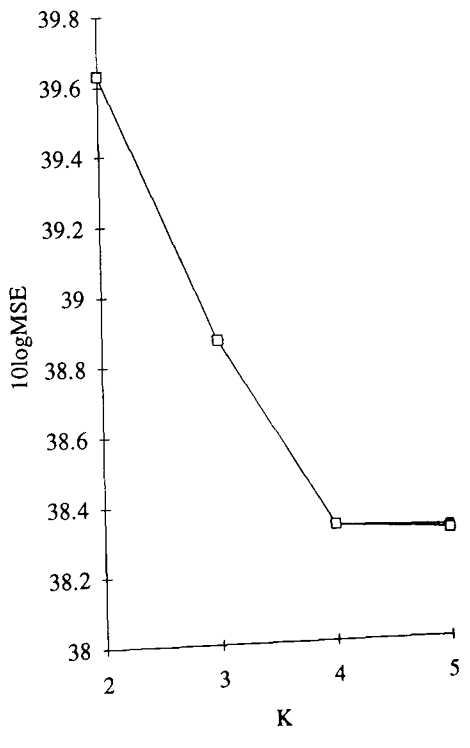
MSE	*K*=2	*K*=3	*K**=4	*K*=5	*K*=5
Fig. 5.19	12041.11	9405.68	7505.74	7533.55	7493.14
Fig. 5.20	9193.86	7719.68	6815.25	6799.31	6768.75
Fig. 5.21	8906144.60	8830534.86	8534303.70	8538921.72	8528535.83
Fig. 5.22	6158596.90	5823209.02	5572463.36	5543407.18	5527261.47

Table 5.4: MSE values for various numbers of clusters in Fig. 5.19-5.22.

To visualise these values and examine their fitness for the validation, they have been put into dB form (10 log MSE) and shown in Fig. 5.23 and Fig. 5.24. In each figure, there is a very distinct change at the correct number of clusters (4 in all these examples). Prior to this number the MSE functions decrease very quickly, but after this number, the MSE function is almost unchanged or decreased much more slowly. These results support the theoretical analysis for the proposed MMSE validation, and show the usefulness of this method when used together with the HSOS network.



(a)



(b)

Figure 5.23: MSE functions. (a) For Fig. 5.19; (b) For Fig. 5.20.

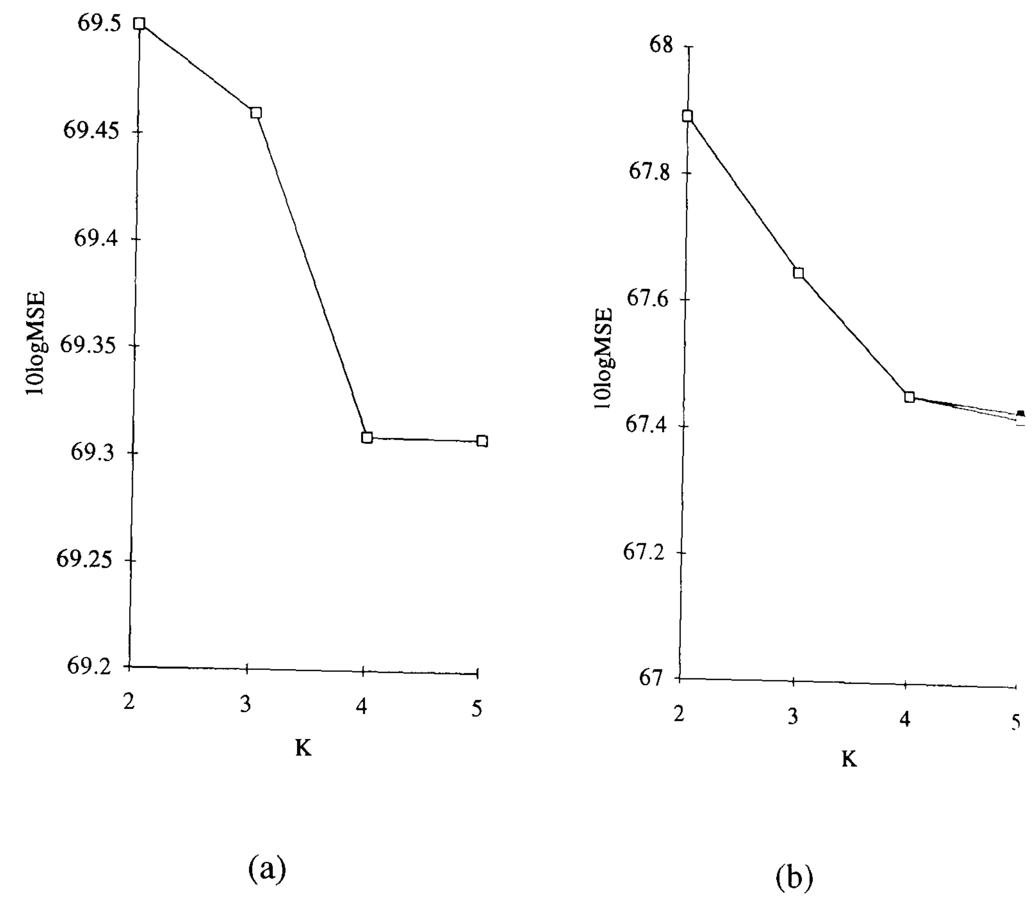


Figure 5.24: MSE functions. (a) For Fig. 5.21; (b) For Fig. 5.22.

5.7 Conclusions

In this chapter, several practical methods, namely, the HSOS network, the BR algorithm, and MMSE validation method, for the unsupervised segmentation of textured images, have been proposed. A step by step approach of both theoretical analysis and discussion, together with experimental verification, has been adopted. These algorithms make full use of the local properties of textures, efficient representation by the MRF model, and the SOM's convergence properties and simplicity. They have been proved to be optimal in some sense, and efficient in implementation and operation.

Chapter 6

CONCLUSIONS

6.1 Summary and Conclusions

This thesis has presented a detailed statistical analysis and treatment of Kohonen's self-organising map (SOM) algorithm, especially in respect of its convergence, feature distribution, and its potential optimality for two major practical applications: vector quantisation (VQ) and pattern classification. The unsupervised segmentation of textured images has also been investigated. Several effective segmenting structures combining the SOM algorithm and Markov random field (MRF) models have been proposed and examined through a step by step analysis of their viability. Significant and novel results are, the formal proof of the general convergence of the algorithm; the asymptotical Gaussian distribution of its feature space; the diminishing effect of the initial weights; the SOM's potential optimum performance for VQs; an optimal SOM-VQ algorithm; an optimal SOM classification algorithm; an practical combination of SOMs and other methods for texture segmentation; a simple yet effective class-number validation method. The major contributions and results are summarised as below:

1. *A proof has been obtained for the general convergence of the SOM algorithm, and its potential optimality for VQs in the sense of minimum mean-square distortion.*

This thesis has shown that the convergence of the SOM algorithm to the centroids of input pattern subsets exists for any dimensional mapping. The algorithm asymptotically satisfies the two necessary conditions for minimising the mean-square distortion between the representing vectors, i.e. *features*, and the data samples.

2. *A comparative study of LBG, CL and SOM -VQs has been presented and a constrained SOM algorithm for an improved VQ has been proposed.*

When the distortion surface is not a single concave, the problem can become very subtle, and most VQ algorithms, such as LBG-, CL-, and SOM- VQs, can only guarantee a local minimum in distortion. However, we have found that the neighbourhood function of the SOM has a significant influence on the local minima problems. For example, it can naturally overcome the under-utilisation problem, and can also assist in escaping from some local minima. A constrained SOM algorithm, based on the "equidistortion principle", has been proposed for guiding the SOM to a global or a good local minimum. Experimental results have shown an improved performance through using this constrained algorithm over standard LBG, CL, and SOM -VQ algorithms.

3. *The thesis has found the diminishing influence of the initial state, and has also given an exploration of the complicated impact of learning rates and neighbourhood functions on convergence and ordering.*

The initial weights will have little quantitative impact on the convergence of the map, as long as the learning rate satisfies the convergence conditions. However forming a globally ordered or a good locally ordered map has been found to rely mainly on the neighbourhood function in terms of its extent and shrinking speed, although the learning rate also has some influence. These parameters can only be chosen empirically with the current understanding of the SOM. They also affect the convergence speed. Slower learning rates, within the convergence conditions, will generally result in slower convergence speeds. However, slow learning rates, together with appropriate shrinking speeds of the neighbourhood function, are helpful for forming the global or a improved local minimum in final distortion and more importantly a better topological ordering. Our study has also emphasised that the choice of an exponential fall for learning rates is not correct and may result in an inaccurate mapping.

4. *A Bayesian SOM has been proposed for unsupervised pattern classification in order to achieve a maximum a posteriori performance.*

The SOM algorithm has been widely applied to the pattern classification and data clustering problems because of its computational simplicity and its ability to avoid the "empty-class" problem. Our study has shown, however, the algorithm is approximately optimal only when classes are compact and well-separated. For most other cases, the SOM needs to be modified in order to achieve the best performance. A general unsupervised pattern classification problem can be described as a mixture distribution problem. An extended SOM, termed the Bayesian SOM, has been proposed for such an application. The theoretical analysis and experimental results have supported this approach. This algorithm should find application in many practical classification problems.

5. *Clear definitions and measures of topological ordering and a meaningful analysis of the inherent fault-tolerance ability of ordered mappings have been presented.*

Two constructive definitions of topological order, one in geometrical sense, the other in mean-square distortion sense, have been formally put in mathematical forms. The second definition can provide a very useful explanation of the fault-tolerance ability of an ordered map. The measuring metrics for the ordering have been discussed. It has been found naturally that the ordering measure can only be meaningful in terms of the ordering definition.

6. A review of approaches to texture processing has been given in this thesis.

The most commonly used descriptions and approaches of textures has been systematically reviewed. The model-based approaches, mainly MRFs, have been discussed fully in terms of their usefulness, simplicity, and effectiveness. Various methods for the parameter estimation of MRFs have been described.

7. A hierarchical self-organised segmentation structure for textured images has been proposed, and it is shown that this structure can achieve a maximum likelihood or maximum a posteriori segmentation.

The unsupervised segmentation of textured images has been investigated as a possible application domain of the SOM algorithm. A direct self-organising segmentation structure and a hierarchical self-organising segmentation structure, which consists of a SOM (or a Bayesian SOM), a local voting network (a simplified SOM) and local MRF parameter estimator, have been proposed through theoretical analysis and extensive experiment.

8. A simple and fast relaxation algorithm has been proposed for improving the segmentation accuracy at region boundaries.

This is a local mean-square error energy comparative relaxation algorithm, which tracks the boundaries obtained from the segmentation algorithm. The search space in the relaxation algorithm has been limited simply to the original boundary and several regular alternatives within a small moving window. The segmentation results can be considerably refined by applying this in conjunction with proposed segmentation structures.

9. A minimum mean-square-error validation method has been found useful for validating the correct class-number in a totally unsupervised segmentation of textured images.

An on-line simple, classical, but applicable validation method has been used successfully for testing the correct class number in a fully unsupervised segmentation problem. This method exploits the homogeneity within each single texture region, tests the fitness of the estimated models in terms of the total mean-square-error. It can be incorporated within proposed segmentation structures.

6.2 Future Work

A great deal of research on the SOM neural network and texture image processing has already undertaken by many researchers. This thesis has made certain contributions to these two subjects, and it has also opened discussion on some further and deeper concepts, such as the role of the neighbourhood functions in optimal convergence and the ordering process; the noise-tolerance capability of ordered maps; a possible application for the proposed Bayesian SOM in kernel smoothing method for function approximation; practical training considerations of the SOM algorithm; further development in MRF model parameter estimation; and a possible link between MRF model-based approaches and multichannel filter-based approaches. These points for future investigation and improvement are noted briefly below:

- *More advanced on-line validation and verification methods for the SOM algorithm are needed in order to "tune" the algorithm to form improved mappings, i.e. in a lowest possible distortion and/or a highest possible order.*

The SOM is a local learning algorithm, which locally minimises the mean-square distortion but aims at achieving a globally low distortion. How to judge the quality of the training process as it progresses, especially on-line, would be very useful.

- *The detailed quantitative role of the neighbourhood function on the ordering process needs to be examined formally.*

Two important roles of the neighbourhood function, which have been stressed in this thesis, are in assisting the escape from some inappropriate local minima and forming a good topological ordering. However, such roles have not yet been analysed functionally or quantitatively. The possible utilisation of the neighbourhood function in other mapping requirements, such as exact density matching, may also be worth investigating.

- *Further exploration of the importance of the fault-tolerance ability of an ordered SOM mapping for VQ, associative memory, and other application, is needed.*

Fault-tolerance is one of important generic features of neural networks, and is a fundamental principle in associative memories. As we have previously stated that a good ordered map can provide greater fault-tolerance than a poorly ordered or disordered map. Such an ability will be very useful in many other areas, such as function approximation, pattern classification, and high dimensional data visualisation.

- *A practical strategy for fast training and/or training with a small number of data samples is urgently required.*

Theoretically, the SOM requires infinite iterations to converge, and this means each data sample has to be input many times (to eliminate the unequal contributions of samples input in different times). To overcome this inefficiency, a short-period training method (e.g. transformation-like), if possible, would be very useful and could make the SOM more practical. Ideally, the SOM also requires a large amount of data for the training. However, in many practical applications, obtaining large volumes of data may be very difficult or even impossible. The behaviour of the SOM in reduced data situations needs to be investigated.

- *Re-learning ability needs to be added in the SOM.*

One of greatest differences between neural network approaches and other non-neural network approaches is the former's ability to learn and re-learn. There is apparently no theoretical analysis of this problem for the SOM neural network. Can a trained map be re-trained when more new data samples are available, while preserving the existing ordering?

- *The function approximation ability of the SOM algorithm has not been explored in depth.*

The difference between VQs and function approximations is that VQs use representing vectors, while function approximations use basis functions. When the weight vectors, i.e. points, are extended to a basis function centred at these points, some similar properties between these two approaches may arise. The feasibility and suitability of the SOM and related algorithms, such as self-organised principle component analysis, to the function approximation problem need to be fully investigated.

- *It is worth investigating a possible link between model-based and multifilter-based approaches to image texture description and analysis.*

Currently these two methods exist in parallel in texture-related image processing. Multifilter-based approaches have also been found useful in some other image processing problems, while model-based approaches seem merely applicable for textured, or homogeneous, images. The relationship between these two could be very close at least in the discrimination of image textures .

- *Parameter estimation for Gibbs distributions needs further study.*

As we have pointed out, the optimal maximum likelihood estimation for model parameters is currently untractable. Most existing estimates use least-square or maximum pseudo-likelihood approaches instead, and this requires the assumption of independence between the pixels. Parameter estimation of MRFs or GDs has long been, and still remains, a challenging topic in statistics.

- *The continuing exploration of application potential of neural networks in general image segmentation, scene understanding, and object recognition.*

This could be a life-long topic, but it is one of most important and eternal objectives of neural network research. The need to understand information, especially visual information, processing mechanism, and so to model the process by means of neural networks has increased greatly as more and more disciplines such as computer vision, robotics, psychology, physiology, and cognitive science, have become actively involved. Developments in neural computing in theses fields are expected to continue. Information processing based on neural computing will become a new and popular computation method.

References

Abbas, H.M. and Fahmy, M.M., 1994, "Neural networks for maximum likelihood clustering," *Signal Processing,* Vol. 36, pp. 111-126.

Ahalt, S.C., Krishnamurthy, A.K., Chen, P., and Melton, D.E., 1990, "Competitive learning algorithms for vector quantization," *Neural Networks,* Vol. 3, pp. 277-290.

Akaike, H., 1974, "A new look at the statistical model identification," *IEEE Trans. on Automatic Control,* Vol. 19, No. 6, pp. 716-723.

Allinson, N.M., 1990, "Self-organising maps and their applications," In *Theory and Applications of Neural networks,* J.G. Taylor and C.L.T. Mannion, eds., pp. 110-118, London: Springer-Verlag.

Amari, S.-I., 1967, "A theory of adaptive pattern classifiers," *IEEE Trans. on Electronic Computers,* Vol. 16, pp. 299-307.

Amari, S.-I., 1972, "Learning patterns and pattern sequences by self-organizing nets of threshold elements," *IEEE Trans. Computers,* vol. 21, pp. 1197-1206.

Amari, S.-I., 1980, "Topographic organization of nerve fields," *Bulletin of Mathematical Biology,* Vol. 42, pp. 339-364.

Amari, S.-I., 1982, "A mathematical theory of self-organizing nerve systems," In *Biomathematics: Current status and Perspectives,* J. Metzler and M. Arbib eds., pp. 159-197, Amsterdam: North-Holland.

Amari, S.-I., 1983, "Field theory of self-organizing neural nets," *IEEE Trans. Systems, Man, and Cybernetics,* Vol. 13, pp. 741-748.

Amari, S.-I and Takeuchi, M., 1978, "Mathematical theory on formation of category detecting nerve cells," *Biological Cybernetics,* Vol. 29, pp. 127-136.

Anderson, J.A., 1968, "A memory storage model utilizing spatial correlation functions," *Kybernetik,* Vol. 5, pp. 113-119.

Andrew, L.L.H. and Palaniswami, M., 1994, "A study on the effect of neighbourhood functions for noise robust VQs," *Proc. IEEE ICNN-94,* Vol. 6, pp. 4159-4162.

Ballard, D. H. and Brown, C. M., 1982, *Computer Vision.* London: Prentice-Hall Inc.

Bauer, H.-U. and Pawelzik, K.P., 1992, "Quantifying the neighbourhood preservation of self-organizing feature maps," *IEEE Trans. Neural Networks,* Vol. 3, No. 4, pp. 570-579.

Besag, J., 1974, "Spatial interaction and the statistical analysis of lattice systems," *Journ. Roy. Statist. Soc.* **B**, Vol. 36. pp. 192-236.

Besag, J., 1975, "Statistical analysis of non-lattice data," *The Statistician,* Vol. 24, pp. 179-195.

Besag, J., 1986, "On the statistical analysis of dirty pictures," *Journ. Roy. Statist. Soc.* **B**, Vol. 48, No. 3, pp. 259-302.

Bienenstock, E.L., Cooper, L.N., and Munro, P.W., 1982, "Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex," *Journal of Neuroscience,* Vol. 2, pp. 32-48.

Bouman, C. and Liu B., 1991, "Multiple resolution segmentation of textured images," *IEEE Trans. Pattern Anal. Machine Intell.,* Vol. 13, pp. 99-113.

Bouman, C. and Sauer, K., 1993, "A generalized Gaussian image model for edge-preserving MAP estimation," *IEEE Trans. Image Processing*, Vol. 2, pp. 296-310.

Bovik, A, Clark, M., and Geisler, W., 1990, "Multichannel texture analysis using localized spatial filters," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 12, pp. 55-73.

Brodatz, P., 1966, *Texture: A Photographic Album for Artists and Designers*, Toronto: Dover Publish. Co.

Budinich, M. and Taylor, J.G., 1995, "On the ordering conditions for self-organizing maps," *Neural Computation*, Vol. 7, No. 2, pp. 284-289.

Carrato, S., 1994, "Image vector quantization using ordered codebooks: properties and applications," *Signal Processing*, Vol. 40, pp. 87-103.

Chang, C. and Chatterjee, S., 1992, "A hybrid approach toward model-based texture segmentation," *Patter Recognition*, Vol. 25, No. 5, pp. 519-531.

Chellappa, R., Kashyap, R.L., and Manjunath, B.S., 1993, "Chapter 2.2: Model-based texture segmentation and classification," in *Handbook of Pattern Recognition and Computer Vision*, C.H. Chen, L.F. Pau, and P.S.P. Wang Eds., pp. 277-310 Singapore: World Scientific Publishing Company.

Chen, J.-L. and Kundu, A., 1994, "Rotation and gray scale transform invariant texture identification using wavelet decomposition and hidden Markov model," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 16, pp. 208-214.

Chen, O.T.-C., Sheu, B.J., and Fang, W.-C., 1994, "Image compression using self-organization networks," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 4, No. 5, pp. 480-489.

Chinrungrueng, C. and Sequin, C.H., 1995, "Optimal adaptive *k*-means algorithm with dynamic adjustment of learning rate," *IEEE Trans. Neural Networks*, Vol. 6, No. 1, pp. 157-169.

Cho, C.-M. and Don, H.-S., 1991, "A parallel Kalman algorithm for fast learning of multilayer neural networks," *Proc. Int. Joint Conf. Neural Networks (IJCNN)-91* (Singapore), pp. 2044-2049.

Chow, Y.S. and Teicher, H., 1978, *Probability Theory: Independence, Interchangeability and Martinggales*, London: Springer-Verlag.

Cohen, F.S. and Cooper, D.B., 1987, "Simple parallel hierarchical and relaxation algorithms for segmenting noncausal Markovian random fields," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 9, pp. 195-219.

Cohen, F.S., Fan, Z., and Patel, M.A., 1991, "Classification of rotated and scaled textured images using Gaussian Markov field model," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 13, pp. 192-202.

Comer, M.L. and Delp, E.J., 1994, "Parameter estimation and segmentation of noisy or textured images using the EM algorithm and MPM estimation," in *Proc. IEEE Intern. Conf. Image Processing*, Vol. II, pp. 650-654.

Cottrell, M. and Fort, J.C., 1986, "A stochastic model of retinotopy: a self-organizing process," *Biological Cybernetics*, Vol. 53, pp. 405-411.

Cross, G.R. and Jain, A.K., 1983, "Markov random field texture models," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 5, pp. 25-39.

Daugman, J., 1985, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *Journ. Opt. Soc. Amer. A*, Vol. 2, No. 7, pp. 1160-1169.

Dempster, A.P., Laird, N.M., and Rubin, D.B., 1977, "Maximum likelihood from incomplete data via the EM algorithm," *Journ. of Roy. Statist. Soc.* **B**, Vol. 39, pp. 1-38.

Der, R., 1993, "Dynamics of self-organized feature mapping," in *New Tends in Neural Computation*, J. Mira, et al. eds., pp. 312-315, Berlin: Springer-Verlag.

Derin, H., 1987, "Estimating components of univariate Gaussian mixtures using Prony's method," *IEEE Trans. Pattern Anal. and Machine Intell.*, Vol. 9, No. 1, pp. 142-148.

Derin, H. and Elliott, H., 1987, "Modelling and segmentation of noisy and textured images using Gibbs random fields," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 9, pp. 39-55.

Dersch, D.R. and Taven, P., 1995, "Asymptotic level density in topological feature maps," *IEEE Trans. Neural Networks*, Vol. 6, No. 1, pp. 230-236.

DeSieno, D., 1988, "Adding a conscience to competitive learning," in *Proc. IEEE Intern. Conf. on Neural Networks* (San Diego, CA), vol. 1, pp. 117-124.

Dony, R.D. and Haykin, S., 1994, "Self-organizing segmentor and feature extractor," in *Proc. IEEE Intern. Conf. Image Processing (ICIP-94)*, Vol. III, pp. 898-902.

Dony, R.D. and Haykin, S., 1995, "Neural network approaches to image compression," *Proc. IEEE*, Vol. 83, No. 2, pp. 288-303.

Duda, R.O. and Hart, P.E., 1973, *Pattern Classification and Scene Analysis*, New York: Wiley.

Dunn, D., Higgins, W.E., and Wakeley, J., 1994, "Texture segmentation using 2-D Gabor elementary functions," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 16, pp. 130-149.

Erwin, E., Obermayer, K., and Schulten, K., 1991, "Convergence properties of self-organizing maps," in *Artificial Neural Networks*. T. Kohonen, *et al.* eds., pp. 409-414, Amsterdam: Elsevier.

Erwin, E., Obermayer, K., and Schulten, K., 1992a, "Self-organizing maps: ordering, convergence properties and energy functions," *Biological Cybernetics*, Vol. 67, pp. 47-55.

Erwin, E., Obermayer, K., and Schulten, K., 1992b, "Self-organizing maps: stationary states, metastability and convergence rate," *Biological Cybernetics*, Vol. 67, pp. 35-45.

Everitt, B.S. and Hand, D.J., 1981, *Finite Mixture Distributions*, London: Chapman and Hall.

Fogel, I. and Sagi, D., 1989, "Gabor filters as texture discrimination," *Biological Cybernetics*, Vol. 61, pp. 103-113.

Fu, L., 1994, *Neural Networks in Computer Intelligence*, New York: McGraw-Hill.

Fukushima, K., 1975, "Cognitron: A self-organizing multilayered neural network," *Biological Cybernetics*, Vol. 20, pp. 121-136.

Gagalowicz, A., 1981, "A new method for texture fields synthesis: some applications to the study of human vision," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 3, no. 5, pp. 520-533.

Gaze, R.M., 1970, *The Information of Nerve Connections*, London: Academic Press.

Geman, S. and Geman, D., 1984, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. and Machine Intell.*, Vol. 6, pp. 721-741.

Geman, S. and Graffigne, C., 1986, "Markov random field image models and their applications to computer vision," in *Proc. Intern. Congr. Math.*, pp. 1496-1517.

Gersho, A., 1979, "Asymptotically optimal block quantization," *IEEE Trans. Information Theory*, Vol. 25, pp. 373-380.

Gersho, A. and Cuperman, V., 1983, "Vector quantization: A pattern-matching technique for speech coding," *IEEE Communication Magazine*, Vol. 21, pp. 15-21.

Gersho, A. and Gray, R.M., 1992, *Vector Quantization and Signal Compression*, Boston: Kluwer Academic.

Gonzalez, R.C. and Woods, R.E., 1992, *Digital Image Processing*, Reading, Massachusetts: Addison-Wesley Publishing Company.

Gray, R.M., 1984, "Vector quantization," *IEEE ASSP Magazine*, Vol. 1, April, pp. 4-29.

Grossberg, S., 1976a, "Adaptive pattern classification and universal recording: I. Parallel development and coding of neural feature detectors," *Biological Cybernetics*, Vol. 23, pp. 121-134.

Grossberg, S., 1976b, "Adaptive pattern classification and universal recording: II. Feedback expectation, olfaction, illusions," *Biological Cybernetics*, Vol. 23, pp. 187-202.

Grossberg, S., 1987, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Science*, Vol. 11, pp. 23-63.

Guerin-Dugue, A and Palagi, P.M., 1994, "Texture segmentation using pyramidal Gabor function and self organising feature maps," *Neural Processing Letters*, Vol. 1, No. 1, pp. 25-29.

Gurelli, M.I. and Onural, L., 1994, "On a parameter estimation method for Gibbs-Markov random fields," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 16, pp. 424-430.

Haralick, R.M., 1979, "Statistical and structural Approaches to Texture," *Proc. IEEE*, Vol. 67, No. 5, pp. 786-804.

Haralick, R.M., 1986, "Statistical image texture analysis" in *Handbook of Pattern Recognition and Image Processing*, T.Y. Young and K.-S. Fu eds., pp. 247-279, New York: Academic Press.

Haralick, R.M. and Shapiro, L.G., 1992, *Computer and Robot Vision*, Reading, Massachusetts: Addison-Wesley Publishing Company.

Haykin, S., 1994, *Neural Networks: A Comprehensive Foundation*, New York: IEEE Macmillan College Publishing.

Hebb, D., 1949, *Organization Behavior*, New York: John Wiley & Sons.

Hecht-Nielsen, R., 1988, "Neuralcomputing: picking the human brain," *IEEE Spectrum*, Vol. 25, No. 3, pp. 36-41.

Hecht-Nielsen, R., 1990, *Neurocomputing*, Reading, MA: Addison-Wesley.

Hopfield, J.J., 1984, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nation. Acad. Sci. USA*, Vol. 81, pp. 3088-3092.

Hostetter, G.H., 1987, "Chapter 13: Recursive estimation," in *Handbook of Digital Signal Processing*, D.F. Elliott ed., New York: Academic Press, Inc.

Jain, A.K. and Dubes, R.C., 1988, *Algorithms for Clustering Data*, Englewood Cliffs, New Jersey: Prentice Hall.

Jain, A.K. and Farrokhnia, F., 1991, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition*, Vol. 23, pp. 1167-1186.

Julesz, B., 1962, "Visual pattern discrimination," *IRE Trans. Inform. Theory*, Vol. 8, pp. 84-93.

Julesz, B., 1975, "Experiments in the visual perception of textures," *Scientific American*, Vol. 232, pp. 34-43.

Julesz, B., 1981, "Textons, the elements of texture perception, and their interactions," *Nature*, Vol. 290, pp. 91-97.

Julesz, B. and Bergen, J.R., 1983, "Textons, the fundamental elements in preattentive vision and perception of textures," *Bell Sys. Tech. Journ.*, Vol. 62, No. 6, pp. 1619-1645.

Kangas, J., 1995, "Increasing the error tolerance in transmission of vector quantized images by self-organizing map," in *Proc. ICANN-95*, Vol. 1, pp. 287-291.

Kangas, J.A., Kohonen, T., and Laaksonen, J.T., 1990, "Variants of self-organizing maps," *IEEE Trans. Neural Networks*, Vol. 1, No. 1, pp. 93-99.

Kashyap, R.L., 1986, "Image models," in *Handbook of Pattern Recognition and Image Processing*, T.Y. Young and K.-S. Fu eds., pp. 281-310, New York: Academic Press.

Kashyap, R.L. and Chellappa, R., 1983, "Estimation and choice of neighbors in spatial-interaction models of images," *IEEE Trans. Inform. Theory*, Vol. 29, pp. 60-72.

Kashyap, R.L., Chellappa, R. and Khotanzad, A., 1982, "Texture classification using features derived from random field models," *Pattern Recognition Letters*, Vol. 1, pp. 43-50.

Kashyap, R.L. and Khotanzad, A., 1986, "A model-based method for ratation invariant texture classification," *IEEE Trans. Pattern Anal. Machine Intell.* Vol. 8, No. 4, pp. 472-481.

Khotanzad, A. and Kashyap, R.L., 1987, "Feature selection for texture recognition based on image synthesis," *IEEE Trans. Sys., Man, Cybern.*, Vol. 17, pp. 1087-1095.

Kim, Y.K. and Ra, J.B., 1995, "Adaptive learning method in self-organizing map for edge preserving vector quantization," *IEEE Trans. Neural Networks*, Vol. 6, No. 1, pp. 278-280.

Kinderman, R. and Snell, J.L., 1980, *Markov Random Fields and Their Applications*, London: Providence.

Kirkpatrick, S., Gelatt, Jr. C.D., and Vecchi, M.P., 1983, "Optimization by simulated annealing," *Science*, Vol. 220, No. 4598, pp. 671-680.

Kirvida, L., 1976, "Texture measurements for the automatic classification of imagery," *IEEE Trans. Electromagnetic Compatibility*, Vol. 18, No. 2, pp. 38-42.

Kohonen, T., 1972, "Correlation matrix memory," *IEEE Trans. Computers*, Vol. 21, pp. 353-359.

Kohonen, T., 1973, "A new model for randomly organized associative memory," *Intern. Journ. of Neuroscience*, Vol. 5, pp. 27-29.

Kohonen, T., 1974, "An adaptive associative memory principle," *IEEE Trans. Computers*, Vol. 23, pp. 444-445.

Kohonen, T., 1980, *Content-Addressable Memories*, Heidelberg: Springer-Verlag.

Kohonen, T., 1982, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, Vol. 43, pp. 59-69.

Kohonen, T., 1984, *Self-Organization and Associative Memory*, Berlin: Springer-Verlag.

Kohonen, T., 1986, "Representation of sensory information in self-organizing feature maps, and relation of these maps to distributed memory networks," *Proc. SPIE*, Vol. 634, pp. 248-259.

Kohonen, T., 1987, "Adaptive, associative, and self-organizing functions in neural computing," *Applied Optics*, Vol. 26, No. 23, pp. 4910-4918.

Kohonen, T., 1988, "An introduction to neural computing", *Neural Networks*, Vol. 1, pp. 3-16.

Kohonen, T., 1990, "The self-organizing map," *Proc. IEEE*, Vol. 78, No. 9, pp. 1464-1480.

Kohonen, T., 1991, "Self-organizing maps: optimization approaches," in *Artificial Neural Networks*, T. Kohonen, K. Makisara, O. Simula, and J. Kangas eds., Vol. 2, pp. 981-990, North-Holland: Elsevier Science Publishers.

Kohonen, T., 1994, "What generalizations of the self-organizing map make sense?" in *Proc. ICANN-94*, Vol. 1, pp. 292-297.

Kosko, B., 1991, "Stochastic competitive learning," *IEEE Trans. Neural Networks*, Vol. 2, No. 5, pp. 522-529.

Lakshmanan, S. and Derin, H., 1989, "Simultaneous parameter estimation and segmentation of Gibbs random fields using simulated annealing," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 11, pp. 799-813.

Lampinen, J. and Oja, E., 1989, "Self-organizing maps for spatial and temporal AR models," in *Proc. the 6th Scandinavian Conference on Image Analysis*, pp. 120-127.

Langan, D., Modestino, J.W, and Zhang, J., 1994, "Cluster validation for unsupervised stochastic model-based image segmentation," in *Proc. IEEE Intern. Conf. on Image Processing* (*ICIP*-94), Austin, TX, USA, Vol. 2, pp. 197-201.

Linde, Y., Buzo, A., and Gray, R.M., 1980, "An algorithm for vector quantizer design," *IEEE Trans. Communications*, Vol. 28, No. 1, pp. 84-95.

Lippmann, R.P., 1987, "An introduction to computing with neural networks," *IEEE ASSP Magazine*, Vol. 4, pp. 4-22.

Liu, J. and Yang, Y.-H., 1994, "Multiresolution color image segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 16, pp. 689-700.

Lloyd, S.P. 1957. "Least squares quantization in PCM's," Bell Telephone Laboratories memorandum; also in, 1982, *IEEE Trans. Inform. Theory*, Vol. 28, pp. 129-137.

Lo, Z.P., and Bavarian, B., 1991, "On the rate of convergence in topology preserving neural network," *Biological Cybernetics*, Vol. 65, pp. 55-63.

Luckman, A.J., Allinson, N.M., Ellis, A.W., and Flude, B.M., 1995, "Familiar face recognition: A comparative study of a connectionist model and human performance," *Neurocomputing*, Vol. 7, pp. 3-27.

Luttrell, S.P., 1989a, "Hierarchical self-organising networks," in *Proc. IEE Intern. Conf. Artif. Neural Networks*. pp. 2-6.

Luttrell, S.P., 1989b, "Self-organisation: A derivation from first principle of a class of learning algorithms," in *Proc. 3rd. IEEE Intern. Joint Conf. Neural Networks*, Washington, DC, Vol. 2, pp. 495-498.

Luttrell, S.P. 1991, "Code vector density in topographic mappings: scalar case," *IEEE Trans. Neural Networks*, Vol. 2, No. 4, pp. 427-436.

Luttrell, S.P., 1994a, "A Bayesian analysis of self-organizing maps," *Neural Computation*, Vol. 6, No. 5, pp. 767-794.

Luttrell, S.P., 1994b, "Partitioned mixture distribution: an adaptive Bayesian network for low-level image processing," *IEE Proc. Vision, Image, and Signal Processing*, Vol. 141, No. 4, pp. 251-260.

MacQueen, J.B., 1967, "Some methods for classification and analysis of multivariate observations, " in *Proc. Fifth Berkeley Symposium on Math., Stat., and Prob.*, Vol. 1, pp. 281-296.

Makhoul, J., Roucos, S., and Gish, H., 1985, "Vector quantization in speech coding," *Proc. IEEE*, Vol. 73, No. 11, pp. 1551-1588.

Malik, J. and Perona, P., 1990, "Preattentive texture discrimination with early vision mechanisms," *Journ. Opt. Soc. Am. A*, Vol. 7, No. 5, pp. 923-932.

Mallat, S.G., 1989, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 11, pp. 674-693.

von der Malsburg, C., 1973, "Self-organization of orientation sensitive cells in the striate cortex," *Kybernetik*, Vol. 14, pp. 85-100.

Manjunath, B.S. and Chellappa, R., 1991, "Unsupervised texture segmentation using markov random field models," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 13, pp. 478-482.

Manjunath, B.S. and Chellappa, R., 1993, "A unified approach to boundary perception: edges, textures, and illusory contours," *IEEE Trans. Neural Networks.*, Vol. 4, pp. 96-108.

Manjunath, B.S., Simchony, T., and Chellappa, R., 1990, "Stochastic and deterministic networks for texture segmentation," *IEEE Trans. Acoust., Speech, and Signal Processing,* Vol. 38, pp. 1039-1049.

Mao, J. and Jain, A.K., 1992, "Texture classification and segmentation using multiresolution simultaneous autoregressive models," *Pattern Recognition,* Vol. 25, No. 2, pp. 173-188.

Mao, J. and Jain, A.K., 1996, "A self-organizing network for hyperellipsoidal clustering (HEC)," *IEEE Trans. Neural Networks,* Vol. 7, No. 1, pp. 16-29.

Marr, D., 1969, "A theory of cerebellar cortex," *Journ. of Physiology, London,* Vol. 202, pp. 437-470.

Marr, D., 1982, *Vision,* San Francisco: W. H. Freeman and Company.

Marroquin, J.L., 1989, "Chapter 3: A probabilistic approach to computational vision," in *Image Understanding 1989,* S. Ullman and W. Richards eds., pp. 44-79, Norwood, New Jersey: Ablex Publishing Corporation.

McCulloch, W.S. and Pitts, W., 1943, "A logical calculus of the ideas immanent in nervous activity," *Bulletin Math. Biophysics,* Vol. 5, pp. 115-133.

Mendelson, B., 1990, *Introduction to Topology,* 3rd Edition, New York, Dover Publications, Inc.

Merhav, N., Gutman, M., and Ziv, J., 1989, "On the estimation of the model order of a Markov chain and universal data compression," *IEEE Trans. Inform. Theory,* Vol. 35, No. 5, pp. 1014-1019.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E., 1953, "Equations of state calculations by fast computing machines," *Journ. Chem. Phys.,* Vol. 21, pp. 1087-1091.

Minsky, M.L. and Papert, S.A, 1969, *Perceptrons,* Cambridge, MA: MIT Press.

Minsky, M.L. and Papert, S.A, 1988, *Perceptrons,* Expanded Edition, Cambridge, MA: MIT Press

Mulier, F. and Cherkassky, V., 1995, "Self-organization as an iterative kernel smoothing process," *Neural Computation,* Vol. 7, No. 6, pp. 1165-1177.

Nasrabadi, N.M. and King, R.A., 1988, "Image coding using vector quantization: A review," *IEEE Trans. Communications,* Vol. 36, No. 8, pp. 957-971.

Nightingale, C. and Hutchinson, R.A, 1990, "Artificial neural nets and their applications to image processing," *British Telcom Technol. Journ.* Vol. 8, No. 3, pp. 81-93.

Nowlan, S.J., 1990, "Maximum likelihood competitive learning," in *Advances in Neural Information Processing Systems* 2, D.S. Tourezky ed., pp. 574-582.

Nowlan, S.J. and Hinton, G.E., 1992, "Simplifying neural networks by soft weight-sharing," *Neural Computation,* Vol. 4, No. 4, pp. 473-493.

Oehler, K.L. and Gray, R.M., 1995, "Combining image compression and classification using vector quantization, " *IEEE Trans. Pattern Anal. Machine Intell.,* Vol. 17, No. 5, pp. 461-473.

Oja, E., 1989, "Neural networks, principal components, and subspaces," *Intern. Journ. of Neural Systems,* Vol. 1, No. 1, pp. 61-68.

Ord, K., 1975, "Estimation methods for models of spatial interaction," *Journ. Amer. Statist. Asso.,* Vol. 70, pp. 120-126.

Osman, H. and Fahmy, M.M., 1994, "Probabilistic winner-take-all learning algorithm for radial-basis-function neural classifiers," *Neural Computation,* Vol. 6, No. 5, pp. 927-943.

Papoulis, A.A., 1965, *Probability, Random Variables, and Stochastic Processes*, New York: McGraw-Hill.

Pratt, W.K., Faugeras, O.D., and Gagalowicz, A., 1981, "Applications of stochastic texture field model to image processing," *Proc. IEEE*, Vol. 69, No. 5, pp. 542-551.

Qian, W. and Titterington, D.M., 1991, "Estimation of parameters in hidden Markov models," *Philosophical Trans. Roy. Soc. London A*, Vol. 337, pp. 407-428.

Rabbani, M. and Jones, P.W., 1991, *Digital Image Compression Techniques*, Bellington, Washington: SPIE Optical Engineering Press.

Richard, M.D. and Lippmann, R.P., 1991, "Neural network classifiers estimate Bayesian *a posteriori* probabilities," *Neural Computation*, Vol. 3, pp. 461-483.

Rioul, O. and Vetterli, M., 1991, "Wavelets and signal processing," *IEEE SP Magazine*, Oct., pp. 14-38.

Rissanen, J., 1984, "Universal coding, information, prediction, and estimation," *IEEE Trans. Inform. Theory*, Vol. 30, No. 4, pp. 629-636.

Ritter, H., 1991, "Asymptotic level density for a class of vector quantization processes," *IEEE Trans. Neural Networks*, Vol. 2, No. 1, pp. 173-175.

Ritter, H. and Schulten, K., 1986, "On the stationary states of Kohonen's self-organizing sensory mapping," *Biological Cybernetics*, Vol. 54, pp. 99-106.

Ritter, H. and Schulten, K., 1988, "Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability, and dimension selection," *Biological Cybernetics*, Vol. 60, pp. 59-71.

Ritter, H., Martinetz, T., and Schulten, K., 1992, *Neural Computation and Self-Organising Maps: An Introduction*, Reading, Massachusetts: Addion-Wesley.

Robbins, H. and Monro, S., 1951, "A stochastic approximation method," *Annals of Math. Statist.*, Vol. 22, pp. 400-407.

Roberts, S. and Tarassenko, L., 1994, "A probabilistic resource allocating network for novelty detection," *Neural Computation*, Vol. 6, No. 2, pp. 270-284.

Rosenblatt, F., 1958, "The Perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, Vol. 65, pp. 386-408.

Rosenfeld, A. and Kak, A.C., 1982, *Digital Picture Processing*, New York: Academic Press Inc.

Ruck, D.W., Rogers, S.K., Kabrisky, M., Maybeck, P.S., and Oxley, M.E., 1992, "Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 14, No. 6, pp. 686-691.

Rumelhart, D.E. and McClelland, J.L., eds., 1986, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Cambridge, MA: MIT Press.

Rumelhart, D.E. and Zipser, D., 1985, "Feature discovery by competitive learning," *Cognitive Science*, Vol. 9, pp. 75-112.

Sakrison, D.J., 1966, "Stochastic approximation: A recursive method for solving regression problems," in *Advance in Communication Systems: Theory and Applications 2*, pp. 51-100.

Schumacher, P. and Zhang, J., 1994, "Texture classification using neural networks and discrete wavelet transform," in *Proc. IEEE ICIP-94*, Vol. III, pp. 903-907.

Sejnowsky, T.J., 1976, "On global properties of neuronal interaction," *Biological Cybernetics*, Vol. 22, pp. 85-95.

Shang, C. and Brown, K., 1992, "Texture image classification based on interframe principal features," in *Artificial Neural Networks 2*, I. Aleksander and J. Taylor eds., pp. 1173-1176, Amsterdam: Elsevier.

Shepherd, G.M., 1988, *Neurobiology*, Oxford: Oxford University Press.

Siew, L.H., Hodgson, R.M., and Wood, E.J., 1988, "Texture measures for carpet wear assessment," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 10, pp. 92-105.

Simpson, P.K., 1990, *Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations*, New York: Pergamon.

Spitzer, F., 1971, "Markov random fields and Gibbs ensembles," *Amer. Math. Mon.*, Vol. 78, pp. 142-154.

Tan, T.N., 1995, "Texture edge detection by modelling visual cortical channels," *Pattern Recognition*, Vol. 28, No. 9, pp. 1283-1298.

Tarassenko, L. and Roberts, S., 1994, "Supervised and unsupervised learning in radial-basis-function neural networks," *IEE Proc. Vision, Image and Signal Processing*, Vol. 141, No. 4, pp. 210-216.

Thiran, P. and Hasler, M., 1994, "Self-organization of a one-dimensional Kohonen network with quantized weights and inputs," *Neural Networks*, Vol. 7. No. 9, pp. 1427-1439.

Traven, H.G.C., 1991, "A neural network approach to statistical pattern classification by 'semiparametric' estimation of probability density functions," *IEEE Trans. Neural Networks*, Vol. 2, No. 3, pp. 366-377.

Tsao, C and Gray, R.M., 1985, "Matrix quantizer design for LPC speech using the generalized Lloyd algorithm," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 36, pp. 537-545.

Tuceryan, M. and Jain, A.K., 1993, "Chapter 2.1: Texture analysis," in *Handbook of Pattern Recognition and Computer Vision*, C.H. Chen, L.F. Pau and P.S.P. Wang eds., Singapore: World Scientific Publishing Company.

Turner, M.R., 1986, "Texture discrimination by Gabor functions," *Biological Cybernetics*, Vol. 55, pp. 71-82.

Ueda, N. and Nakano, R., 1994, "A new competitive learning approach based on an equidistortion principle for designing optimal vector quantizers," *Neural Networks*, Vol. 7, No. 8, pp. 1211-1227.

Uspensky, J.V., 1937, *Introduction to Mathematical Probability*. New York: McGraw-Hill.

Van Hull, M.M. and Tollenaera, T., 1993, "A modular artificial neural network for texture processing," *Neural Networks*, Vol. 6, No. 1, pp. 7-32.

Vickers, A.L. and Modestino, J.W., 1982, "A maximum likelihood approach to texture classification," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 4, pp. 61-68.

Visa, A., Valkealaht, K., and Simula, O. 1991. "Cloud detection based on texture segmentation by neural network methods," in *Proc. of International Joint Conference on Neural Networks IJCNN-91 (Singapore)*, Vol. II, pp. 1001-1006.

Wasserman, P.D., 1989, *Neural Computing, Theory and Practice*, New York: Van Nostrand Reinhold.

Weszka, J.S., Dyer, C.R., and Rosenfeld, A., 1976, "A comparative study of texture measures for terrain classification," *IEEE Trans. Sys., Man, Cybern.*, Vol. 6, pp. 269-285.

Widrow, B., 1962, "Generalization and information storage in networks of adaline 'neuron'," in *Self-Organizing Systems*, M.C. Yovitz, G.T. Jacobi, and G.D. Goldstein eds., pp. 435-461, Washington, DC: Spartan Books.

Widrow, B. and Lehr, M.A., 1990, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proc. IEEE*, Vol. 78, pp. 1415-1442.

Willshaw, D.J., Buneman, O.P., and Longnet-Higgins, H.C., 1969, "Non-holographic associative memory," *Nature* (London), Vol. 222, ppp. 960-962.

Willshaw, D.J. and von der Malsburg, C., 1976, "How patterned neural connections can be set up by self-organisation," *Proc. Roy. Soc. London B*, Vol. 194, pp. 431-445.

Woods, J.W., 1972, "Two-dimensional discrete Markovian fields," *IEEE Trans. Inform. Theory,* Vol. 18, pp. 232-240.

Woods, J.W., Dravida, S., and Mediavilla, R., 1987, "Image estimation using doubly stochastic Gaussian random field models," *IEEE Trans. Pattern Anal. Machine Intell.,* Vol. 9, pp. 245-253.

Wright, W.A., Mackeown, W.P.J., and Greenway, P., 1995, "Chapter 11: The use of neural networks for region labelling and scene understanding," in *Neural Networks,* J.G. Taylor ed., pp. 165-192.

Xie, X.L. and Beni, G., 1991, "A validity measure for fuzzy clustering," *IEEE Trans. Pattern Analy. Machine Intell.,* Vol. 13, No. 8, pp. 841-847.

Yair, E., Zeger, K., and Gersho, A., 1992, "Competitive and soft competition for vector quantizer design, " *IEEE Trans. Signal Processing,* Vol. 40, No. 2, pp. 294-290.

Zador, P.L., 1966, "Asymptotic quantization of continuous random variables," unpublished memorandum, Bell Laboratories.

Zador, P.L., 1982, "Asymptotic quantization error of continuous signals and quantization dimension," *IEEE Trans. Inform. Theory,* Vol. 28, pp. 139-149.

Zeger, K., Vaisey, J., and Gersho, A., 1992, "Globally optimal vector quantizer design by stochastic relaxation, " *IEEE Trans. Signal Processing,* Vol. 40, No. 2, pp. 310-322.

Zhang, J., 1992, "The mean field theory in EM procedures for Markov random fields," *IEEE Trans. Signal Processing,* Vol. 40, pp. 2570-2583.

Zhang, J., Modestino, J.W., and Langan, D.A., 1994, "Maximum-likelihood parameter estimation for unsupervised stochastic model-based image segmentation," *IEEE Trans. Image Processing,* Vol. 3, No. 4, pp. 404-420.