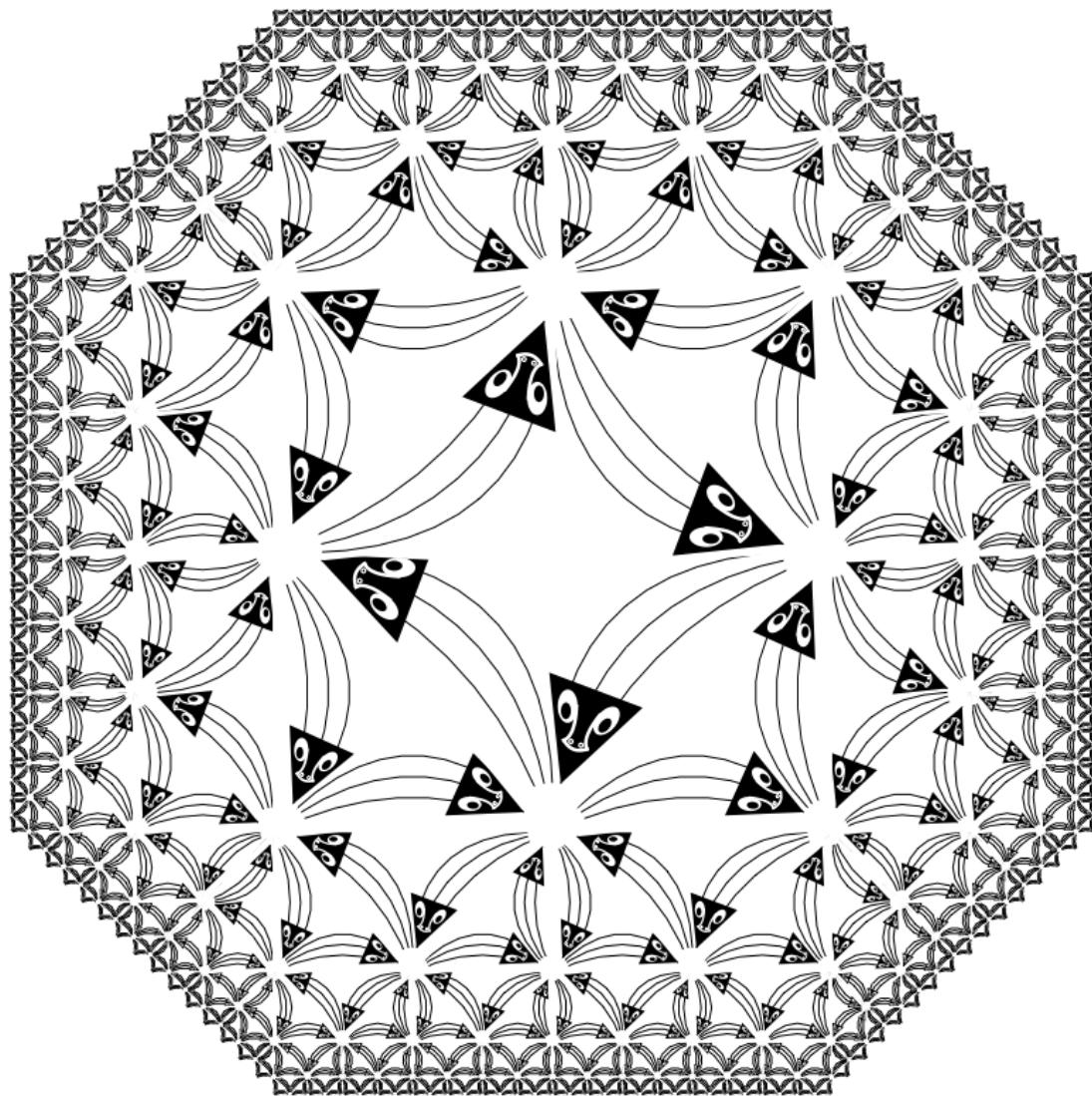


# **COLLECTED WORKS**

**— VOLUME 1 —**

**PUBLISHED PAPERS**



Ubique Sentio

# **COLLECTED WORKS**

**— VOLUME 1 —**

**PUBLISHED PAPERS**

---

**STEPHEN LUTTRELL**

## **COLLECTED WORKS: PUBLISHED PAPERS**

Version 0.9, May 22, 2019

Copyright © Stephen Luttrell 2019.

Published by [www.lulu.com](http://www.lulu.com).

Edited by Stephen Luttrell.

The material in this book was produced from the individual master files (or manuscripts) that were originally submitted for publication, and then the material was cross-checked with the papers as originally published. Many of the graphical illustrations have been recreated from their original source code, and all known typographical errors have been corrected.

Use of copyrighted material:

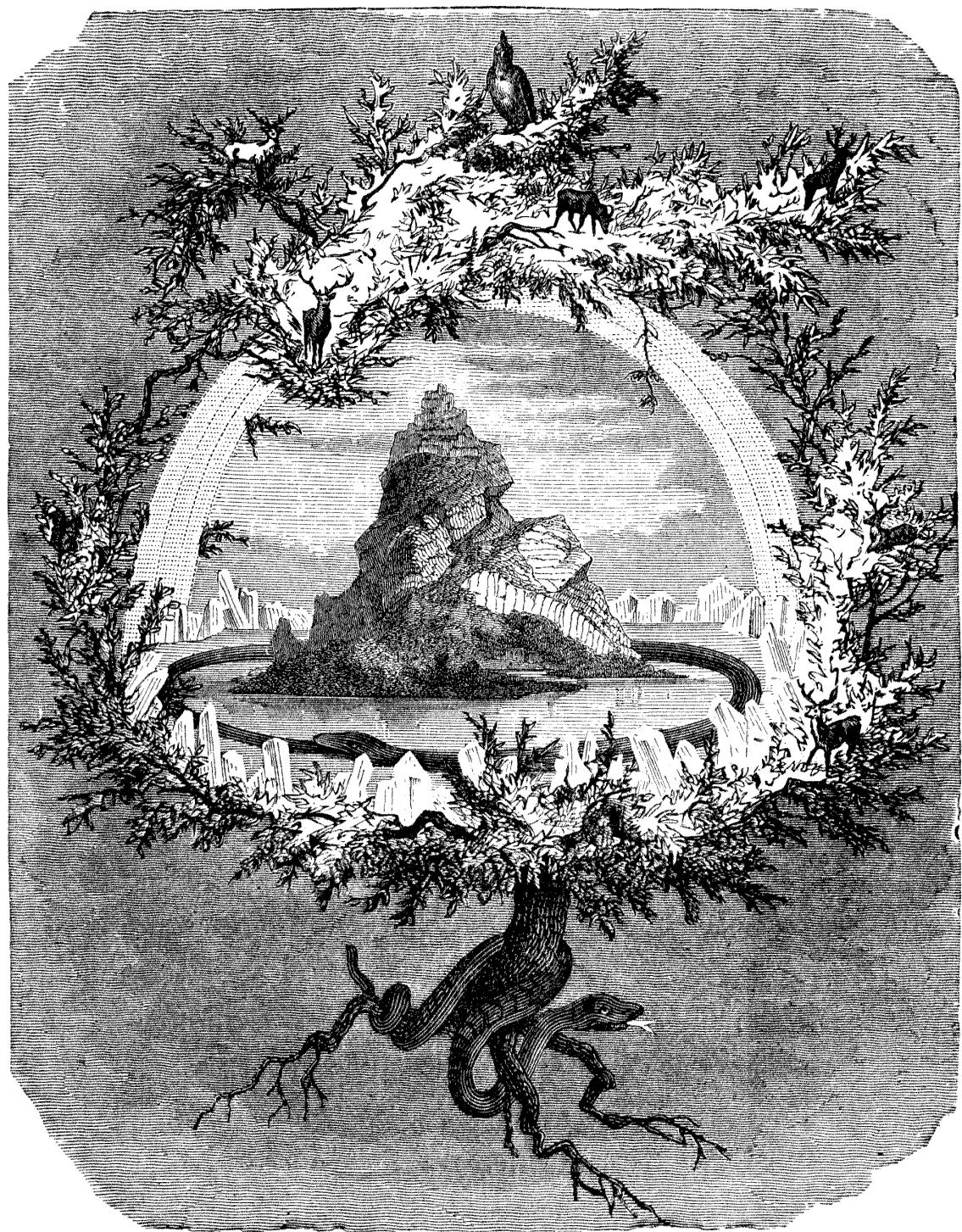
1. Copyrights in the literary works of the papers, journal papers, book chapters, technical reports, research notes, and unpublished papers are held by various organisations: the British Crown, QinetiQ, and occasionally publishing companies. In agreement number 20030206 (dated 15 July 2003) the British Crown granted permission to republish their works.
2. Copyrights in the typographical setting of the original works are held by the various organisations above. The works have been completely retypeset in this publication.
3. The frontispiece artwork created by the author is based on “Regular Division of The Plane VI” (1957) by M C Escher which is subject to Fair Use copyright (see [www.wikiart.org/en/m-c-escher/regular-division-of-the-plane-vi](http://www.wikiart.org/en/m-c-escher/regular-division-of-the-plane-vi)).

First published 2019.

The source text was prepared with LyX (document preparation system), and the source graphics was prepared with Mathematica (technical computing system) and GIMP (graphics editor). All of the source files were then typeset with L<sup>A</sup>T<sub>E</sub>X .

ISBN: XXXX





*An ash I know there stands,  
Yggdrasill is its name,  
a tall tree, showered  
with shining loam.  
From there come the dews  
that drop in the valleys.  
It stands forever green over  
Urðr's well.*



## Contents

<b>Journal Papers</b>	<b>1</b>
<b>1 Prior knowledge and object reconstruction using the best linear estimate technique</b>	<b>3</b>
<b>2 A new method of sample optimisation</b>	<b>11</b>
<b>3 The use of transinformation in the design of data sampling schemes for inverse problems</b>	<b>13</b>
<b>4 Prior knowledge in synthetic-aperture radar processing</b>	<b>27</b>
<b>5 An optimisation of the Metropolis algorithm for multibit Markov random fields</b>	<b>47</b>
<b>6 The use of Markov random field models to derive sampling schemes for inverse texture problems</b>	<b>49</b>
<b>7 The role of prior knowledge in coherent image processing</b>	<b>59</b>
<b>8 A maximum entropy approach to sampling function design</b>	<b>69</b>
<b>9 The inverse cross section problem for complex data</b>	<b>77</b>
<b>10 Image compression using a multilayer neural network</b>	<b>87</b>
<b>11 Hierarchical vector quantisation</b>	<b>93</b>
<b>12 Rapid acquisition of low signal-to-noise carriers</b>	<b>105</b>
<b>13 Derivation of a class of training algorithms</b>	<b>117</b>
<b>14 A Bayesian derivation of an iterative autofocus/super-resolution algorithm</b>	<b>123</b>
<b>15 The theory of Bayesian super-resolution of coherent images: a review</b>	<b>139</b>
<b>16 Code vector density in topographic mappings: scalar case</b>	<b>147</b>

<b>17 Self-supervised adaptive networks</b>	<b>159</b>
<b>18 Partitioned mixture distribution: an adaptive Bayesian network for low-level image processing</b>	<b>169</b>
<b>19 A Bayesian analysis of self-organising maps</b>	<b>183</b>
<b>20 Self-organisation of multiple winner-take-all neural networks</b>	<b>199</b>
<b>21 A discrete firing event analysis of the adaptive cluster expansion network</b>	<b>213</b>
<b>Conference Papers and Book Chapters</b>	<b>217</b>
<b>22 Prior knowledge in synthetic aperture radar (SAR) processing</b>	<b>219</b>
<b>23 The use of Markov random field models in sampling scheme design</b>	<b>225</b>
<b>24 Markov random fields: a strategy for clutter modelling</b>	<b>231</b>
<b>25 Markov random field image models of targets and clutter</b>	<b>239</b>
<b>26 Designing Markov random field structures for clutter modelling</b>	<b>243</b>
<b>27 Self-organising multilayer topographic mappings</b>	<b>249</b>
<b>28 Image compression using a neural network</b>	<b>253</b>
<b>29 The use of Bayesian and entropic methods in neural network theory</b>	<b>263</b>
<b>30 Inference theory</b>	<b>267</b>
<b>31 Self-organisation: a derivation from first principles of a class of learning algorithms</b>	<b>275</b>
<b>32 Hierarchical self-organising networks</b>	<b>279</b>
<b>33 A hierarchical network for clutter and texture modelling</b>	<b>287</b>

<b>34 Self-supervised training of hierarchical vector quantisers</b>	<b>297</b>
<b>35 Adaptive Bayesian networks</b>	<b>305</b>
<b>36 Gibbs distribution theory of adaptive n-tuple networks</b>	<b>315</b>
<b>37 An adaptive Bayesian network for low-level image processing</b>	<b>319</b>
<b>38 An adaptive Bayesian network for texture modelling</b>	<b>327</b>
<b>39 The cluster expansion: a hierarchical density model</b>	<b>341</b>
<b>40 The partitioned mixture distribution: multiple overlapping density models</b>	<b>349</b>
<b>41 Designing analysable networks</b>	<b>355</b>
<b>42 Using self-organising maps to classify radar range profiles</b>	<b>363</b>
<b>43 A theory of self-organising neural networks</b>	<b>371</b>
<b>44 Partitioned mixture distributions: the dynamical case</b>	<b>377</b>
<b>45 Partitioned mixture distributions: a first order perturbation analysis</b>	<b>383</b>
<b>46 Optimal response functions in a network of discretely firing neurons</b>	<b>389</b>
<b>47 Self-organised modular neural networks for encoding data</b>	<b>395</b>
<b>48 An adaptive network for encoding data using piecewise linear functions</b>	<b>413</b>
<b>49 The emergence of dominance stripes and orientation maps in a network of firing neurons</b>	<b>419</b>
<b>50 Self-organised discovery of structure in signal manifolds</b>	<b>433</b>
<b>51 Using stochastic encoders to discover structure in data</b>	<b>439</b>
<b>52 Invariant stochastic encoders</b>	<b>443</b>

<b>53 Adaptive subspace encoders using stochastic vector quantisers</b>	<b>447</b>
<b>54 Adaptive sensor fusion using stochastic vector quantisers</b>	<b>453</b>
<b>55 A self-organising approach to multiple classifier fusion</b>	<b>459</b>
<b>56 Using stochastic vector quantisers to characterise signal and noise subspaces</b>	<b>465</b>
<b>57 A Markov chain approach to multiple classifier fusion</b>	<b>473</b>
<b>58 Guided tour of publications</b>	<b>481</b>

## Preface

I have often received comments that my published papers are spread all over the place, and that it is necessary to read a large subset of my papers before they individually begin to make sense. There is also the problem that many of my key papers were published in conference proceedings, or not published at all, which makes it difficult to obtain copies of them. These are the main reasons for the creation of this set of 4 volumes, which collect all of my papers so that they can be conveniently accessed alongside each other. There is some duplication of material, where for instance a technical report is very similar to a subsequently published paper, but I have included everything for completeness. The material is split between the 4 volumes as follows:

1. Published Papers – journal papers, conference papers, and book chapters.
2. Reports : Part 1 – Royal Signals and Radar Establishment (RSRE), Defence Research Agency (DRA), and Defence Evaluation and Research Agency (DERA) technical reports and research notes.
3. Reports : Part 2 – QinetiQ technical reports.
4. Unpublished Papers – mostly arXiv papers.

I worked as a British scientific civil servant for about 20 years, and during that time I published a lot of scientific papers all of which are British Crown copyright. Fortunately, Her Majesty's Stationery Office (HMSO) has very liberal views on allowing the reproduction of Crown copyright material. This has made possible the creation of this collection of papers which contains retypeset versions of all of my publications.

The route by which I produced this set of volumes was long and torturous – starting sometime in the 1990s, and continuing off-and-on (more off than on) up to the publication of this set of volumes. I experimented with many different software applications for processing the retypeset publications, but I ultimately converged on using L<sup>A</sup>T<sub>E</sub>X – via the LyX front-end to L<sup>A</sup>T<sub>E</sub>X – as being the only document processor that could easily create the high-quality results that I needed. I should have started doing things this way on “day one”, but I wrongly believed that there must be a better approach, so I went off to explore the universe of alternative possibilities before finally returning to L<sup>A</sup>T<sub>E</sub>X.

## Acknowledgements

I would like to thank various people for discussions which led to rapid advances in my understanding during the early stages of my basic research programme into information processing. Ordered chronologically, the key steps were: Chris Oliver (compound statistical modelling), Roy Pike (singular value decomposition), Steve Gull (SVD, Wiener filtering, Bayesian methods) and with John Skilling (maximum entropy method), Andrew Sleigh (artificial intelligence), Jonathan Pritchard (WISARD pattern recognition device, Kohonen self-organising networks), Robert Tough (stochastic processes, stochastic differential equations), Julian Besag (Markov random fields), John Bridle (Boltzmann Machine, hidden Markov models), David MacKay (stimulating discussions about Bayesian methods and information theory), Chris Webber (stimulating discussions about self-organising networks), Andrew Webb (stimulating discussions about many areas of information processing research).

I am indebted to the many people who were involved in various ways with writing software to implement the fruits of my information processing research – they breathed fire into my equations. I would also like to thank Stephen Wolfram for creating Mathematica powered by the Wolfram computer programming language, without which I would not have had the patience to write software myself – this has been my long-term “secret weapon” for making rapid progress in my research.

The Ministry of Defence (United Kingdom) continuously supported my long-term basic research programme into information processing. This started with the MoD’s “basic research advisory committee” recruiting me as a civil servant to work at the government’s Royal Signals and Radar Establishment. This establishment went through a couple of reorganisations and name-changes as the Defence (Evaluation and) Research Agency, and it was eventually sold off and privatised as QinetiQ.

## Introduction

The motivation for the research that is described in these volumes is the wish to explain things in terms of their underlying causes, rather than merely being satisfied with phenomenological descriptions. When this reductionist approach is applied to information processing it allows the internal structure of information to be analysed, so information processing algorithms can then be derived from first principles.

One of the simplest examples of this approach is the diagonalisation of a data covariance matrix – there are many variants of this basic approach, such as singular value decomposition – in which the assumed independent components of high-dimensional data are identified and extracted. The main limitation of this type of information analysis approach is that it is based on linear algebra applied globally to the data space, so it is unable to preserve information about any local data structure in the data space. For instance, if the data lives on a low-dimensional curved manifold embedded in the data space, then only the global properties of this manifold would be preserved by global linear algebra methods.

In practice, data whose high-dimensional structure is non-trivial typically lives on a noisy version of a curved manifold, so techniques for analysing such data must automatically handle this type of structure. For instance, a blurred image of a point source is described by its underlying degrees of freedom – i.e. the position of the source – and as the source moves about it generates a curved manifold that lives in the high-dimensional space of pixel values of the sampled image. The basic problem is then to deduce the internal properties of this manifold by analysing examples of such images. A more challenging problem would be to extend this analysis to images that contain several overlapping blurred images of point sources, and so on. There is no limit to the complexity of the types of high-dimensional data that one might want to analyse.

These methods then need to be automated so that they do not rely on human intervention, which would then allow them to be inserted as “components” into information processing networks. The purpose of the research that is described in these volumes is to develop principled information processing methods that can be used for such analysis. Self-organising information processing networks arise naturally in this context, in which ways of cutting up the original manifold into simpler pieces emerge automatically.



# Journal Papers



# Prior Knowledge and Object Reconstruction Using the Best Linear Estimate Technique \*

S P Luttrell

Royal Signals and Radar Establishment, Malvern, Worcestershire, England

Prior knowledge and image data are combined to produce an object reconstruction, using the best-linear-estimate technique. It is shown how this technique is related to the minimum norm method and the singular function method. How prior knowledge influences the object reconstruction is examined in detail. The concentration of image degrees of freedom to produce a resolution enhancement in the object reconstruction is demonstrated for suitable types of prior knowledge.

## I. INTRODUCTION

The process of image formation is imperfect (that is, information about the object is lost), and so the image is not a faithful representation of the object. Here we investigate how the introduction of prior knowledge of the object's form assists the reconstruction of an object from its image [12]. This will amount to a partial inversion of the imaging process (for a discussion of which see, e.g., [3, 4]). We must choose a reconstruction algorithm which is stable with respect to noise errors in the image data [14], and which combines the information present in the noisy image with that available as prior knowledge. There is no technique which stands out as always being preferable to any other. However, we use the best-linear-estimate method [4] since its fulfils all the requirements while possessing a simple mathematical form.

In Section II we summarize the best linear estimate method, and indicate its relation to the minimum norm method (with Miller regularisation). In Section III we derive some properties of the reconstruction formula and indicate a relationship with the singular function method. In Section IV we investigate these properties numerically for an imaging system consisting of a simple one-dimensional lens with a point-sampled image.

## II. CHOICE OF RECONSTRUCTION PROCEDURE

We use operator notation for the imaging process to simplify the derivation of the necessary results. Let us represent the object by the state  $f$ , which is transformed by a linear imaging operator  $T$  to yield an image state

$$g = T f + n \quad (2.1)$$

where  $n$  is additive image noise. The states  $g$  and  $n$  lie in the same space, and we assume that  $f$ ,  $g$  and  $n$  are all members of Hilbert spaces. The data which represent  $g$

can be used to characterize  $f$  up to a resolution length determined by the properties of  $T$ . We are concerned here with the inverse problem of recovering  $f$  from the data with the assistance of prior knowledge of the form of  $f$ .

Let us restrict ourselves to a linear reconstruction algorithm

$$f_r = R g \quad (2.2)$$

where  $f_r$  lies in the same space as  $f$ , and  $R$  is a linear reconstruction operator. The simplest criterion for selecting  $R$  is to demand that the expected norm  $e$  of the reconstruction error,  $f_r - f$ , be minimised [4]. Thus we define

$$\begin{aligned} e &\equiv \langle (f_r - f)^\dagger (f_r - f) \rangle \\ &= \langle ((RT - 1) f + Rn)^\dagger ((RT - 1) f + Rn) \rangle \\ &= \text{tr} \left\{ (RT - 1) W (RT - 1)^\dagger + R N R^\dagger \right\} \end{aligned} \quad (2.3)$$

where we have introduced the notation

$$\begin{aligned} \langle f f^\dagger \rangle &\equiv W \\ \langle n n^\dagger \rangle &\equiv N \\ \langle f n^\dagger \rangle &\equiv 0 \end{aligned} \quad (2.4)$$

Minimising  $e$  with respect to  $R$  gives the best-linear-estimate (BLE) formula for  $f_r$ ,

$$f_r = W T^\dagger [T W T^\dagger + N]^{-1} g \quad (2.5)$$

For the special case where  $f$  and  $n$  have zero-mean gaussian prior probability distributions (with covariances  $W$  and  $N$  respectively)

$$P[f] \propto \exp(-f^\dagger W^{-1} f) \quad (2.6)$$

$$P[n] \propto \exp(-n^\dagger N^{-1} n) \quad (2.7)$$

the conditional probability distribution of  $g$ , given  $f$ , is (see Equation 2.1)

$$P[g|f] \propto \exp \left( - (T f - g)^\dagger N^{-1} (T f - g) \right) \quad (2.8)$$

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 10, 2019.

This paper appeared in *Optica Acta*, 1985, vol. 32, no. 6, pp. 703-716. Received 9 February 1984; revision received 18 October 1984.  
© Controller, Her Majesty's Stationery Office, London, 1985.

and the *a posteriori* probability distribution of  $f$ , given  $g$ , is (using Bayes' rule)

$$P[f|g] \propto \exp \left( - (Tf - g)^\dagger N^{-1} (Tf - g) - f^\dagger W^{-1} f \right) \quad (2.9)$$

The value of  $f$  which maximises  $P[f|g]$  is obtained by minimising

$$M[f] \equiv (Tf - g)^\dagger N^{-1} (Tf - g) + f^\dagger W^{-1} f \quad (2.10)$$

with respect to  $f$ . The stationary point is given by Equation 2.5, the BLE reconstruction. This is not surprising since choosing the peak point of a gaussian distribution must minimise the average squared error between that choice and all other possible choices. (Note that this follows from the choice of gaussian prior distribution, and does not hold in general.) However, Equation 2.10 has the form used in the minimum norm method with Miller regularisation, which has been studied extensively [1, 2, 8, 9]. The derivation used here is less empirical than the methods used elsewhere, since the physical meaning of the two matrices  $W$  and  $N$  is clear. In particular we are not restricted to using diagonal  $W$  and  $N$ .

The linear reconstruction procedure given by Equation 2.2, together with the prescription (minimum average error norm) for  $R$  is clearly not unique. However, it is minimal in the sense that there is no procedure with a simpler functional form. In Section III we examine its properties, and in particular how the prior knowledge  $W$  can influence the object reconstruction.

### III. THE EFFECT OF PRIOR KNOWLEDGE

Let us consider an additive image noise covariance  $N$  which is isotropic:

$$N_{ij} \equiv n^2 \delta_{ij} \quad (3.1)$$

Equation 2.5 may be clarified by introducing the eigenstates  $\psi_i$  and the associated eigenvalues  $\lambda_i$  of  $TWT^\dagger$ :

$$TWT^\dagger \psi_i = \lambda_i \psi_i \quad (3.2)$$

where

$$\lambda_i \geq 0 \quad (3.3)$$

because  $TWT^\dagger$  is non-negative definite. Let us normalise the eigenstates,

$$\psi_i^\dagger \psi_j = \delta_{ij} \quad (3.4)$$

and expand  $g$ :

$$g = \sum_i g_i \psi_i \quad (3.5)$$

Note that any  $g$  may be expanded in this way since the  $\psi_i$  form a complete orthonormal set. Equation 2.5 now yields

$$f_r = WT^\dagger \sum_i \frac{g_i \psi_i}{\lambda_i + n^2} \quad (3.6)$$

Comparing Equation 3.6 and Equation 3.5 reveals that the reconstruction process consists of making the following two replacements in Equation 3.5:

$$\begin{aligned} g_i &\longrightarrow \frac{g_i}{\lambda_i + n^2} \\ \psi_i &\longrightarrow WT^\dagger \psi_i \end{aligned} \quad (3.7)$$

This reconstruction procedure is almost identical to that introduced in [7]. There the prior  $W$  was taken to be diagonal, with a top-hat profile, and a singular function decomposition (SVD) of the object-lens-image system was made. The analogue of Equation 3.6 used in [7] does not contain the image noise term  $n^2$ ; instead the reconstruction in [7] is regularised by omitting all terms in the summation which are excessively corrupted by image noise. When the energy contained in the image signal is much greater than that in the image noise there is a negligible difference between the various possible ways of regularising Equation 3.6, and so the SVD method will yield the same results as the BLE method for the problem examined in [7]. However, the BLE goes beyond what was achieved in [7] since it will admit prior knowledge in the form of a full covariance. The SVD has been extended to close this gap in capability by relaxing the support constraint into a weight function (diagonal  $W$ ) [6, 11] and by introducing a full covariance (non-diagonal  $W$ ) via a Karhunen-Loëve expansion [11]. This latter generalisation is fully equivalent to the BLE for a large signal-to-noise ratio. However, the BLE derivation of Equation 3.6 is to be preferred for its simplicity, and regularisation is achieved without having to use a hard cut-off of the summation in Equation 3.6, as is usually done with the SVD. We point out that the  $\lambda_i = 0$  states do not contribute to  $f$ , in Equation 3.6, so there is an implicit cut-off in the BLE.

A desirable property of the BLE is that it ‘gracefully degrades’ in the absence of useful prior knowledge; that is,  $f$ , contains no more information than  $g$  does under such conditions. The simplest example of this is when  $T$  is the bandlimiting operator  $B$ , and  $g$  is a Nyquist-sampled image. Then, for isotropic  $W$  (no useful prior knowledge) it is easy to show that the  $\lambda_i$  are all equal (see Equation 3.2). The  $\psi_i$  may then be chosen to be the image samples themselves (by a unitary transformation). Thus the reconstruction procedure given by Equation 3.7 yields (in one dimension)

$$f_r(x) \propto \sum_k \frac{g_k \sin c(x - \frac{k\pi}{c})}{(x - \frac{k\pi}{c})} \quad (3.8)$$

where  $k$  labels the image samples and  $c$  is the band limit of  $T$ . Equation 3.8 is just the usual ‘sinc interpolation’ of Nyquist-sampled data.

We may investigate how a non-trivial  $W$  influences  $f_r$  by examining the Fourier transformed reconstruction. From Equation 3.7 we see that  $f_r$  is composed of the  $WT^\dagger \psi_i$ . From Equation 3.2 we may write the following equation for these components:

$$WT^\dagger T (WT^\dagger \psi_i) = \lambda_i (WT^\dagger \psi_i) \quad (3.9)$$

In coordinate representation this becomes

$$W(x) \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} dy dz T^*(z, x) T(z, y) \xi_i(y) = \lambda_i \xi_i(x) \quad (3.10)$$

where

$$\xi_i \equiv WT^\dagger \psi_i \quad (3.11)$$

and where (for simplicity) we have assumed that  $W$  is diagonal. If, in addition, we assume that  $T$  is translation invariant (isoplanatic),

$$T(x, y) \equiv T(x - y) \quad (3.12)$$

then the Fourier transform of Equation 3.10 is

$$\hat{W} * |\hat{T}|^2 \xi_i = \lambda_i \xi_i \quad (3.13)$$

Since we need consider only non-zero  $\lambda_i$ , the bandwidth of the  $\xi_i$  is the sum of the bandwidths of  $\hat{W}$  and of  $|\hat{T}|^2$  (i.e. the bandwidth of  $\hat{W} * |\hat{T}|^2$ ). In general the reconstruction  $f_r$  will contain frequencies higher than those which pass through  $T$ , where the band limit is extended by the bandwidth of  $W$ . Thus non-trivial  $W$  will enhance the bandwidth of the reconstruction. Note that non-diagonal  $W$  may be analysed in a similar fashion by first transforming (unitarily) to the frame in which  $W$  is diagonal. In Section IV we examine how the effects of such a bandwidth extension manifest themselves in the reconstruction.

#### IV. NUMERICAL RESULTS

We can now present some numerical results for a one-dimensional simple lens with a point-sampled image, for which we must solve (see Equation 3.2) the equation

$$\sum_\nu \int_{-\infty}^{+\infty} dy \frac{\sin[c(x_\mu - y)]}{\pi(x_\mu - y)} W(y) \frac{\sin[c(x_\nu - y)]}{\pi(x_\nu - y)} \psi_i(x_\nu) = \lambda_i \psi_i(x_\mu) \quad (4.1)$$

where the  $x_\mu$  are the image sample points. The integration and diagonalisation required to solve this equation may be accomplished entirely by using standard routines [13]. The corresponding object basis is then given by (see Equation 3.11)

$$\xi_i(x) = W(x) \sum_\nu \frac{\sin[c(x_\nu - x)]}{\pi(x_\nu - x)} \psi_i(x_\nu) \quad (4.2)$$

Note that we order the eigenstates in decreasing size of eigenvalue.

There are several parameters which can be varied in order to explore the behaviour of  $f_r$ . We may vary the way in which the image is point-sampled (the  $x_\mu$ ), but we restrict ourselves to a specified number of samples uniformly spaced along the image. We may also vary the bandwidth (the parameter  $c$ ) of the system. However, we use the ‘effective Shannon number’ of the system, which measures the number of resolution cells spanned by the object. It is this quantity which determines how effectively the prior knowledge  $W$  combines with the image data in forming an object reconstruction. In [7] it is shown how small Shannon numbers are potentially the most interesting for object reconstruction because of the

large resolution enhancement which is obtainable. Here we restrict ourselves to Shannon numbers of the order of unity, and also to diagonal (uncorrelated)  $W$  which we refer to as a ‘weight’.

As a preliminary check on our results we present in the table a comparison of the eigenvalues produced from Equation 4.1 for  $c = \frac{1}{2}\pi$ ,  $\pi$  and weight  $D$  (the top-hat function of unit half-width), with the known eigenvalues of the equivalent prolate spheroidal functions [7]. We have chosen to use 25 sample points spaced at the Nyquist frequency and centred on the top-hat weight, and have adjusted the eigenvalues to compensate for the trivial sampling density factor. The small differences between the continuous and the sampled results are due to edge effects. We observe immediately that the errors are always such that the discrete problem has a smaller eigenvalue than the equivalent continuous problem. This indicates that when the continuous image is projected onto a finite number of (Nyquist) samples some further information is lost; thus we have to work harder to recover the object, and hence the smaller eigenvalues.

If we increase the number of samples while retaining the same total length of image (i.e. oversample), then the

Table I: Comparison of discrete and continuous sampling for an impulse response  $B$  with a top-hat object weighting  $D$ . The discrete image is sampled 25 times at the Nyquist frequency, and the eigenvalues are scaled to compensate for the sampling density factor.

Band limit $c$	Eigenstate	Eigenvalue	
		Discrete	Continuous
$\frac{1}{2}\pi$	0	0.783	0.783
	1	0.197	0.205
	1	$0.114 \times 10^{-1}$	$0.114 \times 10^{-1}$
	3	$0.190 \times 10^{-3}$	$0.215 \times 10^{-3}$
	4	$0.215 \times 10^{-5}$	$0.216 \times 10^{-5}$
	5	$0.109 \times 10^{-7}$	$0.136 \times 10^{-7}$
	6	$0.581 \times 10^{-10}$	$0.592 \times 10^{-10}$
	0	0.981	0.981
	1	0.736	0.750
	2	0.244	0.244
$\pi$	3	$0.219 \times 10^{-1}$	$0.246 \times 10^{-1}$
	4	$0.106 \times 10^{-2}$	$0.107 \times 10^{-2}$
	5	$0.220 \times 10^{-4}$	$0.274 \times 10^{-4}$
	6	$0.473 \times 10^{-6}$	$0.480 \times 10^{-6}$

eigenvalues are even better estimates of the continuous values. This can be viewed as a way of suppressing the edge effects by overdetermining the state of the image. A detailed analysis of the effects of sampling is contained in [5].

Now let us generalise the results somewhat by adding a pedestal to  $D$ :

$$W(x) = \begin{cases} 1 + \varepsilon & |x| \leq 1 \\ \varepsilon & |x| > 1 \end{cases} \quad (4.3)$$

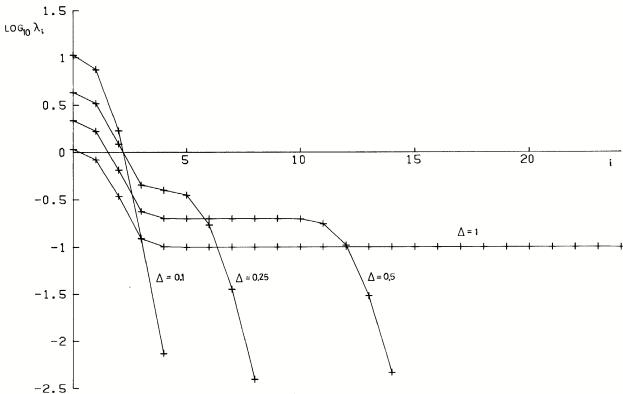


Figure 1: Plot of eigenvalue versus order of eigenstate for a simple lens with an image sampled 25 times at the Nyquist rate. A band limit  $c = \pi$  and an object weight  $D + 0.1I$  (see Equation 4.3) have been used. Four different values of the parameter  $\Delta$  have been used to show the response of the eigenvalue spectrum to the sample spacing.

Some results on the effect of varying the sample spacing when  $\varepsilon = 0.1$  are presented in Figure 1. Lines have been drawn to guide the eye from eigenvalue to eigenvalue under fixed conditions. The parameter  $\Delta$  is the sampling interval in units of the Nyquist length, thus  $\Delta = 1$  corresponds to Nyquist sampling. Note how as the order is increased the initial decrease in the eigenvalues is followed by a region over which there is no change. The difference in this constant eigenvalue between the various sample spacings is an artefact of the sampling density and is of no fundamental significance. More interesting is the way in which the eigenvalues behave for the different sample spacings as the order is increased further. The Nyquist ( $\Delta = 1$ ) sampling case is well behaved. However the other cases, all of which represent some degree of oversampling, have a second knee in the curve beyond which the eigenvalues drop steeply. This is easily explained: because the total number of samples is constant (25), the total length of image sampled is inversely proportional to the sampling interval. Since the total number of degrees of freedom which have propagated from the object to the sampled image is approximately the number of Nyquist lengths contained in the interval which is sampled, then oversampling while retaining the same total number of samples reduces the amount of extracted information. The presence of image noise which corrupts the above information and which introduces additional degrees of freedom in the form of out-of-band components (which are aliased) does not affect this argument since the eigenvalues relate only to the propagated information. Clearly, halving the sampling interval means that approximately half the number of degrees of freedom are measured, so reconstructing an object with more than this number is going to be exceedingly difficult. This is reflected in the steep decline in the eigenvalues beyond about the eleventh order for  $\Delta = 0.5$ , and beyond about the fifth order for  $\Delta = 0.25$ . When  $\Delta = 0.1$  such a short piece of image is sampled that the eigenvalues never attain a constant value. Thus the effect of reducing the sample spacing to below the Nyquist length (for a fixed total number of samples) is to rob ourselves of information on the true image, although such oversampling may be used to combat the effects of noise. Conversely, we find that if the sample spacing is increased beyond the Nyquist length (i.e. undersampling), then the eigenvalue curve (not shown in Figure 1) tails off to the same limit as for  $\Delta = 1$  (shown in Figure 1) and does not drop off at large orders as for  $\Delta < 1$ . This is because samples for which  $\Delta > 1$  are almost independent, so the same number of degrees of freedom are measured as for  $\Delta = 1$ . We comment below on the effect of undersampling on the object reconstruction. Further work on how to optimize the sample positions is presented elsewhere [10].

We now present some eigenvalue spectra for various Shannon numbers and various  $c$ . These are shown in Figure 2 for a simple lens with top-hat weight  $D$  producing images sampled at 25 points at the Nyquist frequency. Two band limits ( $c = \frac{1}{2}\pi$  and  $\pi$ ) and three possible ex-

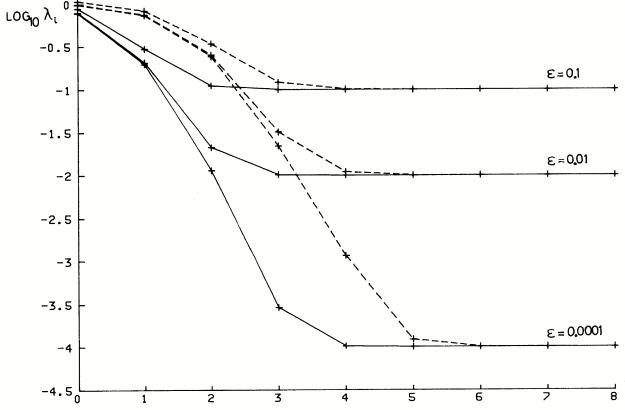


Figure 2: Plot of eigenvalue versus order of eigenstate for a simple lens with an image sampled 25 times at the Nyquist rate. The continuous curves are for  $c = \pi$ , and the dashed curves for  $c = \frac{1}{2}\pi$ . Three different object weights of the form  $D + \varepsilon I$  have been used, with  $\varepsilon = 0.0001, 0.01$  and  $0.1$ . The eigenvalues have been scaled to unit effective sample density.

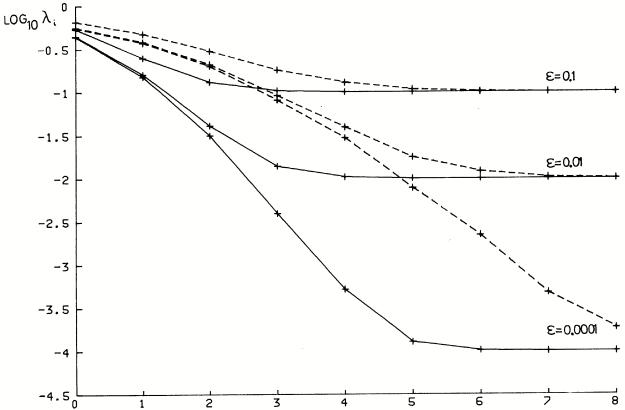


Figure 3: As Figure 2, but for an object weight of the form  $G + \varepsilon I$ .

tended weights ( $\varepsilon = 0.1, 0.01$  and  $0.0001$ ) were considered. As in the table, a scale factor has been applied in Figure 2 to convert the effective sample density to unity. In Figure 3 the corresponding results are shown for a smoother weight,  $D$  being replaced by a gaussian  $G$  given by

$$G(x) \equiv \exp\left(-\frac{1}{4}\pi x^2\right) \quad (4.4)$$

The height of this gaussian weight is chosen to be the same as that of the top-hat weight; the relative heights of the central and extended regions (for given  $\varepsilon$ ) are thus unchanged. The width of the gaussian is chosen so that the area under the weight remains the same. The only significant difference between Figure 2 and Figure 3 is that the gaussian weight has smaller eigenvalues for low orders and larger eigenvalues for high orders; this is because the gaussian weight has softer edges than the cor-

responding top-hat weight.

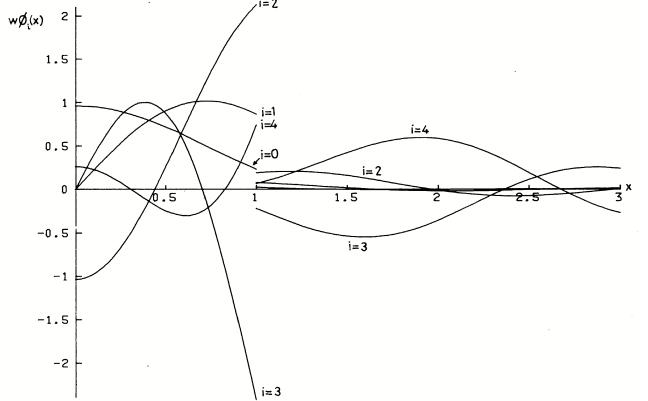


Figure 4: Plot of the object basis functions for a simple lens with an image point sampled 25 times at the Nyquist rate. The object weight is  $D + 0.1 I$ , and  $c = \pi$ .

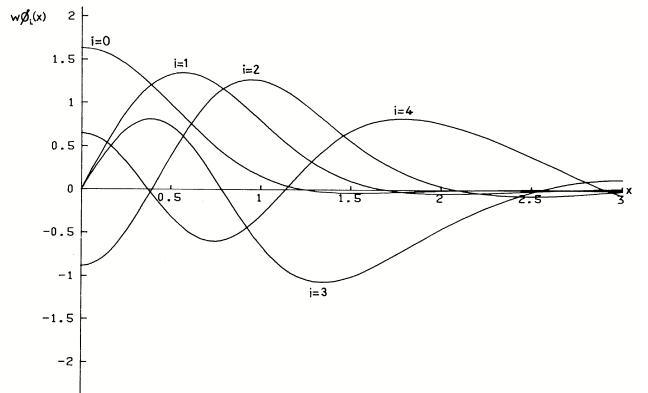


Figure 5: As Figure 4, but for an object weight of the form  $G + 0.1 I$ .

In Figure 4 and Figure 5 we present the object basis states for  $c = \pi$ , computed using Equation 3.11. For the extended weight chosen  $\varepsilon = 0.1$ , which is the most dilute form of prior knowledge which usefully contributes to the object reconstruction. Note how each state reconstructs the central and the extended region, but that only the low orders are of any use in reconstructing the central region. Note also that the zeros in the central region of the low-order object basis functions are closer together than are any other zeros. This suggests that the resolution which can be obtained is greater in the central region. This is intuitively reasonable since the central region is weighted more heavily than elsewhere, and so the object structure in that region is given greater priority in the reconstruction.

This interpretation is valid only for Shannon numbers of the order of unity or less. We have shown (see Equation 3.13) that it is the finite bandwidth of  $W$  that gives an increase of the bandwidth in the object reconstruc-

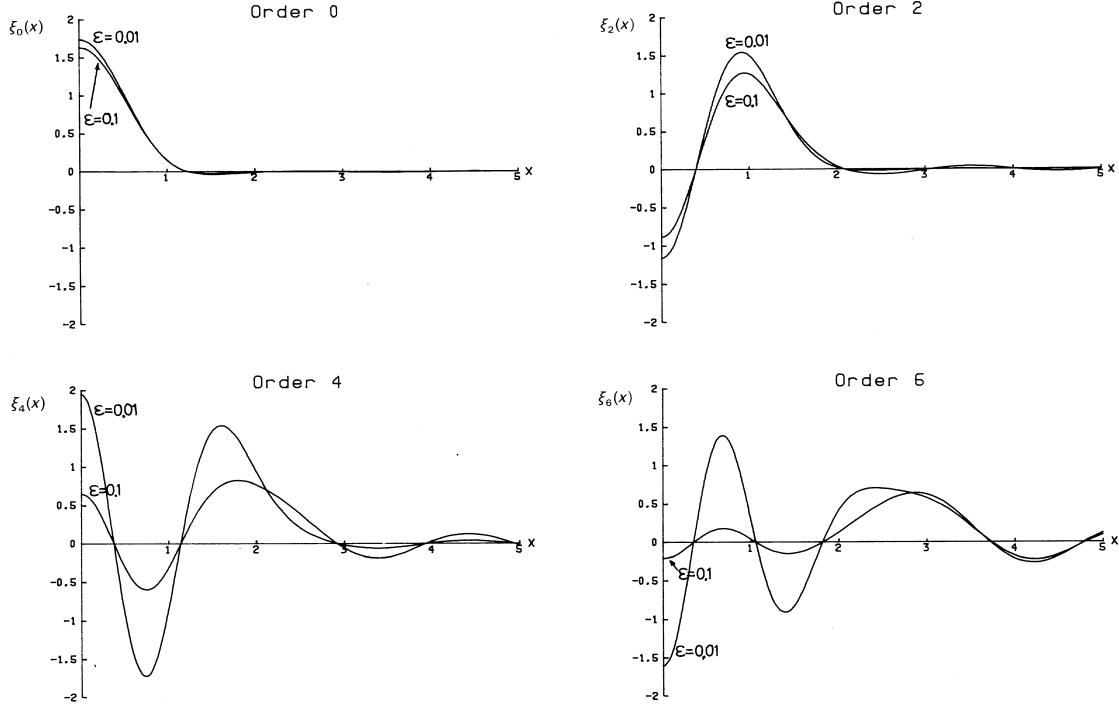


Figure 6: Plots of the even object basis functions for a simple lens with an image point sampled 25 times at the Nyquist rate. The object weight is  $G + \varepsilon I$ , and  $c = \pi$ . Two extended weights have been used:  $\varepsilon = 0.1$  and  $0.01$ .

tion. We may loosely interpret the bandwidth of  $W$  as arising from those regions where  $W$  varies - the edges of the object. For small Shannon numbers the object is all ‘edge’ on a length scale given by the lens resolution. For large Shannon numbers the edge regions are confined to the edges (and not the centre) of the object. We therefore expect that the resolution enhancement seen at low Shannon numbers will be absent at large Shannon numbers. The crucial point is that the effect of the increased bandwidth is localised at those places where  $W$  is not constant.

In Figure 6 we show an example of how the object basis for a gaussian central weight is affected by the size of the extended weight. The even states are shown for  $c = \pi$ , and for two values of  $\varepsilon$ , 0.01 and 0.1. For a given order the curve with higher values of  $\xi$  at  $x = 0$  corresponds to the smaller values of  $\varepsilon$ . This is entirely reasonable since a smaller  $\varepsilon$  should allow a more effective reconstruction of the central object to be made. Note that as  $\varepsilon$  is increased the higher-order states rapidly become weaker in the central region; this is because the extent to which  $W$  provides non-trivial information about the true form of  $f$  decreases as  $\varepsilon$  increases. Thus the degree of resolution enhancement for the small Shannon number will decrease as  $\varepsilon$  increases (raising the effective Shannon number). This follows naturally because the number of states which effectively reconstruct the central region will decrease.

The effect on the object basis of increasing the image sample spacing to  $\Delta > 1$  is similar to that of increasing

$\varepsilon$ . For  $\Delta > 1$  the higher-order object states become very weak in the central region. This is because an under-sampled image has fewer samples near the origin than a Nyquist sampled image, and so it provides a poorer constraint on the form of the central object. Thus the enhanced resolution found in the central region when  $\Delta \leq 1$  is less marked when  $\Delta > 1$ .

We now define a measure of resolution enhancement which may be used to compare the various weights and Shannon numbers when the extended weight is the factor which limits the resolution. The measure defined in [7] relies on the fact that a top-hat weight without extension ( $\varepsilon = 0$ ) forces the  $n$ th object basis function to have  $n$  zeros in the central region. We have seen that when an extended weight is included, the strength with which a state reconstructs the central region first decreases as the order increases, then rapidly tails off as the knee of the eigenvalue curve (see Figure 2) is passed. We could therefore define a resolution measure in the spirit of [7] by identifying the highest-order object state which gives a significant reconstruction in the central region. However, we do not restrict ourselves to top-hat type weights, so in general we shall be unable to make definite predictions about the form of the object basis or its eigenvalue spectrum. Therefore we define an empirical measure of resolution enhancement as the ratio of the width of the impulse response of the lens to the width of its reconstruction. Thus the resolution enhancement factor  $E$  will

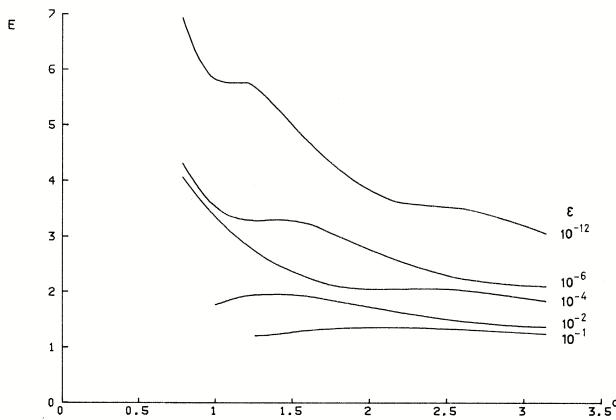


Figure 7: Plot of resolution enhancement versus order of eigenstate for a simple lens with an image sampled 25 times at the Nyquist rate. The object weight is  $D + \varepsilon I$ , and  $c = \pi$ .

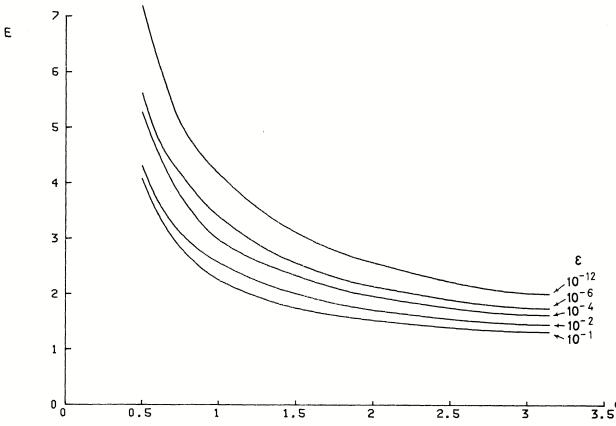


Figure 8: As Figure 7, but for an object weight  $G + \varepsilon I$ .

be defined as

$$E \equiv \frac{a_i}{a_r} \quad (4.5)$$

where  $a_i$  is the width at half-maximum of the intensity of the image of the point source, and  $a_r$  is the corresponding width of its reconstruction. Figure 7 and Figure 8 show some plots of  $E$  versus  $c$  for various weights with the point source placed at the origin. The results for the top-hat central weight (shown in Figure 7) show an irregular variation of resolution enhancement. This irregularity is not as marked as when a smooth gaussian central weight is used (Figure 8).

## V. CONCLUSIONS

We have chosen to use the best-linear-estimate method of object reconstruction because it is the simplest recon-

struction algorithm, given prior knowledge in the form of a covariance. We have indicated that this method is equivalent to the minimum norm method with Miller regularisation when the prior distributions are gaussian, and that the singular value decomposition method is negligibly different when the signal-to-noise ratio is large.

We have examined in detail how the best-linear-estimate method reconstructs the object. A weight  $W$  with a large central component and a small extended component was used to calculate the relevant eigenvalues and eigenstates for the object reconstruction process.

We have found that the object basis states recombine the image degrees of freedom to provide an object reconstruction with an enhanced resolution when the Shannon number is low, and that such resolution enhancement is concentrated in those regions where the weight takes its greatest value. For large Shannon numbers the rearrangement of the image degrees of freedom which occurs when the object is reconstructed using the best-linear-estimate formula cannot be interpreted as a local resolution enhancement.

The smaller and brighter the object, the greater the resolution enhancement of an object reconstruction; a bright object dominates the image data, while a small object reduces the possibility of overlap of images of separate parts of the object. The extended weight could be regarded as regularising the reconstruction process by limiting the extent to which we may interpret the image data as having propagated from the region of greater weight.

We have examined a gaussian profile for the central object weight plus a constant extended weight. The smoothness of reconstructions obtained in this way make the gaussian weight (plus extension) a good candidate for ‘first-guess’ analysis of images. Unlike the top-hat weight, the gaussian weight gives rise to a resolution enhancement factor  $E$  which varies smoothly with the imaging parameters.

A more refined form of prior knowledge is specified by allowing  $W$  to be non-diagonal. This type of  $W$  will specify correlations between the values the object reconstruction takes at two different points. Such problems may be analysed in terms of individual object reconstruction states by first diagonalising  $W$ .

## VI. ACKNOWLEDGMENT

The author thanks Drs. A. H. Greenway, C. J. Oliver, J. G. Walker and J. W. Wood, and colleagues at RAE Farnborough, for stimulating discussions during the course of this work.

- 
- [1] J B Abbiss, C de Mol, and H S Dhadwal, *Regularised iterative and non-iterative procedures for object restoration from experimental data*, Optica Acta **30** (1983), no. 1, 107–124.
- [2] J B Abbiss, M Defrise, C de Mol, and H S Dhadwal, *Regularised iterative and non-iterative procedures for object restoration in the presence of noise: an error analysis*, Journal of the Optical Society of America **73** (1983), no. 11, 1470–1475.
- [3] H P Baltes, *Inverse source problems in optics: Topics in current physics*, Springer-Verlag, Berlin, 1978.
- [4] ———, *Inverse scattering problems in optics: Topics in current physics*, Springer-Verlag, Berlin, 1980.
- [5] M Bertero, P Brianzi, P Parker, and E R Pike, *Resolution in diffraction-limited imaging, a singular value analysis III. The effect of sampling and truncation of the data*, Optica Acta **31** (1984), no. 2, 181–201.
- [6] M Bertero, C de Mol, E R Pike, and J G Walker, *Resolution in diffraction-limited imaging, a singular value analysis IV. The case of uncertain localisation or non-uniform illumination of the object*, Optica Acta **31** (1984), no. 8, 923–946.
- [7] M Bertero and E R Pike, *Resolution in diffraction limited imaging, a singular value analysis I. The case of coherent illumination*, Optica Acta **29** (1982), no. 6, 727–746.
- [8] C L Byrne, R M Fitzgerald, M A Fiddy, T J Hall, and A M Darling, *Image restoration and resolution enhancement*, Journal of the Optical Society of America **73** (1983), no. 11, 1481–1487.
- [9] A M Darling, T J Hall, and M A Fiddy, *Stable non-iterative object reconstruction from incomplete data using prior knowledge*, Journal of the Optical Society of America **73** (1983), no. 11, 1466–1469.
- [10] S P Luttrell, *A new method of sample optimisation*, Optica Acta **32** (1985), no. 3, 255–257.
- [11] S P Luttrell and C J Oliver, *Resolution enhancement in coherent images by the introduction of prior knowledge*, Memorandum 3697, Royal Signals and Radar Establishment, Malvern, 1984.
- [12] Optical Society of America, *Topical meeting on signal recovery and synthesis with incomplete information and partial constraints*, Incline Village, 1983.
- [13] Numerical Algorithms Group Staff, *NAG FORTRAN library manual - mark 9*, Numerical Algorithms Group, 1981.
- [14] A N Tikhonov and V Y Arsenin, *Solutions of ill-posed problems*, V H Winston and Sons, Washington DC, 1977.

# A New Method of Sample Optimisation \*

S P Luttrell

*Royal Signals and Radar Establishment, Malvern, Worcestershire, England*

The transinformation measure is used to define the average information supplied by data samples. This is maximised by suitably positioning the samples, which optimises the data collection process. The application of the scheme to a linear imaging system is demonstrated.

## I. LETTER

The purpose of this Letter is to outline a new scheme for optimising the way in which experimental data samples are taken. Hitherto there have been sampling schemes proposed to deal with special cases, such as Nyquist sampling for band-limiting signals, and exponential sampling for Laplace transforms [2]. Here we shall use information theory in order to formulate a more general scheme.

We shall regard an experiment as a mapping  $T$  (which may be nonlinear) from a state  $f$  of the system (s) under study to a state  $g$  of the data samples (d). The mapping will usually involve a random noise element. Let us introduce the a priori system probability distribution function (PDF),  $P_s[f]$ , and the conditional data PDF,  $P_{d|s}[g|f]$ . We may then obtain the unconditional data PDF,  $P_d[g]$ , the a posteriori system PDF,  $P_{s|d}[f|g]$ , and the joint system/data PDF,  $P_{s,d}[f,g]$ .  $P_{s,d}$  summarises our state of knowledge of the current system state  $f$ , bearing in mind the current data observation  $g$  and our a priori knowledge.

An experimenter might reasonably ask where a given number of data samples should be placed, in order that, on average, the maximum possible information is recorded about the current state of the system. We shall use information theory as formulated by Shannon [3] to define a rigorous information measure. The quantity of interest is called transinformation [1], and is defined as

$$I \equiv \int [df] [df] P_{s,d}[f,g] \log_2 \left\{ \frac{P_{s|d}[f|g]}{P_s[f]} \right\} \quad (1.1)$$

Loosely speaking, the larger  $I$  is, the narrower  $P_{s|d}$  is relative to  $P_s$ , and so the surer we are about the system state. More formally  $I$  measures the average increase in the selective information available to the experimenter when the data samples are taken. Clearly maximising  $I$  will lead to the optimum sampling scheme. Note that in maximising  $I$  with respect to the sample positions we modify the ‘recording medium’ which the experimenter uses, rather than modify the code whereby the system

output is transmitted. This latter possibility would require changes in the experimental procedure which are beyond the scope of this note.

We should note that this method of optimising the data sampling scheme is independent of what the experimenter proposes to do with the data. All pertinent a priori knowledge has been fed in by specifying  $P_s$  and  $P_{d|s}$ . Having optimised the data acquisition stage of the experiment, the subsequent data interpretation stage has the greatest amount of useful information to work with. More complicated experiments in which successive data sets are used iteratively to enhance the experimenter’s state of knowledge of the system do not factorise into independent data acquisition and interpretation stages. Currently there is no general theory which can handle all such general adaptive (learning) systems, and we make no attempt to analyse such situations.

The analysis is particularly simple if the mapping  $T$  is linear, and the system state  $f$  and the data state  $g$  are elements of (different) Hilbert spaces. We shall assume that  $P_s$  and  $P_{d|s}$  are gaussian.

$$P_s[f] \equiv \frac{1}{\det[\pi W]} \exp[-f^\dagger W^{-1} f] \quad (1.2)$$

$$P_{d|s}[g|f] \equiv \frac{1}{\det[\pi N]} \exp[-(g - Tf)^\dagger N^{-1} (g - Tf)] \quad (1.3)$$

where the distributions are over complex Hilbert space elements  $f$  and  $g$ , and  $W$  and  $N$  are covariance matrices. In this simple model the data noise is additive with covariance  $N$ . A simple calculation leads to

$$I = \log_2 \left\{ \frac{\det[TW T^\dagger + N]}{\det[N]} \right\} \quad (1.4)$$

The ratio of the determinants measures the average boost in ‘volume’ occupied by the data when there is a signal present, which is directly related to the ability of the experimenter to discriminate which state the system currently occupies.

In order to demonstrate the utility of Equation 1.4 let us analyse the information content of the output of a one-dimensional bandlimiting filter (e.g. a simple lens). Thus

$$T_a(x) \equiv \frac{\sin[c(x_a - x)]}{c(x_a - x)} \quad (1.5)$$

where  $c$  is the bandlimit,  $x$  measures the (continuous) position in ‘object’ space, and  $x_a$  is the position of sample

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This letter appeared in *Optica Acta*, 1985, vol. 32, no. 3, pp. 255-257. Received 9 November 1984. © Controller, Her Britannic Majesty’s Stationery Office, London, 1985.

number  $a$  in ‘image’ space. We shall assume that both the object and the image noise are uncorrelated, so that

$$\begin{aligned} W(x, y) &\equiv W \delta(x - y) \\ W &= \text{constant} \end{aligned} \quad (1.6)$$

$$\begin{aligned} N_{a,b} &\equiv N \delta_{a,b} \\ N &= \text{constant} \end{aligned} \quad (1.7)$$

Then

$$TW T_{a,b}^\dagger = \frac{\pi W}{c} \frac{\sin[c(x_a - x_b)]}{c(x_a - x_b)} \quad (1.8)$$

From these results it follows that  $I$  (see Equation 1.4) is maximised when

$$\begin{aligned} x_a - x_b &= K_{a,b} \frac{\pi}{c} \\ K_{a,b} &= \text{integer} \end{aligned} \quad (1.9)$$

where  $K_{a,b} \neq 0$  for all  $a \neq b$ . The maximum value  $I_{\max}$  is then given by

$$I_{\max} \equiv m \log_2[S + 1] \quad (1.10)$$

where  $m$  is the number of data samples and  $S$  is the signal-to-noise ratio defined as

$$S \equiv \frac{\pi W}{c N} \quad (1.11)$$

Equation 1.9 is reassuring since it takes the form of the usual Nyquist sampling theorem when  $m \rightarrow \infty$ . However, Equation 1.9 goes further by indicating that optimal sampling is achieved when the samples are all separated by multiples of the Nyquist length  $\frac{\pi}{c}$  whatever the total number of samples (though the positioning is not critical when  $S$  is large). Note that this simple result is not guaranteed in more general cases than that assumed in Equation 1.6, Equation 1.7, and Equation 1.8; almost invariably the solution is not analytically solvable.

The most general expression which we have derived for which  $I$  can be calculated easily is given by Equation 1.4, where the mapping was assumed to be linear, and the probability distributions were assumed to take the forms given in Equation 1.2 and Equation 1.3. This expression enables us to calculate  $I$  for many cases of interest; in particular we can explore the effect of limiting the support of the object domain, and other types of prior knowledge. Equation 1.4 has a simple interpretation in terms of the singular value decomposition (SVD) method, which decomposes the imaging system into a number of independent communication channels each of which may be analysed independently of the rest for information carrying capacity. Note that the restrictions which were applied (Equation 1.2 and Equation 1.3) in order to derive Equation 1.4 also apply if the SVD is to be attempted, although this has not always been made entirely clear in the literature. Work in progress is using the transinformation measure to optimise the sampling scheme in a variety of experiments.

- 
- [1] B R Frieden, *Probability, statistical optics, and data testing*, Springer Verlag, Berlin, Heidelberg, New York, 1983.  
[2] N Ostrowsky, D Sornette, P Parker, and E R Pike, *Exponential sampling method for light scattering polydispersity analysis*, Optica Acta **28** (1981), no. 8, 1059–1070.  
[3] C E Shannon and W Weaver, *The mathematical theory of communication*, University of Illinois Press, Urbana, 1949.

# The Use of Transinformation in the Design of Data Sampling Schemes for Inverse Problems \*

S P Luttrell

Royal Signals and Radar Establishment, St Andrews Road, Great Malvern, Worcs, WR14 3PS, UK

We analyse the average useful information content of data samples by using the transinformation entropy (rate of transmission) of Shannon's information theory. We derive a simple expression for the transinformation in linear experiments with gaussian a priori distributions. We use this expression to examine various schemes for sampling the image spaces of a translation invariant (sine) and a conformally invariant (Laplace) mapping. The optimum sampling scheme is found to be considerably better than the naive sampling scheme (e.g. Nyquist) when the number of samples is small and the a priori knowledge is non-trivial.

## I. INTRODUCTION

The mathematical theory of communication as formulated principally by Shannon [15, 16] is directly applicable to the measurement of information acquired by an experimenter in the course of conducting an experiment. We choose the Shannon information measure because it is unique in satisfying the axioms one assigns to additive information. Accordingly we shall regard an experiment as a mapping  $T$  (which may be non-linear) from a state  $f$  of the system ( $s$ ) under study to a state  $g$  of the data samples ( $d$ ). The mapping will usually involve a random noise element. Let us introduce the *a priori* system probability distribution function (PDF)  $P_s[f]$ , and the conditional data PDF  $P_{d|s}[g|f]$ . We may then obtain the unconditional data PDF  $P_d[g]$ , the *a posteriori* system PDF  $P_{s|d}[f|g]$ , and the joint system/data PDF  $P_{s,d}[f,g]$ .  $P_{s,d}$  summarises our state of knowledge of the current system state  $f$  bearing in mind the current data observation  $g$  and our *a priori* knowledge  $P_s[f]$ . Formally the ensemble of possible joint system/data states and the measure of the corresponding joint PDF forms a probability measure space [8]. The joint entropy (measured in bits) of the system and data is defined as [15, 16]

$$H[s, d] \equiv - \int [df] [dg] P_{s,d}[f, g] \log_2 \{P_{s,d}[f, g]\} \quad (1.1)$$

where we are taking  $f$  and  $g$  to be continuous. Note that the argument of the logarithm depends on the density of elementary events. The useful information content of the data is given by the rate of transmission  $I$  (or transinformation; see [9]) defined as [15, 16]

$$I \equiv H[s] - H[s|d] \quad (1.2)$$

$$\equiv H[d] - H[d|s] \quad (1.3)$$

$$\equiv H[s] + H[d] - H[s, d] \quad (1.4)$$

where the entropies  $H[s]$ ,  $H[d]$ ,  $H[s|d]$  and  $H[d|s]$  are obtained from Equation 1.1 by suitably modifying the PDF inside the logarithm.  $I$  does not depend on the density of elementary events [8], as required for a consistent measure of useful information content. The transinformation is the entropy of the source (system) as reduced by the conditional entropy characterising the channel of communication (data acquisition scheme) [9].

The simple situation outlined above assumes that prior knowledge in the form of *a priori* PDFs actually exists. This will be the case if we have a model of the system, and the model parameters ( $f$  in Equation 1.1) are assigned their *a priori* distributions. However, there is a large class of problems in which the roles of *a priori* information and information contained in data are inextricably intertwined. Such situations occur when the model of the system under observation is updated in the light of data already acquired; thus the status of prior data is elevated to that of prior knowledge. Our analysis does not apply to such adaptive models with any rigour.

With these restrictions in mind we shall explore how maximising transinformation optimises data acquisition in a variety of simple situations. In Section II we interpret what such an optimisation achieves. In Section III we introduce a simple model for which the transinformation may be calculated easily, and we show how the singular value decomposition method may be used to analyse this type of transinformation. In Section IV, Section V and Section VI we analyse specific examples where the model of Section III may be applied. The shift-invariant case is studied in Section IV, and its generalisation to include a support constraint on the object is studied in Section V. The (conformally invariant) Laplace mapping with an object support constraint is studied in Section VI.

## II. OPTIMISING DATA ACQUISITION

We have introduced the transinformation  $I$  in order to measure the average additional (useful) information supplied by the sampled output of an experiment. We maximise  $I$  in order to optimise the experiment with respect to the average amount of useful information which is acquired [11]. Note that such an optimisation procedure is

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 10, 2019.

This paper appeared in *Inverse Problems*, 1985, vol. 1, no. 3, pp. 199-218. Received 29 April 1985, in final form 24 May 1985. © 1985 The Institute of Physics.

designed to maximise the amount of useful information as measured with respect to the stated prior knowledge  $P_s[f]$ . If in fact the prior knowledge is incomplete or incorrect, then the optimum data acquisition scheme will not be found. Also, the whole of the data set must be taken into consideration when calculating transinformation. This is important if we make more than one type of measurement on the system, by using a multispectral scanner for instance: the data from all the wavebands must be considered together when calculating the transinformation. The detail of this procedure is highly problem dependent, but the universality of the principle of maximising the transinformation (once it is measured) is preserved. However, we shall see that there are many situations in which this whole procedure may be carried out without difficulty.

We commented in Section I that  $I$  does not depend on the density of elementary events in a continuous probability measure space. The variation of  $I$  with sample position arises because the chosen sampling scheme (which includes the data quantisation scheme, i.e. the discretisation of the data values used by the experimenter) defines a mapping from an infinite-dimensional continuous probability measure space to a finite-dimensional discrete probability measure space. The particular discrete measure which is mapped to, and hence the transinformation, depends on the sampling scheme. Note that we shall assume that the quantisation levels are spaced sufficiently finely that they may be regarded as a continuum of levels, and so the transinformation will be unaffected by the exact details of the density of levels. However, a similar program of transinformation maximisation could be carried out in which the positions of the quantisation levels are varied. Note that maximising transinformation affects only the data acquisition stage of an experiment: it places no constraints on how the data should be interpreted.

It is instructive to interpret what sample position optimisation achieves. One extreme case is when the signal component of the sample values is highly redundant (loosely speaking, the signal is oversampled). Therefore there is not much potential signal information present (undesirable), but it is robust to the disruptive effects of data noise (desirable). At the other extreme we could place the samples so that the signal component of the sample values has very little redundancy. Therefore there is a lot of potential signal information present (desirable), but it is not robust to the disruptive effects of noise (undesirable). We would like to have the advantages of both sampling schemes, but sadly they are mutually exclusive; we must accept a trade-off between the two extremes. We achieve this by maximising the useful Shannon information contained in the data samples, which may be deduced from  $P_s[f]$  and  $P_{d|s}[g|f]$ . This requires that sample positions must be chosen to ‘match’ the prior knowledge. The basic inequalities which summarise the effect of redundancy (and/or independence) on information content

are [15, 16]

$$H[x] \text{ and } H[y] \leq H[x, y] \leq H[x] + H[y] \quad (2.1)$$

where  $x$  and  $y$  are information sources. The left-hand inequality tells us that together  $x$  and  $y$  cannot supply information at a lower rate than either  $x$  or  $y$  do separately: equality occurs when  $x$  and  $y$  are completely redundant. The right-hand inequality tells us that together  $x$  and  $y$  cannot supply information at a greater rate than the sum of their separate rates: equality occurs when  $x$  and  $y$  are completely independent.

The dimensionality  $D$  (or number of degrees of freedom) of a data set is a figure which is often quoted as a measure of its information content. The dimensionality is the number of basic data coordinates (dimensions) which are independent; there are many empirical ways in which we can define such a measure. However, we shall define dimensionality  $D(m)$  for  $m$  samples as the ratio of the maximum transinformation that is obtainable with  $m$  samples to the maximum transinformation  $I(1)|_{\max}$  that is obtainable with a single sample

$$D(m) = \frac{I(m)|_{\max}}{I(1)|_{\max}} \quad (2.2)$$

This may be interpreted as the effective number of data samples which are collecting transinformation. However, note that in general the transinformation is distributed over all the samples, and so one cannot simply discard some samples if the dimensionality is substantially less than the actual number of samples; one must re-optimize the positions of the remaining samples.

We should mention that maximising transinformation has nothing at all to do with the maximum entropy method (MEM) of deconvolving data. We wish to acquire data samples which give rise to the maximum amount of useful information (on average), and hence give rise to the most useful *a posteriori* PDF over system states (on average). On the other hand, the MEM goes on to deconvolve the data by selecting a single-system state from the set of states (the feasible set) allowed by the *a posteriori* PDF. In order to make such a selection the MEM measures an entropy of each member of the feasible set, and selects the state which has the maximum such entropy. Thus we maximise the transinformation in order to minimise the size of the feasible set (on average). Then one might use the MEM in order to select a particular member of the feasible set which corresponds to each particular set of data.

### III. A SIMPLE EXPERIMENTAL MODEL

The purpose of this paper is to demonstrate the usefulness of the transinformation measure as a unified construct which may be used to analyse data sampling schemes. In general it is very difficult to obtain a useful analytic expression for the transinformation, so we usually have to resort to numerical methods to generate the

relevant PDFs and entropies. However, we shall adopt a simple model of the prior PDFs and the experimental  $T$  which is analytically tractable.

The analysis is particularly simple [11, 12] if  $T$  is linear, and the system state  $f$  is an element of a Hilbert space  $F$ , and the data state  $g$  is an element of a (finite- or infinite-dimensional) Hilbert space  $G$ . Thus the experiment takes the form

$$g = T f + n \quad (3.1)$$

where  $n$  is additive data noise in  $G$  which we assume to be distributed independently of the signal. The simplest prior PDFs which give non-trivial results are gaussian; these may be used to specify means and covariance alone. Accordingly we shall define  $P_s$ , and  $P_{d|s}$  as

$$P_s[f] = \frac{1}{\det[\pi W]} \exp \left[ - (f - f_0)^\dagger W^{-1} (f - f_0) \right] \quad (3.2)$$

$$P_{d|s}[g|f] = \frac{1}{\det[\pi N]} \exp \left[ - (g - T f)^\dagger N^{-1} (g - T f) \right] \quad (3.3)$$

where  $W$  and  $N$  are covariance matrices

$$W \equiv \langle (f - f_0) (f - f_0)^\dagger \rangle \quad (3.4)$$

$$N \equiv \langle n n^\dagger \rangle \quad (3.5)$$

and  $f_0$  is a mean

$$f_0 \equiv \langle f \rangle \quad (3.6)$$

The choice of  $P_{d|s}$  in Equation 3.3 then specifies that the data have additive, zero mean, correlated gaussian noise present. The position of the mean of the additive data noise does not affect the information content of the data, and so we assume it to be zero for simplicity. We have modelled  $P_s$  in Equation 3.2 by using a gaussian PDF, which is more *ad hoc* than modelling data noise as gaussian random variables. However, if we have the prior knowledge that the possible system states  $f$  are clustered around state  $f_0$  and that the covariance of this cluster is  $W$ , then the chosen  $P_s$  is the least committal model [10]. The unconditional data PDF  $P_d[g]$  may be derived as

$$P_d[g] = \frac{1}{\det[\pi \bar{W}]} \exp \left[ - (g - g_0)^\dagger \bar{W}^{-1} (g - g_0) \right] \quad (3.7)$$

where  $\bar{W}$  is the covariance matrix

$$\begin{aligned} \bar{W} &\equiv \langle (g - g_0) (g - g_0)^\dagger \rangle \\ &= T W T^\dagger + N \end{aligned} \quad (3.8)$$

and  $g_0$  is the mean

$$\begin{aligned} g_0 &\equiv \langle g \rangle \\ &= T f_0 \end{aligned} \quad (3.9)$$

Again the position of the mean of  $P_d$  does not affect the information content of the data, and so we could have set  $f_0 = 0$  in Equation 3.2.

We shall use Equation 1.3 to calculate  $I$ . The entropies  $H[d]$  and  $H[d|s]$  are given by

$$H[d] = \log_2(e) + \log_2(\det[\pi T W T^\dagger + \pi N]) \quad (3.10)$$

$$H[d|s] = \log_2(e) + \log_2(\det[\pi N]) \quad (3.11)$$

which gives the following expression for the transinformation:

$$I = \log_2 \left( \frac{\det[T W T^\dagger + N]}{\det[N]} \right) \quad (3.12)$$

This is the central theoretical result [11] from which we shall calculate the transinformation in a variety of situations. The interpretation of Equation 3.12 is that the ratio of the determinants gives the average boost in the volume of data space within which the data can be found when the signal  $T f$  is added to the data noise  $n$ . This ratio is directly related to the ability of the experimenter to discriminate which state  $f$  the system currently occupies. We could have anticipated this interpretation of  $I$  on more general grounds from Equation 1.3. If the noise entropy  $H[d|s]$  is held constant, then  $I$  increases with  $H[d]$ . However,  $H[d]$  is a measure of (the logarithm of) the occupied volume of data space, and so the form of Equation 3.12 is not surprising.

In the following sections we shall obtain some results for particular types of  $T$ . We shall assume there that the additive data noise is an isotropic tensor

$$\begin{aligned} N_{a,b} &\equiv N \delta_{a,b} \\ N &= \text{constant} \end{aligned} \quad (3.13)$$

so that Equation 3.12 may be written in the form

$$I \equiv \log_2(\det[M]) \quad (3.14)$$

where

$$M_{a,b} \equiv R_{a,b} + \delta_{a,b} \quad (3.15)$$

$R_{a,b}$  is the covariance matrix of the signal component of the data with the convention that the data are scaled so that the additive white noise has unit variance. Some expressions for transinformation which are derived from these formulae are contained in Appendix VII. We shall use the following factorisation of  $R_{a,b}$

$$R_{a,b} \equiv R r_{a,b} \quad (3.16)$$

where  $R$  is a suitably defined signal-to-noise ratio (SNR).

We shall now examine how the expression for  $I$  given in Equation 3.12 may be decomposed into a sum over independent contributions to the transinformation. To achieve this we need to diagonalise simultaneously the matrices  $T W T^\dagger + N$  and  $N$ ; this is just a simultaneous

Karhunen-Loëve expansion. Let the diagonal elements of the matrices be given by

$$T W T^\dagger + N \longrightarrow \text{diag} \{ \lambda_1, \lambda_2, \dots \} \quad (3.17)$$

$$N \longrightarrow \text{diag} \{ \nu_1, \nu_2, \dots \} \quad (3.18)$$

then  $I$  may be expressed as

$$I \equiv \sum_a I_a \quad (3.19)$$

where

$$I_a \equiv \log_2 \left( \frac{\lambda_a}{\nu_a} \right) \quad (3.20)$$

and  $a$  labels the transformed data states. In general these states are not orthogonal and so we must be careful to use distinct covariant and contravariant components. However, the states do carry independent transinformation  $I$ , and so we have decomposed the mapping  $T$  into a number of independent effective communication channels.

This analysis formalises the rationale behind part of the singular-value decomposition (SVD) method of data interpretation; namely the data decomposition step of the SVD. The SVD may be used to perform the diagonalisation in Equation 3.17 and Equation 3.18. Such an analysis has been performed for the sine imaging operation [4, 6, 7, 12], for the (sinc)<sup>2</sup> imaging operation [3], and for the Laplace transform [1, 2, 5, 13]. The theoretical tools from which all these applications may be derived are contained in reference [12]. We assumed a specific pair of distributions  $P_s$  and  $P_{d|s}$ , and we assumed a linear mapping  $T$  in order to derive Equation 3.12; in general the result is not so simple. The SVD rigorously decouples the data space  $G$  into pieces carrying independent transinformation if, and only if, the two-point correlation function (and the mean) completely specifies the statistical properties of the data, i.e. when all higher-order correlations may be deduced from the two-point correlations. Thus the data PDFs (signal and signal with noise) must be gaussian for the SVD to provide a decomposition into independent partial transinformations. The basic reason for this restriction is that the SVD, which is a linear transformation, can diagonalise only two-point correlations.

The SVD is also used extensively (same references as the previous paragraph) to provide a means of inverting data to recover the object under observation: this corresponds to completing the interpretation of the data, which began with a decomposition using the SVD as discussed above. The object and image singular functions (or states) and the corresponding singular values provide a means for tracing how information passes through  $T$ . For situations where the SVD provides a rigorous decomposition of the transinformation (i.e. gaussian *a priori* PDFs) the object reconstruction which is obtained is the one which has the maximum *a posteriori* probability density [12]. For situations where the *a priori* PDFs

are not all gaussian, the separate singular functions do not in general contain independent information, and the object reconstruction does not have such a simple interpretation. Nevertheless, the SVD method proves to be a very powerful analytical tool, as long as results obtained by using it are interpreted with care.

#### IV. SHIFT-INVARIANT MAPPING

Image formation may be modelled, to a first approximation, by a shift-invariant (isoplanatic) linear mapping  $T$ . For simplicity we shall restrict our attention to one-dimensional problems with a continuous object coordinate  $x$  where  $-\infty < x < +\infty$  and a discrete image coordinate  $x_a$  where  $a = 1, 2, \dots, m$  (i.e.  $m$  data samples). Furthermore, we shall assume that the covariance matrices  $W$  and  $N$  (Equation 3.4 and Equation 3.5) are isotropic tensors

$$\begin{aligned} W(x, y) &\equiv W \delta(x - y) \\ W &= \text{constant} \end{aligned} \quad (4.1)$$

$$\begin{aligned} N_{a,b} &\equiv N \delta_{a,b} \\ N &= \text{constant} \end{aligned} \quad (4.2)$$

The matrix elements of  $T W T^\dagger$  are then given by

$$\begin{aligned} T W T_{a,b}^\dagger &\equiv W \int_{-\infty}^{+\infty} dz T(x_a - z) T^*(x_b - z) \quad (4.3) \\ &= \frac{W}{2\pi} \int_{-\infty}^{+\infty} dk \exp[i k (x_a - x_b)] |\hat{T}(k)|^2 \end{aligned}$$

where  $\hat{T}$  is the Fourier transform of  $T$  given by

$$\hat{T}(k) \equiv \int_{-\infty}^{+\infty} dz \exp[-ikz] T(z) \quad (4.4)$$

Note that  $T W T_{a,b}^\dagger$  in Equation 4.3 is a function of the difference in coordinates,  $x_a - x_b$ , alone, and so the transinformation in Equation 3.12 is unchanged if all the data sample positions are displaced by the same amount. We shall define the signal-to-noise ratio of the system  $R$  as the ratio of the expected signal energy (per sample) to the expected noise energy (per sample):

$$R \equiv \frac{W}{2\pi N} \int_{-\infty}^{+\infty} dk |\hat{T}(k)|^2 \quad (4.5)$$

We shall also define the covariance matrix of the signal component of the data normalised to unit signal energy per sample

$$r_{a,b} \equiv \frac{\int_{-\infty}^{+\infty} dk \exp[i k (x_a - x_b)] |\hat{T}(k)|^2}{\int_{-\infty}^{+\infty} dk |\hat{T}(k)|^2} \quad (4.6)$$

We may now use Equation 3.14, Equation 3.15 and Equation 3.16 to calculate the transinformation. The maximum possible range for  $I$  is

$$\log_2(mR + 1) \leq I \leq m \log_2(R + 1) \quad (4.7)$$

where the lower limit is obtained when the signals in the samples are completely redundant, and the upper limit is obtained when they are completely independent. To achieve the maximum possible transinformation we must place the samples in such a way that all their separations lie at zeros of the signal covariance (see Equation 3.14); this is not usually possible.

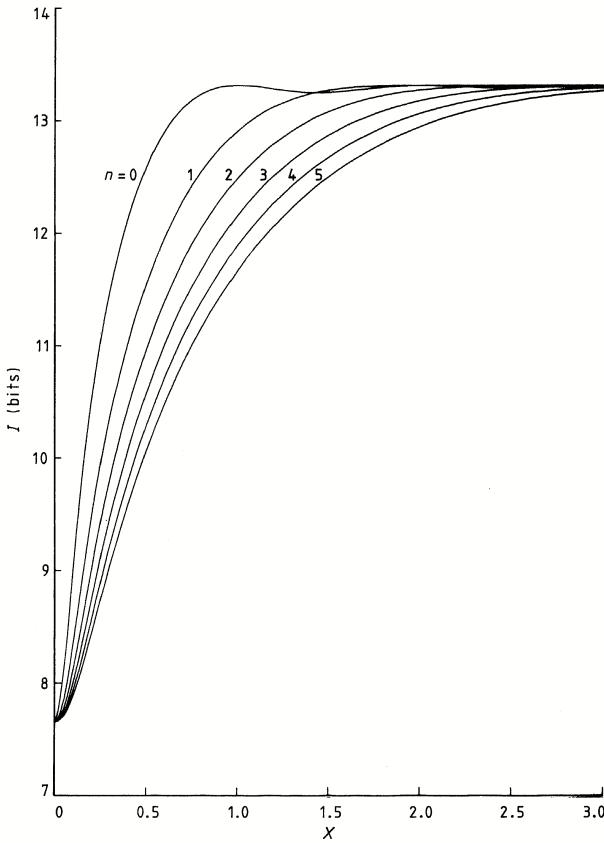


Figure 1: Plots of  $I$  (in bits) for a two-sample image against sample separation  $X$  (in Nyquist lengths) for the cosine weighted filters of Equation 4.9.  $n = 0, 1, 2, 3, 4$  and  $5$  are depicted for a SNR  $R = 100$ .

The most common special case is when  $|\hat{T}|$  takes the form

$$|\hat{T}(k)| = \begin{cases} 1 & |k| \leq c \\ 0 & |k| > c \end{cases} \quad (4.8)$$

In this case the samples can all be placed so that their separations correspond to zeros of the signal covariance, and so the upper limit in Equation 4.7 can be attained. This requires that all the intersample spacings be integer multiples of  $\frac{\pi}{c}$  (the Nyquist length) [11]. Note that

this result is true for any number of samples. However, it is important to note the assumptions which were used (Equation 3.1, Equation 3.2, Equation 3.3, Equation 4.1, Equation 4.2 and Equation 4.8) in order to derive this result. If any of these conditions is violated then the Nyquist sampling scheme does not achieve the upper limit in Equation 4.7, and more importantly it does not maximise the transinformation in general.

A useful class of  $T$  is given by the cosine weighted filters

$$|\hat{T}(k)| = \begin{cases} \cos^n\left(\frac{\pi k}{2c}\right) & |k| \leq c \\ 0 & |k| > c \end{cases} \quad (4.9)$$

Equation 4.8 is a special case of Equation 4.9 with  $n = 0$ .  $r_{a,b}$  is then given by

$$r_{a,b} \equiv (n!)^2 \sin[c(x_a - x_b)] \times \sum_{j=-n}^{+n} \frac{(-1)^j}{(n-j)! (n+j)!} \frac{1}{c(x_a - x_b) - j\pi} \quad (4.10)$$

In Figure 1 we show how  $I$  varies with the separation of the samples for the case  $m = 2$  and  $R = 100$ . Note how the transinformation decreases as  $n$  increases for a fixed sample separation. This is clearly a result of the progressively more drastic tapering of  $\hat{T}$  as  $n$  increases, which removes the higher frequencies from the response. As a consequence the samples have to be moved further apart in order to retain the same degree of independence.

## V. RESTRICTED OBJECT SUPPORT FOR THE SINC MAPPING

In the previous section we derived some results for the transinformation obtained from a sampled image space of an isoplanatic mapping  $T$  where the support of the continuous object space was assumed to be infinite. However, in practice we are more interested in situations where we have available some further constraints (prior knowledge) on the form of the object. Under such circumstances there is the possibility of obtaining a large gain in transinformation by optimising the sample positions. In order to keep the analysis as simple as possible we shall restrict our attention to the case studied in Equation 4.8, but with the additional restriction that the object support be limited. The generalisation to an arbitrary  $W$  and  $N$  is obvious.

Let us restrict the object support to the interval  $[-1, +1]$ , then  $TWT^{\dagger}$  is given by

$$TWT_{a,b}^{\dagger} \equiv W \int_{-\infty}^{+\infty} dz \frac{\sin[c(x_a - z)]}{\pi(x_a - z)} \frac{\sin[c(x_b - z)]}{\pi(x_b - z)} \quad (5.1)$$

which should be compared with Equation 4.3. Introducing the prolate spheroidal functions  $\psi_k(z; c)$  [17] which

have the properties

$$\sum_{k=0}^{\infty} \psi_k(x; c) \psi_k(y; c) = \frac{\sin [c (x - y)]}{\pi (x - y)} \quad (5.2)$$

$$\int_{-1}^{+1} dx \psi_k(x; c) \psi_l(x; c) = \lambda_k(c) \delta_{k,l} \quad (5.3)$$

allows us to write  $T W T^\dagger$  as

$$T W T_{a,b}^\dagger = W \sum_{k=0}^{\infty} \lambda_k(c) \psi_k(x_a; c) \psi_k(x_b; c) \quad (5.4)$$

We shall define the SNR as the ratio of the expected signal energy to the expected noise energy in a sample placed at the origin, which gives

$$R \equiv \frac{W}{N} \sum_{k=0}^{\infty} \lambda_k(c) (\psi_k(0; c))^2 \quad (5.5)$$

We may use Equation 3.14, Equation 3.15 and Equation 3.16 to calculate  $I$  if we define the normalised covariance  $r_{a,b}$  as

$$r_{a,b} \equiv \frac{\sum_{k=0}^{\infty} \lambda_k(c) \psi_k(x_a; c) \psi_k(x_b; c)}{\sum_{k=0}^{\infty} \lambda_k(c) (\psi_k(0; c))^2} \quad (5.6)$$

These matrix elements  $r_{a,b}$  depend on a single dimensionless combination of the object support width and system bandwidth  $c$ . This is the well known Shannon number  $S$  which is the object support width measured in Nyquist lengths.  $S$  is given by the expression

$$S \equiv \frac{2c}{\pi} \quad (5.7)$$

for the support  $[-1, +1]$ .

We shall now select a simple class of image sampling schemes in order to study how the transinformation varies with Shannon number and sample position. For  $m$  data samples there are  $\frac{m}{2}$  (e.g.  $\frac{3}{2} = 1$ ) independent sample positions if we assume that the samples are to be symmetrically placed either side of the origin. We shall denote these positions  $0, \pm x_1, \pm x_2, \dots, \pm x_{\frac{m}{2}}$  (omitting zero when  $m$  is even). We can display  $I$  graphically for  $\frac{m}{2} = 1$  and  $\frac{m}{2} = 2$  alone; we shall choose  $\frac{m}{2} = 2$  since this will show the greatest wealth of structure.

In Figure 2 we show contour plots of  $I$  against  $(x_1, x_2)$  for various  $S$  and for  $R = 100$ , and we have marked the optimum configurations with a cross. The sample positions are measured in Nyquist lengths, and the contour plots are symmetric because the data space is unchanged by swapping  $x_1$  and  $x_2$ . For large Shannon numbers (Figure 2(d)) the optimal scheme is very similar to the Nyquist scheme, but there is a marked departure from the Nyquist scheme at low Shannon numbers. The optimum may be explained as the result of an attempt by the samples to position themselves so that they receive a large

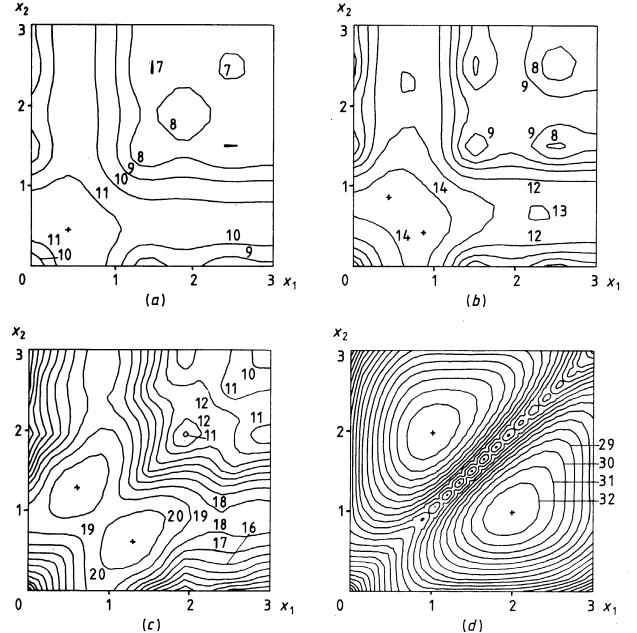


Figure 2: Contour plots of  $I$  (in bits) against the two independent positions,  $x_1$  and  $x_2$  (in Nyquist lengths) of a five-sample image. The sinc filter of Equation 4.8 has been used. Parts (a), (b), (c) and (d) depict  $I$  for Shannon numbers  $S = 0.5, 1, 2$  and  $5$ , respectively, for  $R = 100$ . The point(s) where  $I$  has its maximum value is (are) marked +.

amount of signal energy whilst at the same time sampling the signal at points which are as independent as possible. For large Shannon numbers the signal energy does not vary much with sample position, and so the sample values seek only to be independent: this leads to Nyquist sampling as the optimal scheme. For small Shannon numbers the signal energy is predominantly contained within a Nyquist length of the origin, and so the optimum position is a trade-off between maximising the signal energy and minimising the signal redundancy.

An interesting feature of Figure 2 is that the optimum sample positions are not always separate (Figure 2(a)). The meaning of two (or more) coincident samples is that the data measurement is taken two (or more) times. The signal is the same in each such measurement, but the noise is distributed according to its PDF each time. Therefore such coincident samples enhance the signal energy with respect to the noise energy at the expense of measuring redundant signal information. Clearly such a situation becomes more attractive as the original SNR is decreased, and we should expect that the optimal sample positions will coalesce progressively. For a small enough SNR the situation is complementary to the case of the Nyquist limit discussed above, since the samples would then seek only to maximise the signal energy. In practice there are constraints on the allowed relative position of the samples, and on the amount of signal energy which each sample can acquire when in competition with

other samples. However, the principle of transinformation maximisation is not affected by such considerations.

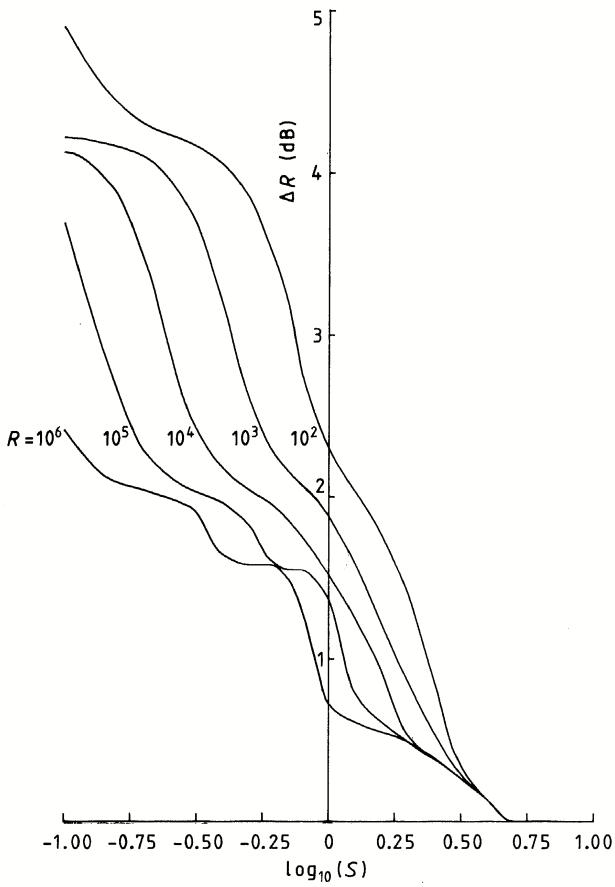


Figure 3: Plots of the effective SNR gain  $\Delta R$  (in dB) against  $\log_{10}(S)$  for  $R = 10^2$  (20 dB),  $10^3$  (30 dB),  $10^4$  (40 dB),  $10^5$  (50 dB) and  $10^6$  (60 dB). The sinc filter of Equation 4.8 has been used.

In Figure 3 we display the increase  $\Delta R$  in SNR which is required in order that the Nyquist sampling scheme (with samples centred on the object support) has a transinformation equal to that achieved by the optimum sampling scheme. Thus  $\Delta R$  is a measure of the effective power gain obtained by using the optimum scheme instead of the Nyquist scheme. No significant enhancement in the sampling scheme can be obtained when the Shannon number is greater than about five (the chosen number of samples). This is because when the Shannon number is large the support constraint does not significantly constrain the data. In the opposite limit as the Shannon number approaches zero the effective power gain approaches five (the number of samples). This is because the optimal scheme then has all the samples placed at the origin, which leads to a transinformation of  $\log_2(m R + 1)$  bits, whereas the Nyquist sampling scheme leads to a transinformation of  $\log_2(R + 1)$  bits because only the sample at the origin receives any signal. Of course such a superpo-

sition of samples could lead to a depletion of the signal energy available to each sample in practice. If the signal energy which would be available to a single sample is shared out (unequally) amongst  $m$  coincident samples each with the original noise level, there is no increase in the transinformation. This completely removes the advantage of multiple sampling when the samples compete for signal energy. If such competition is only partial then the transinformation will be depressed from its nominal value (for non-competing samples). In practice the extent to which samples compete for signal energy is usually arranged to be minimal. Thus the problem of superimposed samples is obviously alleviated by taking sequential samples at the same point (in order to allow more signal energy to arrive). Another intuitively appealing result which can be read from Figure 3 is that  $\Delta R$  increases as  $R$  is decreased (for a constant  $S$ ). Thus the extent to which sample position optimisation helps is greater the smaller the amount of signal energy present. More generally we observe that in order to preserve information pertaining to weak signals in a (hypothetical) continuous sampling scheme, we must be very careful which pieces of data we retain (i.e. which discrete sampling scheme we use). The uneven shape of the curves in Figure 3 is caused by secondary maxima in the transinformation surface (see Figure 2) falling close to the Nyquist sample point(s). This causes the Nyquist scheme to be fortuitously good, and so the optimisation gain is reduced in comparison with what it would have been had the secondary maxima been absent.

In Figure 4 we show how the maximum transinformation depends on the number of samples for  $S = 1$  and various SNR. We have normalised  $I$  to be measured as a dimensionality  $D(m)$  (defined in Equation 2.2). For a given SNR each successive sample increases  $D(m)$  by less than its predecessor. For a given  $m$ ,  $D(m)$  increases as the SNR increases. The upper limit to  $D(m)$  which is suggested by these graphs is in accord with the number of components which may be extracted from (hypothetical) continuous data by using the SVD [7]. Note that this comparison is *ad hoc* in so far as the SNR in the continuous case is defined in terms of energies *per unit length*, and not energies *per sample* as in the discrete case. However, we may identify an approximate correspondence between these two cases which justifies the comparison. We have not increased the number of samples to the point where the optimum solution requires that samples be superimposed, so our results do not require an analysis of how samples may compete.

## VI. LAPLACE MAPPING

We shall now analyse the transinformation content of the data samples acquired in experiments which are described by the Laplace transform mapping [2, 13]. The

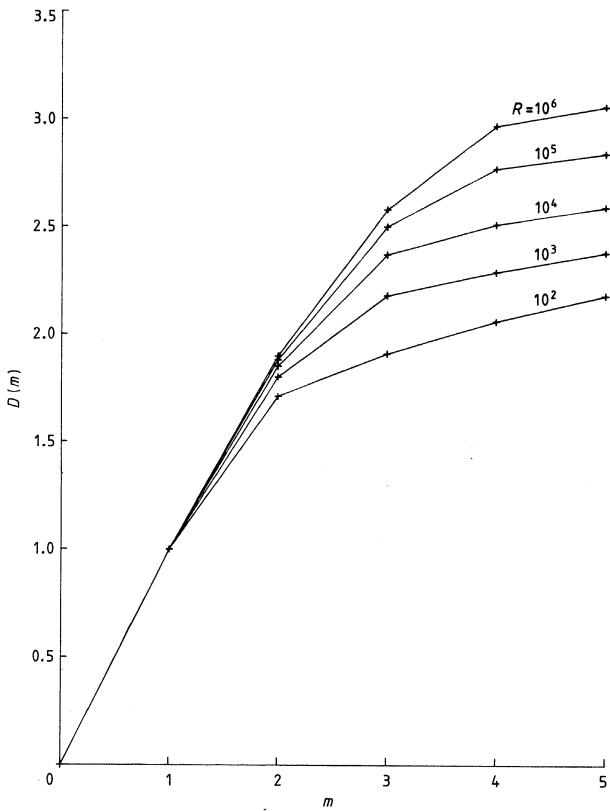


Figure 4: Plots of the effective dimensionality  $D(m)$  against the number of data samples  $m$  for various SNRs and  $S = 1$ . The sinc filter of Equation 4.8 has been used.

mapping  $T$  is then given by

$$T(x_a, z) \equiv \exp(-x_a z) \quad 0 \leq x_a < \infty \quad 0 \leq A \leq z \leq B < \infty \quad (6.1)$$

The Laplace transform has a conformal invariance which will be used later to simplify the results. We shall assume that the forms of  $W(x, y)$  and  $N_{a,b}$  are as in Equation 4.1 and Equation 4.2. The matrix elements of  $T W T^\dagger$  are then given by

$$T W T_{a,b}^\dagger = \frac{W}{x_a + x_b} (\exp[-(x_a + x_b) A] - \exp[-(x_a + x_b) B]) \quad (6.2)$$

$I$  has the following two limits:

$$I \rightarrow \log_2(2R + 1) \text{ as } X \rightarrow 0 \quad (6.10)$$

$$I \rightarrow \log_2(R + 1) \text{ as } X \rightarrow \infty \quad (6.11)$$

Let us define the signal-to-noise ratio  $R$  as the ratio of the expected signal energy to the expected noise energy in a sample at  $x_a = 0$ . This gives

$$R \equiv \frac{(B - A) W}{N} \quad (6.3)$$

We may use Equation 3.14, Equation 3.15 and Equation 3.16 to calculate  $I$  if we define the normalised covariance matrix  $r_{a,b}$  as

$$r_{a,b} \equiv \frac{\exp[-(x_a + x_b) A] - \exp[-(x_a + x_b) B]}{(x_a + x_b)(B - A)} \quad (6.4)$$

$I$  has a conformal invariance under the following scale transformation:

$$\begin{aligned} A &\longrightarrow \beta A \\ \text{and } B &\longrightarrow \beta B \\ \text{and } x_a &\longrightarrow \frac{x_a}{\beta} \text{ for all } a \\ \text{and } R &\text{ held constant} \end{aligned} \quad (6.5)$$

Therefore the optimum sampling scheme depends on  $R$  and  $\frac{B}{A}$  alone, although the  $x_a$  must be expressed in units of  $\frac{1}{A}$  (or  $\frac{1}{B}$ ). As in reference [2] we shall define the ratio  $\gamma$  by

$$\gamma \equiv \frac{B}{A} \quad (6.6)$$

When the data space has a single sample then  $I$  is given by

$$I \equiv \log_2 \left( \frac{R}{2x_1(B - A)} (\exp(-2x_1 A) - \exp(-2x_1 B)) \right) \quad (6.7)$$

$I$  decreases monotonically as  $x_1$  is increased, so the optimum choice for  $x_1$  is zero. Obviously this is because the data acquire the greatest amount of signal energy for this choice of sample position. In fact, however many samples there are, at least one of them must be placed at zero in order to maximise  $I$ . Thus we shall assume  $x_1 = 0$  in the following calculations.

When there are two samples let us define the reduced position  $X$  of the second sample as

$$X = x_2 A \quad (6.8)$$

Then  $I$  may be written as

---


$$I \equiv \log_2 \left| \frac{R + 1}{\frac{R}{X(\gamma-1)} (e^{-X} - e^{-X}\gamma)} \frac{1 + \frac{R}{X(\gamma-1)} (e^{-X} - e^{-X}\gamma)}{1 + \frac{R}{2X(\gamma-1)} (e^{-2X} - e^{-2X}\gamma)} \right| \quad (6.9)$$


---

where intersample competition as  $X \rightarrow 0$  has been ig-

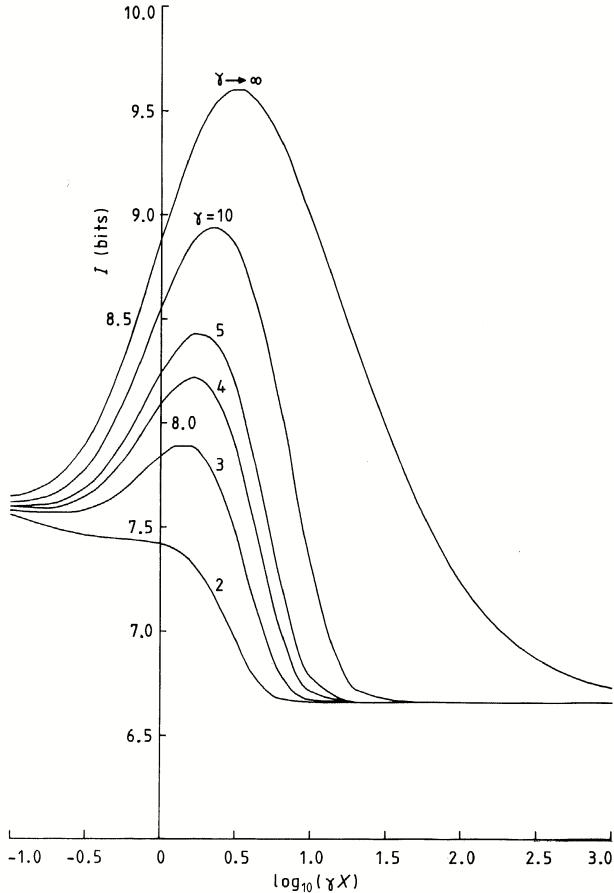


Figure 5: Plots of  $I$  (in bits) for a two-sample image against  $\log_{10}(\gamma X)$  for the Laplace filter of Equation 6.1. The first sample is placed at the origin, and  $X$  is the position of the second sample in inverse units of the lower limit of the object support.  $\gamma = 2, 3, 4, 5, 10$  and  $\gamma \rightarrow \infty$  are depicted for  $R = 100$ .

nored. In Figure 5 we show  $I$  plotted against  $X$  for  $R = 100$  and various values of  $\gamma$ . The graphs show that provided  $\gamma$  is large enough then there is a non-zero  $\gamma X$  which maximises  $I$ . Clearly if  $X \rightarrow \infty$  then the second sample is wasted since it measures nothing but noise (Equation 6.11). On the other hand, as  $X \rightarrow 0$  the signals measured by the two samples are completely redundant. Between these limits of  $X$  there is an optimum choice which is a trade-off between maximising the signal energy and minimising the signal redundancy. However, for a small enough  $\gamma$  (for a fixed SNR) the optimum choice is  $X = 0$ , where we have ignored the effect of intersample competition.

In order to make contact with the suggested exponential sampling method for Laplace transforms [14], we display the optimum sample positions as functions of  $\gamma$  for  $R = 10^6$  in Figure 6. For a fixed  $\gamma$  we have used the largest number of samples for which the optimum positions are all distinct; thus there are several regions within

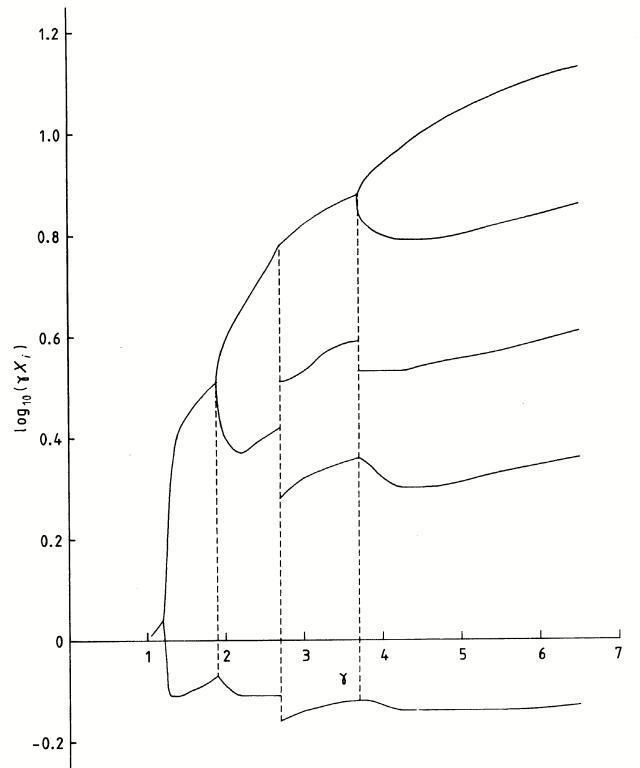


Figure 6: Plots of the optimum sample positions against  $\gamma$  for the Laplace filter and  $R = 10^6$ . For each  $\gamma$  the maximum possible number of samples has been used (see the text).

each of which the number of samples remains fixed. As  $\gamma$  is increased the dimensionality of the data increases, and so a greater number of samples is permitted. The most important feature of this graph is that the separation of the logarithms of the sample positions is approximately uniform; this is especially true at large  $\gamma$ . Thus the optimum data acquisition scheme requires that the samples be approximately exponentially spaced in accord with reference [14]. The exact choice of sample position depends on the prior knowledge which has been incorporated systematically into the transinformation.

In Figure 7 we show how the maximum transinformation (scaled to a dimensionality  $D(m)$ ) depends on the number of samples for  $\gamma = 5$  and various SNR. This figure is completely analogous to Figure 4 in all other respects, and the apparent upper limits to  $D(m)$  in Figure 7 are consistent with the results of the SVD analysis of (hypothetical) continuous image data in reference [2]. Note that the same difficulties with relating a discrete SNR to a continuous SNR arise in this case as in the sinc case of Section V.

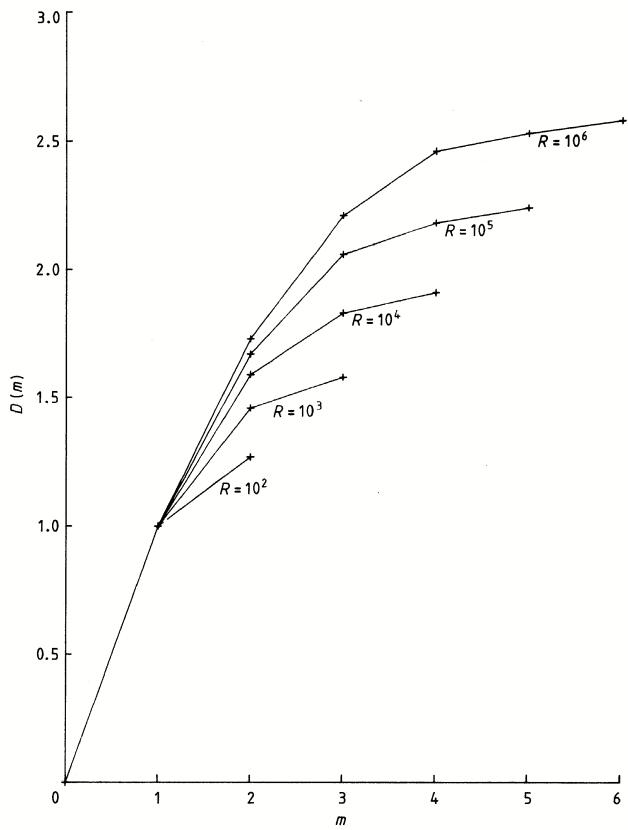


Figure 7: Plots of the effective dimensionality  $D(m)$  against the number of data samples  $m$  for various SNRs and  $\gamma = 5$ . The Laplace filter of Equation 6.1 has been used.

## VII. CONCLUSIONS

Transinformation is the unique measure of additive information which is contained in a data set when prior knowledge is expressed in the form of PDFs. This quantity depends on both the prior knowledge and the way in which the data are acquired. We have restricted our attention to linear  $T$  and to prior knowledge which can be expressed as gaussian PDFs, although the theory is completely general. Such situations have a transinformation which may be expressed in the form given by Equation 3.12. This form may be re-expressed as a sum of independent transinformations (Equation 3.19), which justifies the SVD method of data decomposition for gaussian prior PDFs.

The data themselves are a discrete representation of a continuous function, and the transinformation depends both on the sample positions and the sample value quantisation methods employed. We have assumed that the quantisation levels are of sufficient number and are sufficiently finely spaced that they constitute a continuum of levels, and so do not affect the transinformation. Instead we have investigated the dependence of the transinformation on the sample positions when there are only a

few samples available; this is a realistic experimental situation. For linear  $T$  and gaussian prior PDFs we have expressed the transinformation in terms of a normalised data covariance matrix, and also in terms of the intensity moments of the corresponding gaussian PDF (see Appendix VII). The numerical results which we have obtained from these expressions are in accord with our intuitive notions of how an information measure should behave.

We studied shift-invariant filters in Section IV and Section V. In Figure 1 we observed that when an object of infinite support is observed with a shift-invariant filter whose image space is sampled at two points, then the transinformation reflects the fact that the correlation length of the image increases as the filter is tapered more strongly. Thus the two samples must be separated by a distance greater than of order of the correlation length in order that the signal redundancy is reduced to a negligible level. For the sinc filter the optimum sample positions are always separated by integer multiples of the Nyquist length whatever the number of samples (though note the restrictions mentioned after Equation 4.8). In Figure 2 we observed how a non-trivial form of prior knowledge, a support constraint on the object, modifies the optimum sample positions for a sine filter. The optimum positions are modified progressively as the prior knowledge becomes stronger (Shannon number decreases). We may interpret the modification as arising through a trade-off between two conflicting conditions which the data samples are trying to satisfy: minimising the redundancy of the signal component of the data, and maximising the signal-to-noise ratio in each sample value. In Figure 3 we have shown the increase in the effective SNR which is obtained when one chooses the optimal sampling scheme instead of the Nyquist sampling scheme for the sine filter. The gain in effective SNR increases as the Shannon number decreases. We may interpret this gain as arising from the assistance which the prior knowledge provides, and we could use the extent of the gain to define the strength of the prior knowledge. The gain exists because the Nyquist sampling scheme is optimal only for the limiting case when the support is of infinite extent. In Figure 4 we have shown how the dimensionality of the data increases as we increase the number of optimally placed samples, until it attains a ceiling beyond which no more transinformation is obtainable. This dimensionality also increases with SNR for a fixed number of optimally placed samples, as expected. Our dimensionality results are consistent with published results for the number of significant data components (i.e. the size of the signal space) which may be extracted by using the SVD.

We studied the Laplace transform in Section VI. The Laplace transform is an example of a conformally invariant mapping; there is a choice of scalings of the variables which leaves the kernel invariant. In order to maximise the transinformation we require that at least one data sample is located at zero (of the Laplace transform vari-

able). In Figure 5 we have shown how the transinformation for a two-sample data space varies as the position of the second sample is varied. As for the case depicted in Figure 2 (sinc mapping) the optimum sample position is modified progressively as the strength of the prior knowledge is increased ( $\gamma$  is decreased). In Figure 6 we have seen that the sampling scheme which is selected by maximising the transinformation corresponds to the exponential sampling suggested in reference [14]. This follows because Figure 6 shows that the logarithms of the optimum sample positions are approximately uniformly spaced. The data dimensionality results which we have shown in Figure 7 (analogous to Figure 4 for the sinc mapping) are consistent with SVD results.

We should not expect there to be gross differences between our results and those obtained by using the SVD, because the model which we have used to generate our numerical results (Equation 3.12) is closely related to the SVD method as explained in Section III and in references [11, 12]. However, use of the transinformation measure is not restricted to such simple situations. Indeed any data which are not distributed in a (correlated) gaussian manner cannot be analysed for their transinformation content by a simple diagonalisation. For more complicated distributions where correlations between more than two points (samples) must be separately specified we must use the basic equations of Section I to measure the transinformation. An estimate of the dimensionality of the data which is obtained from the SVD method will be an overestimate if there are important higher-order correlations present. This is because any multi-point correlations (in addition to those implied by a correlated gaussian distribution) will decrease the transinformation content of

the data. In general we should perform a full stochastic (Monte-Carlo) simulation of the data measurement process and the system under observation in order to derive the transinformation in complicated cases. We may then determine its dependence on sample position, sample type (quantisation levels, etc) and any other relevant parameters.

## Appendix

In this appendix we shall summarise some formulae which are too long to include in the main text. We shall assume that the transinformation has been expressed in the form

$$I \equiv \log_2(\det[M]) \quad (1)$$

where

$$M_{a,b} \equiv R_{a,b} + \delta_{a,b} \quad (2)$$

This form corresponds to that used in the text if we identify

$$R_{a,b} \equiv R r_{a,b} \quad (3)$$

where  $R$  is a suitably defined signal-to-noise ratio. Thus  $R_{a,b}$  is the covariance matrix of the signal component of the data with the convention that the data are scaled so that the additive white noise has unit variance. We may evaluate the determinant in Equation 1 for various dimensions ( $m$ ) of the data space. Thus we have

---


$$\begin{aligned} m = 1 : \det[M] &= (R_{1,1} + 1) \\ m = 2 : \det[M] &= (R_{1,1} + 1)(R_{2,2} + 1) - |R_{1,2}|^2 \\ m = 3 : \det[M] &= (R_{1,1} + 1)(R_{2,2} + 1)(R_{3,3} + 1) \\ &\quad - (R_{1,1} + 1)|R_{2,3}|^2 - 2 \text{ perms} \\ &\quad + 2 \operatorname{Re}(R_{1,2} R_{2,3} R_{3,1}) \\ m = 4 : \det[M] &= (R_{1,1} + 1)(R_{2,2} + 1)(R_{3,3} + 1)(R_{4,4} + 1) \\ &\quad - (R_{1,1} + 1)(R_{2,2} + 1)|R_{3,4}|^2 - 5 \text{ perms} \\ &\quad + |R_{1,2} R_{3,4}|^2 + 2 \text{ perms} \\ &\quad + 2(R_{1,1} + 1)\operatorname{Re}(R_{2,3} R_{3,4} R_{4,2}) + 3 \text{ perms} \\ &\quad - 2\operatorname{Re}(R_{1,2} R_{2,3} R_{3,4} R_{4,1}) - 2 \text{ perms} \end{aligned} \quad (4)$$


---

Note that it is important to retain the cyclic order of the variables when permuting terms which are not modulus squared.

On the other hand, we may define a gaussian PDF which has covariance  $R_{a,b}$

$$P[z] \equiv \frac{1}{\det[\pi R]} \exp(-z^\dagger R^{-1} z) \quad (5)$$

where  $z$  is an  $m$ -dimensional complex vector and  $R$  is a

matrix with elements  $R_{a,b}$ . Using the factorisation properties of gaussian PDFs we may derive the expectation values of products of the intensities (not to be confused with transinformation) where

$$I_k \equiv |z_k|^2 \quad (6)$$

Thus we obtain

$$\begin{aligned}
\langle I_1 \rangle &\equiv \langle 1 \rangle = R_{1,1} \\
\langle I_1 I_2 \rangle &\equiv \langle 1 \cdot 2 \rangle = R_{1,1} R_{2,2} + |R_{1,2}|^2 \\
\langle I_1 I_2 I_3 \rangle &\equiv \langle 1 \cdot 2 \cdot 3 \rangle = R_{1,1} R_{2,2} R_{3,3} \\
&\quad + R_{1,1} |R_{2,3}|^2 + 2 \text{ perms} \\
&\quad + 2 \operatorname{Re}(R_{1,2} R_{2,3} R_{3,1}) \\
\langle I_1 I_2 I_3 I_4 \rangle &\equiv \langle 1 \cdot 2 \cdot 3 \cdot 4 \rangle = R_{1,1} R_{2,2} R_{3,3} R_{4,4} \\
&\quad + R_{1,1} R_{2,2} |R_{3,4}|^2 + 5 \text{ perms} \\
&\quad + |R_{1,2} R_{3,4}|^2 + 2 \text{ perms} \\
&\quad + 2 R_{1,1} \operatorname{Re}(R_{2,3} R_{3,4} R_{4,2}) + 3 \text{ perms} \\
&\quad - 2 \operatorname{Re}(R_{1,2} R_{2,3} R_{3,4} R_{4,1}) - 2 \text{ perms}
\end{aligned} \tag{7}$$

The quantities which appear in Equation 4 may be expressed in terms of the expected values of the intensity products to give

$$\begin{aligned}
m = 1 : \det[M] &= (\langle 1 \rangle + 1) \\
m = 2 : \det[M] &= ((\langle 1 \rangle + 1)(\langle 2 \rangle + 1) - (\langle 1 \cdot 2 \rangle - \langle 1 \rangle \langle 2 \rangle)) \\
m = 3 : \det[M] &= ((\langle 1 \rangle + 1)(\langle 2 \rangle + 1)(\langle 3 \rangle + 1) \\
&\quad - (2 \langle 1 \rangle + 1)(\langle 2 \cdot 3 \rangle - \langle 2 \rangle \langle 3 \rangle) - 2 \text{ perms} \\
&\quad + (\langle 1 \cdot 2 \cdot 3 \rangle - \langle 1 \rangle \langle 2 \rangle \langle 3 \rangle)) \\
m = 4 : \det[M] &= ((\langle 1 \rangle + 1)(\langle 2 \rangle + 1)(\langle 3 \rangle + 1)(\langle 4 \rangle + 1) \\
&\quad - (6 \langle 1 \rangle \langle 2 \rangle + 2 \langle 1 \rangle + 2 \langle 2 \rangle + 1)(\langle 3 \cdot 4 \rangle - \langle 3 \rangle \langle 4 \rangle) - 5 \text{ perms} \\
&\quad + 2 (\langle 1 \cdot 2 \rangle \langle 3 \cdot 4 \rangle - \langle 1 \rangle \langle 2 \rangle \langle 3 \rangle \langle 4 \rangle) + 2 \text{ perms} \\
&\quad + (2 \langle 1 \rangle + 1)(\langle 2 \cdot 3 \cdot 4 \rangle - \langle 2 \rangle \langle 3 \rangle \langle 4 \rangle) + 3 \text{ perms} \\
&\quad - (\langle 1 \cdot 2 \cdot 3 \cdot 4 \rangle - \langle 1 \rangle \langle 2 \rangle \langle 3 \rangle \langle 4 \rangle))
\end{aligned} \tag{8}$$

These formulae allow us to express the transinformation either in terms of covariance matrix elements (as in Equation 4) or in terms of intensity moments (as in Equation 8).

- [1] M Bertero, P Bocacci, and E R Pike, *On the recovery and resolution of exponential relaxation rates from experimental data II. The optimum choice of experimental sampling points for Laplace transform inversion*, Proceedings of the Royal Society A **393** (1984), no. 1804, 51–65.
- [2] M Bertero, P Bocacci, and E R Pike, *On the recovery and resolution of exponential relaxation rates from experimental data: a singular value analysis of the Laplace transform inversion in the presence of noise*, Proceedings of the Royal Society A **383** (1982), no. 1784, 15–29.
- [3] ———, *Resolution in diffraction-limited imaging, a singular value analysis II. The case of incoherent illumination*, Optica Acta **29** (1982), no. 12, 1599–1611.
- [4] M Bertero, P Brianzi, P Parker, and E R Pike, *Resolution in diffraction-limited imaging, a singular value analysis III. The effect of sampling and truncation of the data*, Optica Acta **31** (1984), no. 2, 181–201.
- [5] M Bertero, P Brianzi, and E R Pike, *On the recovery and resolution of exponential relaxation rates from experimental data III. The effect of sampling and truncation of data on the Laplace transform inversion*, Proceedings of the Royal Society A **398** (1985), no. 1814, 23–44.
- [6] M Bertero, C de Mol, E R Pike, and J G Walker, *Resolution in diffraction-limited imaging, a singular value analysis IV. The case of uncertain localisation or non-uniform illumination of the object*, Optica Acta **31** (1984), no. 8, 923–946.
- [7] M Bertero and E R Pike, *Resolution in diffraction limited imaging, a singular value analysis I. The case of coherent illumination*, Optica Acta **29** (1982), no. 6, 727–746.
- [8] W M Brown and C J Palermo, *Random processes, communications and radar*, McGraw-Hill, New York, 1969.
- [9] B R Frieden, *Probability, statistical optics, and data testing*, Springer Verlag, Berlin, Heidelberg, New York, 1983.
- [10] E T Jaynes, *Prior probabilities*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 3, 227–241.
- [11] S P Luttrell, *A new method of sample optimisation*, Optica Acta **32** (1985), no. 3, 255–257.
- [12] ———, *Prior knowledge and object reconstruction using the best linear estimate technique*, Optica Acta **32** (1985), no. 6, 703–716.
- [13] J G McWhirter and E R Pike, *On the numerical inversion of the Laplace transform and similar Fredholm integral equations of the first kind*, Journal of Physics A: Mathematical and General **11** (1978), no. 9, 1729–1746.
- [14] N Ostrowsky, D Sornette, P Parker, and E R Pike, *Exponential sampling method for light scattering polydispersity analysis*, Optica Acta **28** (1981), no. 8, 1059–1070.
- [15] C E Shannon, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 3, 379–423.

- [16] \_\_\_\_\_, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 6, 623–656.
- [17] D Slepian and H O Pollack, *Prolate spheroidal wave func-*  
*tions Fourier analysis and uncertainty I*, Bell System Technical Journal **40** (1961), no. 1, 43–63.



# Prior Knowledge in Synthetic-Aperture Radar Processing \*

S P Luttrell and C J Oliver  
Royal Signals and Radar Establishment, Great Malvern, Worcs WR14 3PS, UK

We briefly review the role of models as a means of encoding prior knowledge with which to interpret data. We then examine the specific case of synthetic-aperture radar (SAR) images. We review the current state of SAR terrain clutter models, and their role in target detection. We present numerical results which demonstrate the consistency of a correlated gamma-distributed surface cross section model with SAR terrain data. We then review the theory of target super-resolution by the use of the singular-value decomposition (SVD). We emphasise the need to generalise the basic SVD technique in order to achieve success with SAR target data. Furthermore we demonstrate that the general SVD technique is a special case of a Bayesian reconstruction scheme which we interpret in terms of Shannon information theory. Numerical super-resolution results from simulated SAR data are presented.

## I. INTRODUCTION

### A. Introduction

This introductory section has been written to convey our current philosophy with regard to data interpretation. At the most austere extreme we can interpret a set of data measurements as just a set of numbers and nothing more. Whilst this is certainly true of the data values themselves, we must not ignore the context within which the data were acquired. We must therefore endow the bare data measurements with ‘meaning’. We do not intend to delve into the philosophical minefield that surrounds studies in this area; instead we shall limit ourselves to the observation that we use models to express the context within which data measurements are made. A physical model embodies those aspects of our past experience that we believe are related to the data measurements, and thus forms a microcosm within which the data have a particular meaning. We shall use the phrase ‘prior knowledge’ to describe the microcosm (or physical model) that we shall use for data interpretation. In this review we shall return to this same basic theme repeatedly.

### B. Synthetic-aperture radar image analysis

Let us next discuss how the concepts of prior knowledge may be applied to the interpretation of synthetic-aperture radar (SAR) imagery. A SAR is a coherent radar which synthesises a large effective aperture by suitable coherent recombination of the received signal from many radar pulses. This large synthetic aperture gives

a radar image with very high resolution when compared with conventional radar. SAR images typically contain an enormous quantity of data, e.g. 500000 samples per second, of which the principal component is map-like terrain imagery, which we shall call clutter, within which may be set a few significant objects, termed targets. In some remote sensing applications it is the terrain imagery that is of interest; in others it is only the targets. In the former instance the detail of the clutter has to be studied closely; in the latter the clutter provides the background and context to the objects of significance.

The previous paragraph contains a description of a SAR image which assumes a wealth of information that is not to be found in the SAR image alone. For instance the partitioning of an image into clutter and targets does not have an intrinsic meaning; it is a meaningful operation to perform only in the context of a wider comprehension of the type of world that we inhabit. Thus we have introduced the prior knowledge that (typically) targets are small bright blobs that are embedded in a weaker clutter background. This expression of our prior knowledge could be read as a definition of what we mean by ‘target’ and ‘clutter’.

We are very fortunate that many targets may be partitioned out of the image by a thresholding operation alone. However in order to define a suitable threshold level we must develop a sophisticated understanding of the properties of targets and clutter. We start with the (incorrect) assumption that the whole image is clutter, and that the task is to detect anomalies which we call targets. The primary physical model that we must construct is therefore a clutter model. In Section II we shall review some work that has had success in the modelling of SAR terrain clutter. Using this clutter model we shall define a target detection threshold. Detected targets may then be subjected to a closer examination by introducing a detailed ‘target microcosm’ within which the target data may be interpreted. In Section III we shall review a number of related areas of study. These may be generically classified as super-resolution theory.

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 11, 2019.

This paper appeared in *J. Phys. D: Appl. Phys.*, 1986, vol. 19, no. 3, pp. 333-356. Received 23 July 1985, in final form 8 September 1985. © 1986 The Institute of Physics.

S P Luttrell’s contribution to this paper was the work on super-resolution.

## II. CLUTTER MODELS AS PRIOR KNOWLEDGE

In Figure 1 we show a section of SAR image obtained with the RSRE X-band system. Let us study the nature of this image and the types of prior knowledge about the general terrain that are relevant. We distinguish three relevant classes of prior knowledge which we represent in terms of models: (i) the scattering model, (ii) the texture model and (iii) the structure model. Examples of data interpretation where we require each of these three aspects of prior knowledge can be seen in Figure 1.

(i) A striking feature of Figure 1 is the observed speckle which appears to affect the entire image. This can be represented in terms of a scattering model in which we make no attempt to describe the scatterer configuration within the resolution cell, merely postulating that each cell contains many randomly distributed scatterers giving rise to some total cross section per resolution cell from which the observed image is derived. When incident coherent radiation is scattered from such a surface the random interference between the individual scatterers gives rise to speckle. The variation in image intensity from pixel to pixel is the result, as we shall show in Section II A of the convolution of the underlying cross section per resolution cell with a negative exponential speckle contribution. Where there is no fluctuation in the cross section the image corresponds to classical Rayleigh speckle (e.g. [29], p 478).

(ii) In order to progress beyond the scattering model introduced above we need to introduce knowledge about the terrain structures themselves. The texture model is intended to contain prior knowledge about the local fluctuations in cross section between pixels within an extended terrain feature, such as a built-up area, woodland or open field. We shall introduce a form of this model that assumes that such local texture may be described in terms of a generalised noise process with characteristic depths of fluctuation and correlation properties. We further assume in our clutter model that the noise is stationary and ergodic.

(iii) The boundaries of this local texture are provided by the macroscopic structural model which is essentially derived from map data. The precise division between this and the microscopic texture is not clear-cut. One might describe a wood in various ways, for example. At one level only the boundaries of the wood are described and a random texture of appropriate properties used to represent the internal structure. This would generate typical wood-like images with the correct statistical properties. Furthermore if we needed to generate a particular wood, then we would have to use collateral data from maps about tree position and use a random statistical method

to represent the scattering from the individual tree. For ultimate detail one would be forced to represent the position of every branch and leaf on every tree (which is obviously a massive computing task). We choose to select the boundaries of regions of similar texture on the basis of the macroscopic structural model and treat the region as representable in terms of random processes whose properties are summarised in the texture model. This approach may be used to describe natural features such as woods or fields, but it is unlikely to represent man-made features such as towns very convincingly.

In Section II A we consider the theoretical foundation for the above three models. The measured correlation properties for natural textures (fields and woods) in experimental SAR data are analysed and the implications for the clutter statistics in systems of arbitrary resolution indicated in Section II B. In Section II C we shall briefly review the various techniques that could be used to set a target detection threshold.

### A. Theoretical foundation for prior knowledge models

The basis of the physical scattering model is that the surface is represented by a large number of randomly distributed scatterers per resolution cell. The total-field contribution at a position  $y$  is then found by summation over the  $N$  illuminated scatterers, so

$$\varepsilon(y) = B \sum_{i=1}^N a_j e^{i\phi_j} T(y, x_j) \quad (2.1)$$

where  $a_j$  and  $\phi_j$  are the amplitude and (random) phase of the field from the  $j$ th scatterer,  $T(y, x_j)$  is the imaging response function (i.e. point spread function) and  $B$  is a term related to the conventional radar equation which is discussed in detail by [21].  $B$  is defined by the equation

$$B = \left[ \frac{(P_T Z_0 G_0 A_r)^{\frac{1}{2}}}{4 \pi R_0^2} \right] F_T$$

where  $P_T$  is the transmitter power,  $Z_0$  the characteristic impedance of the propagation medium,  $G_0$  the antenna transmitter aperture gain,  $A_r$  the receiving antenna area,  $R_0$  the range and  $F_T$  the impulse response of the range pulse compression subsystem of the SAR. Since the parameter  $B$  only affects the overall scale of the received power, its value does not affect our image interpretation. Therefore we shall discard  $B$  in the following analysis.

The received power (image intensity) in the scattered field is given by the square of the envelope of the field, so



Figure 1: A section of SAR imagery ( $6.75 \times 3.75\text{km}^2$ ) obtained with the RSRE X-band SAR system.

$$I(y) \equiv |\varepsilon(y)|^2 = |B|^2 \sum_{j,k=1}^N a_j a_k^* \exp[i(\phi_j - \phi_k)] T(y, x_j) T^*(y, x_k) \quad (2.2)$$


---

If we assume that there is no underlying variation in the cross section then Equation 2.1 and Equation 2.2 reduce to the classical Rayleigh random walk process yielding a complex gaussian field. The detected intensity (the square modulus of the field) then has a negative exponential probability density function (PDF), while that for the envelope (square-root intensity) is Rayleigh distributed. Thus, classical speckle (see, e.g., [29], p 478), will be observed when there is no underlying cross section fluctuation.

Let us next examine how the fluctuations in the scattering amplitudes of the individual scatterers affect the observed intensity. Since these scatterers are assumed to be randomly positioned and if we assume that the underlying cross section fluctuations between pixels are determined by a stationary ergodic process then the properties of the intensity will be conveniently described in terms of ensemble-averaged quantities taken over all possible realisations of the detected intensity. For the average intensity we observe that the random phases of scatterers  $j$  and  $k$  in Equation 2.2 cause all terms except those where  $j = k$  to average to zero. Thus

$$\langle I \rangle = |B|^2 \sum_{j=1}^N \langle \sigma_j \rangle |T(y, x_j)|^2 \quad (2.3)$$

where

$$\sigma_j \equiv a_j a_j^*$$

is defined as the cross section of the  $j$ th scatterer. Note that we implicitly assume that the point spread function is translation-invariant. The mean intensity thus depends only on the summation of the individual cross section contributions from each of the randomly positioned scatterers within the resolution cell rather than on any interference between them. It can be shown [22–24] that all the higher correlation moments of the intensity can similarly be expressed solely in terms of the properties of these cross section contributions from the

individual scatterers. In this scattering model we ignore the detail of the cross section fluctuations of individual scatterers and assume that all those within a resolution cell have the same cross section. For the rest of this paper, therefore, we shall consider only how this averaged cross section varies with position. Similarly, due to the absence of interference terms already noted, we observe an intensity equivalent to what would result from scattering the incident field from a single scatterer with the appropriate value of cross section for that position. Only this observed intensity will be considered for the rest of these sections on clutter; the intensity contribution from individual scatterers may be ignored. The basic assumption of this scattering model that each cell contains many randomly positioned scatterers is sufficient to ensure that the contributions all combine incoherently in this manner. Of course, the model will be violated wherever a strong single scatterer (e.g. a target) is present.

The simplest statistical properties of the observed intensity that one might measure in order to characterise the texture would be the single-point higher-order moments and two-point autocorrelation function. Obviously these properties can only represent a minimum characterisation. When applied to woodland, for example, the autocorrelation function might be expected to be primarily related to the underlying spatial scale determined by quantities such as tree size and separation. The single-point statistics, on the other hand, describes more the depth of fluctuation of the cross section, which might be governed by the extent of shadowing between trees and their variability of cross section.

The normalised intensity autocorrelation function is defined by

$$g^{(2)}(r) \equiv \frac{\langle I(0) I(r) \rangle}{\langle I \rangle^2} \quad (2.4)$$

Taking into account only dominant terms once the effect of the random phases is included this has been shown to be given by [23]

$$g^{(2)}(r) = \frac{\sum_{j,k=1}^N \langle \sigma_j \sigma_k \rangle \left( |T(y, x_j)|^2 |T(y+r, x_k)|^2 + T(y, x_j) T^*(y+r, x_j) T^*(y, x_k) T(y+r, x_k) \right)}{\langle \sigma \rangle^2 \sum_{j,k=1}^N |T(y, x_j)|^2 |T(y, x_k)|^2} \quad (2.5)$$


---

As already noted this result depends only on the cor-

relation properties of the local cross section and on the

imaging response function. Equation 2.5 may be simplified analytically for various forms of cross section and imaging response [22–24].

On analysing the coherent scattering from the random scatterers with a local cross section  $\sigma$  then the dominant term of the incoherent summation yields an  $n$ th-order normalised intensity moment consisting of  $n!$  terms containing the  $n$ th-order normalised surface cross section

moment,  $S^{(n)}$  [23]. Thus

$$\begin{aligned} I^{(n)} &= \frac{\langle I^n \rangle}{\langle I \rangle^n} \\ &= n! S^{(n)} \end{aligned} \quad (2.6)$$

where

$$\begin{aligned} S^{(n)} &\equiv \frac{\langle S^n \rangle}{\langle S \rangle^n} \\ &= \frac{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} dx_1 \cdots dx_n \langle \sigma_1 \cdots \sigma_n \rangle |T(y, x_1)|^2 \cdots |T(y, x_n)|^2}{\left( \langle \sigma \rangle \int_{-\infty}^{\infty} dx |T(y, x)|^2 \right)^n} \end{aligned} \quad (2.7)$$

Again, the intensity depends only on the properties of the local cross section. The role of the speckle process, as shown in Equation 2.6, is to determine the number of terms of the form of Equation 2.7 that contribute to the observed intensity.

Having discussed the relationship between the textural properties of the surface and those of the observed intensity as characterised by their one- and two-point statistical properties, we now wish to construct a specific analytic surface texture model. One such model that has been proposed [10, 13–15] is that the surface cross section should be represented by a random process having a gamma-distributed PDF given by

$$p(\sigma) = \frac{1}{\langle \sigma \rangle} \left( \frac{\sigma}{\langle \sigma \rangle} \right)^{\nu-1} \left( \frac{1}{\nu \Gamma(\nu)} \right) \exp \left( -\frac{\sigma}{\langle \sigma \rangle} \right) \quad (2.8)$$

On scattering, the observed intensity is the convolution of the gamma distribution with a negative exponential, termed a  $K$ -distribution, having a PDF given by

$$p(I) = \frac{1}{\langle I \rangle} \left( \frac{I}{\langle I \rangle} \right)^{\frac{\nu-1}{2}} \frac{1}{\Gamma(\nu)} K_{\nu-1} \left[ 2 \left( \frac{I}{\langle I \rangle} \right)^{\frac{1}{2}} \right] \quad (2.9)$$

where  $\Gamma(\nu)$  is the gamma function of order  $\nu$  and  $K_{\nu-1}$  is the  $K$ -distribution (modified Bessel function) of order  $\nu - 1$  [3], p 374). The cross section and intensity PDFs are characterised by their mean values,  $\langle \sigma \rangle$  and  $\langle I \rangle$ , and an order parameter,  $\nu$ . If the surface were represented as classical thermal noise, rather than in this generalised form, it would have a negative exponential PDF characterised only by the mean. Thus the effect of the order parameter in Equation 2.8 and Equation 2.9 is to introduce another degree of freedom into the noise model. Indeed  $\nu$  controls the ‘spikiness’ of the PDF. For  $\nu$  infinite,  $p(\sigma)$  is a  $\delta$ -function; for  $\nu = 1$  it corresponds to thermal noise; while as  $\nu$  becomes small the PDF becomes very spiky. We may regard the gamma distribution, therefore, as a

generalisation of classical thermal noise. Such behaviour would result from a physical model in which the density of scatterers (or the local cross section) depended on a birth-death-migration process [10]. Conventional thermal noise, on the other hand, may be regarded as the result of a physical model containing migration (diffusion) alone. Thus the step of including a local source and sink of scatterers is represented in the order parameter. For a  $K$ -distribution the normalised intensity moments are given by [10]:

$$I^{(n)} = \frac{n! \Gamma(n + \nu)}{\nu^n \Gamma(\nu)} \quad (2.10)$$

A variety of scattering experiments at both microwave and optical wavelengths [9, 12, 13, 26, 30?, 31] have shown that such a distribution is consistent with the data. Ward [30, 31] demonstrated that the effects of speckle and the underlying cross section fluctuations could be factorised for sea clutter data where fluctuations took place on a scale much greater than the resolution cell.

In addition to considering the single-point statistics of the surface cross section it is obviously essential to include spatial correlation effects. Indeed these may well prove to be of much greater significance than the detail of the single-point statistics. Various simple theoretical models of this kind, based on two-point correlations only, have been proposed [22–24] and the interaction of the surface cross section fluctuations with the imaging response function calculated.

After introducing a suitable texture model, such as a correlated gamma distribution, we must next examine how the higher-level structural model is used to interpret the final image. As already outlined, the structural model delineates the boundaries of different features, each of which will have some texture defined in terms of its statistical and correlation properties. Further prior knowledge in the form of the direction, range and altitude of

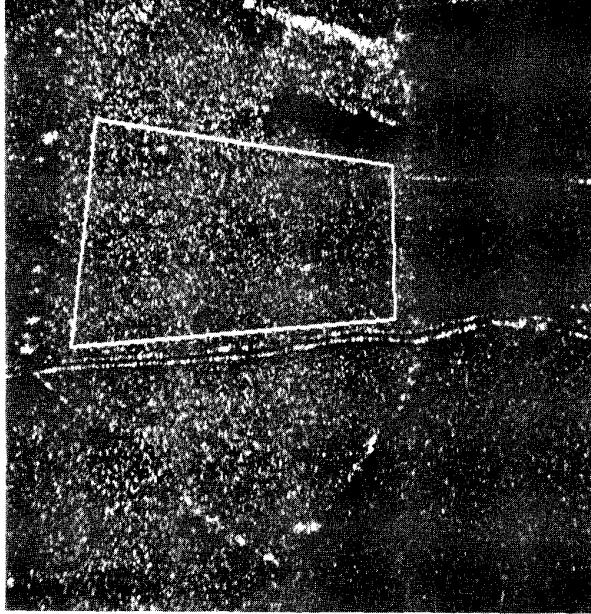


Figure 2: The sample image taken from Figure 1 containing a region identified as ‘field’ denoted by the full line.

the SAR sensor, together with parameters such as feature height, can then be included to predict brighter areas, such as the front edge of a wood, or shaded regions, behind the wood for example. This approach, together with texture simulation methods [25] has been used to simulate SAR images [9]. For a more complete understanding the interaction of the imaging response function with the correlation properties of the surface must be included [22–24]. In the extreme case of a low-resolution radar the resolution cell will include contributions from many different features each of which has its own order parameter and correlation length. The scattering model adopted then implies that, by following the same line of reasoning, the total observed intensity may be regarded as the incoherent sum of the contributions from each feature. Thus each feature may be treated as if imaged independently and the resultant intensity is then summed directly. At present this model has only been applied in the direct imaging sense; the application of the model to the inverse problem has yet to be attempted.

### B. Experimental verification of clutter model

As already noted the  $K$ -distributed clutter model has been tested against a variety of scattering situations showing close agreement [9, 11–13, 22–24, 26, 30, 31]. Of particular relevance to the present discussion are the radar measurements on sea clutter [30, 31], which demonstrate that the clutter statistics could indeed be represented by a convolution of an underlying gamma-

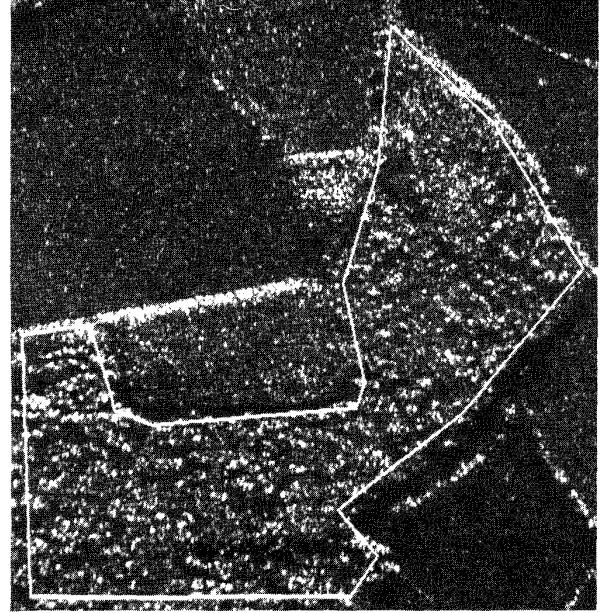


Figure 3: A sample image containing a region identified as ‘wood’ denoted by the full line.

distributed variable with gaussian speckle, and the land clutter measurements based on SAR [23, 24]. The latter examined only the overall properties of a large region containing a mixture of features such as fields, hedges and small woods rather than the detail of the individual components. It also measured the spatial correlation properties of land clutter and made the first test of the effects of varying resolution on the resultant clutter properties, demonstrating close agreement between theory and experiment. More recently the properties of the individual textures have been analysed in a similar manner [9]. Some of these results will be summarised in the present paper.

Figure 2 and Figure 3 show sections of SAR imagery containing areas which have been identified as ‘field’ and ‘wood’ respectively. Let us establish the extent to which the clutter within the marked regions can be represented in terms of the texture model. Table I contains the first six normalised moments of the intensity of the field region. If there were no underlying cross section fluctuation then the order parameter,  $\nu$ , of the model would be infinite and the  $n$ th moment would take the value  $n!$ . In fact an order parameter value of 33.3 gives a closer fit to the data which might correspond to some slight underlying fluctuation. Close study of the image suggests that there is indeed a slight gradient across the image. Table II shows the equivalent result for the wooded area in Figure 3. Clearly there is some underlying structure and an order parameter of 2.63 yields reasonable agreement between theory and experiment. Thus the single-point statistics for these two natural textures are consistent

Table I: Comparison of the measured normalised intensity moments for the field region with the predictions of the  $K$ -distributed clutter model for  $\nu = 33.3$ . We used 42103 pixels.

Moment	Measured	Expected
First	1	-
Second	2.06	-
Third	6.53	6.55
Fourth	$2.82 \times 10$	$2.86 \times 10$
Fifth	$1.54 \times 10^2$	$1.60 \times 10^2$
Sixth	$1.01 \times 10^3$	$1.10 \times 10^3$

Table II: Comparison of the measured normalised intensity moments for the wooded region with the predictions of the  $K$ -distributed clutter model for  $\nu = 2.63$ . We used 82253 pixels.

Moment	Measured	Expected
First	1	-
Second	2.76	-
Third	$1.43 \times 10$	$1.46 \times 10$
Fourth	$1.11 \times 10^2$	$1.25 \times 10^2$
Fifth	$1.14 \times 10^3$	$1.57 \times 10^3$
Sixth	$1.43 \times 10^5$	$2.74 \times 10^5$

with the model.

The azimuthal intensity autocorrelation functions for the two regions are illustrated in Figure 4 and Figure 5 respectively. The linear plot of the autocorrelation function (ACF) for the field (Figure 4) shows no evidence for any correlation structure. The initial peak at zero lag results from the coherent interference between contributions within the imaging response and is entirely consistent with the theory [23]. The azimuthal ACF for the wood is plotted on a logarithmic scale which indicates that the result is well fitted by a single negative exponential decay corresponding to an underlying cross section fluctuation with a lorentzian spectrum. This form of spatial correlation has been analysed previously with different imaging response functions [23, 24]. The ACF measurements are therefore also consistent with the texture model. The equivalent results for range show the same underlying intensity variation. However, the range resolution is much poorer than the azimuth resolution, so much of the structure is disguised.

The field texture is uncorrelated from pixel to pixel with essentially a negative exponential intensity PDF. The wood texture can be described as a gamma-lorentzian cross section fluctuation. The previous analyses [23, 24] of the effect of the imaging response function on the final-clutter properties may therefore be applied directly. Since the scattering from the field region is essentially represented by pure gaussian speckle, the clutter statistics are unchanged on summing many independent contributions. However, for the wooded region

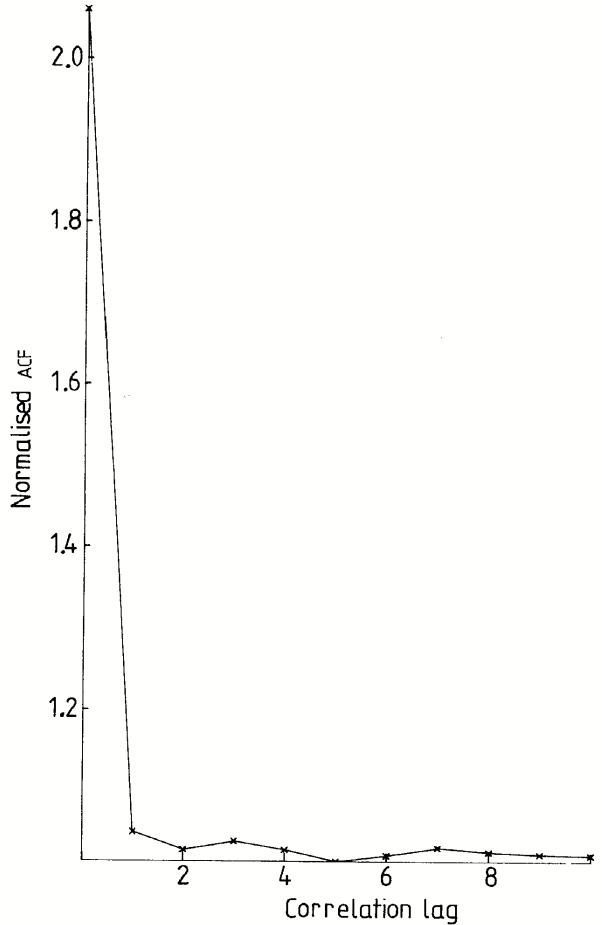


Figure 4: A linear plot of the intensity ACF in the azimuthal direction for the field region in Figure 2.

the underlying cross section fluctuations, describable as gamma-distributed noise of order parameter 2.63, introduce correlations between pixels that affect the dependence of the clutter statistics on system resolution. As an example we illustrate in Figure 6 the predicted dependence of the normalised intensity moments as the resolution in the azimuthal direction is degraded; we assume a rectangular response function for simplicity. The full curves denote theoretical predictions [23], while the broken curves show the equivalent results for uncorrelated gamma-distributed noise of the same order. In the latter instance the intensity PDF is always a sum over independent gamma distributions so the resultant intensity is  $K$ -distributed with an order parameter proportional to the resolution length. No such simple relation exists for the correlated surface. The values of the moments are chosen to be identical in the limit of very high resolution. The values are also identical in the opposite limit of very low resolution where both predict gaussian speckle. In the intermediate region, however, the correlation in the surface cross section determines the intermediate shape of the curves. It is important to note that integrated

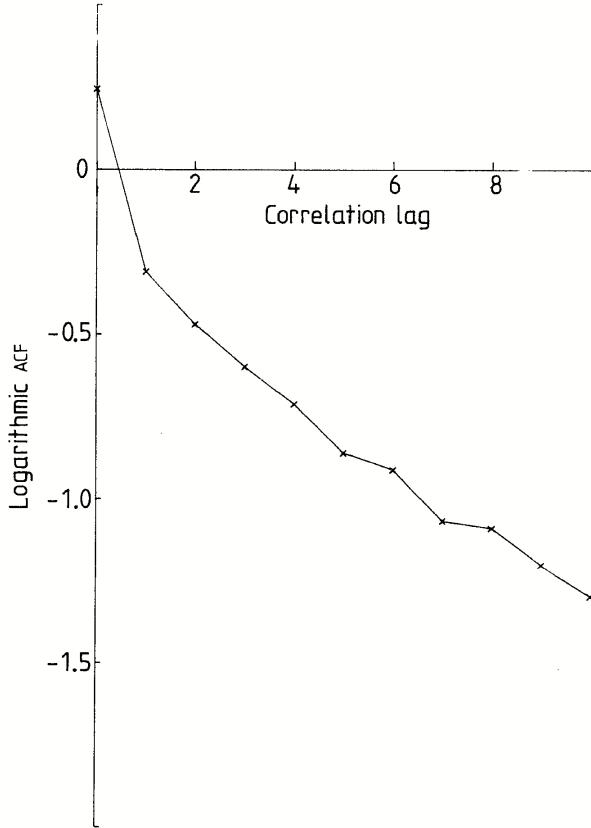


Figure 5: A logarithmic plot of the intensity ACF in the azimuthal direction for the wood region in Figure 3.

clutter statistics predicted on the assumption of an uncorrelated surface of the appropriate order tend to the gaussian speckle limit much faster than would in fact be the case for a correlated surface. This highlights the importance of including the correlation properties of the surface in the clutter model.

### C. Target detection

Target detection occurs when a localised area (a few resolution cells, say) of the SAR image is not considered to be a feasible piece of clutter; usually this means that the local area is too bright. The simplest type of technique for achieving this detection is that of [16], where a local average of the mean and variance of the intensity is used to define a local target detection threshold. A more sophisticated technique is to use a detailed ‘clutter map’ which defines the statistical properties that are expected at each point in the SAR image. Such a clutter map would have been created from map data and/or previous SAR flights past the same region. The clutter map is then used to define a ‘target detection threshold map’. A hybrid of these two techniques uses a local av-

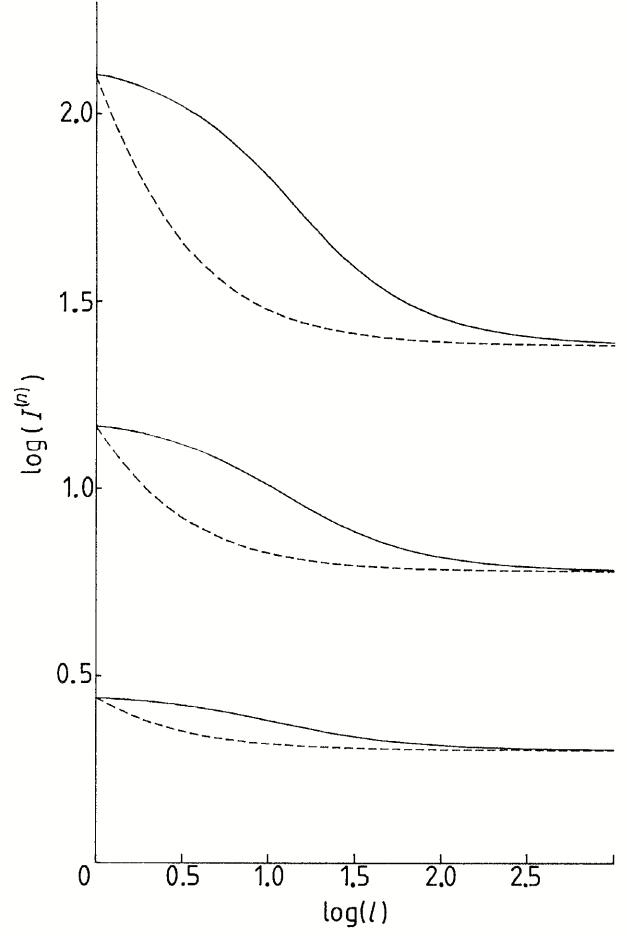


Figure 6: A comparison of the dependence of the higher-order single-point normalised intensity moments,  $I^{(n)}$ , as a function of system azimuthal resolution. The full curves denote the results for the fluctuations having the statistical and correlation properties of the wood region in Figure 3; the broken curves illustrate results for a surface having the same statistics but no correlation.

eraging process to determine which type of clutter (e.g. wood, field) is present at each point in the SAR image, and then uses an external map of typical clutter types to then define a local target detection threshold.

### III. TARGET SUPER-RESOLUTION FROM PRIOR KNOWLEDGE

The process of target classification can begin once the target has been detected. Classification is necessarily a more exacting task than detection, because the detailed response to the target must be examined for clues about its nature. We shall develop a scheme whereby the blurred image of a target may be transformed into a more convenient representation on which target classification may operate with greater ease. This is achieved by

introducing a model that is more specific about targets than the clutter model that was used for target detection. We do not wish to be overspecific about the nature of targets at this early stage, so we wish our model to express only general properties of targets at first. Clearly target detection already uses the general property that targets are usually strong scatterers. The next general property that we can introduce is that targets are usually restricted in spatial extent. We must express this in the form of a model, and process the detected targets accordingly. The outcome of this is an enhancement in the resolution of the target, which justifies the use of the term ‘super-resolution’ to describe this form of target image processing.

In the following subsections we shall review the theory and practice of target superresolution in SAR images. We shall assume at the outset that a suitable clutter model has been used to define a target detection threshold, and that any detected targets are then passed on to be superresolved.

### A. Target models as prior knowledge

We shall introduce the super-resolution technique that we use by tracing its origins in simpler techniques. Thus we must begin by recalling the ‘Rayleigh resolution criterion’. This states that two point sources are only just resolved when the first null of one point spread function (PSF) lies on the main lobe of the other PSE. This is a very useful operational criterion for determining the ultimate resolution of an imaging device.

The Rayleigh resolution criterion (RRC) appears to be unnecessarily restrictive, because if indeed there were only two point sources present then we could inspect the detailed shape of the image to determine their amplitudes, phases and positions. The limitation on what could be achieved would be determined by the extent to which the data were corrupted by noise. This approach is the essence of the ‘monopulse’ technique ([29], p 160), which *assumes* that there are only two point sources present. Resolutions that are far better than that predicted by the RRC may be obtained with ease. This apparent violation of the RRC has occurred because we have excluded the possibility that the data derive from anything other than two point sources. Although the RRC refers to two point sources, the resolution criterion that is obtained is of course far too weak for such a case (as monopulse demonstrates convincingly). To be more precise the RRC refers to a *global* resolving capability; it specifies the separation that each neighbour of a whole array of point sources must have in order that they may be simultaneously resolvable. Henceforth we shall assume that the conventional resolving capability of an imaging device is given by the RRC. We shall call the enhanced resolution that may be obtained under more specific circumstances (e.g. monopulse) super-resolution. We may now identify what type of circumstance gives rise to a

super-resolving capability. The example afforded by the monopulse suggests that any situation in which we have prior knowledge about the object being imaged may give rise to super-resolution.

Much effort has been expended in formulating ways of incorporating more general types of prior knowledge into data interpretation schemes. Of course we could generalise the monopulse technique in a trivial fashion by fitting the data with a larger number of point sources. Provided that the object does indeed consist of point sources and that the noise level in the data is not too great then the amplitudes, phases and positions of the point sources may be recovered. This is achieved by fitting the detailed shape of the data (which consist of a set of overlapping PSF) to within an error given by the noise level. If the noise level is too great then the fit is ambiguous, and a great many alternative solutions may exist. A more specific procedure for selecting a solution must then be formulated in order to obtain a unique fit. Typically one might accept as the solution the smallest set of point sources that gives a reasonable fit to the data. This scheme has the advantage of not incorporating any more point sources than is absolutely necessary.

The monopulse generalisation has the disadvantage that its prior knowledge is specified in terms of point sources only. We would prefer a data interpretation scheme that permitted a larger class of objects to be considered. This is because we cannot ever be sure that the ‘targets’ that we extract from clutter are best described as consisting only of point sources. Although the model introduced in Section II describes objects as consisting of (point-like) scattering centres, it should be recognised that if there is more than of the order of one such scatterer per resolution cell then the cross section may be described in terms of a continuum scattering model. Typically we would use the continuum representation for extended objects (with many scatterers per resolution cell), and we would use the scattering centre representation for point-like targets. However this distinction is largely a matter of subjective viewpoint because the discrete model is a special case of the continuum model.

We shall use the continuum model in order to introduce a weaker form of prior knowledge about the form of targets than was expressed in the discrete model (the monopulse generalisation). If we consider a target in isolation (i.e. not embedded in clutter), then a weak form of prior knowledge that may be specified with high confidence is that the spatial extent of the target is limited. This information alone does not specify anything about the scattering centres that the target might possess; these are determined by the data. Thus we interpret the data in the light of the fact that it derives from an object that has a limited spatial extent. The basic principle that we have used here is similar to that used in monopulse; we are interpreting the data within the confines of a restrictive model of the object.

### B. The singular-value decomposition (SVD)

A particular way of interpreting data within the confines of a ‘support constraint’ is afforded by the singular-value decomposition (SVD) method [4, 6]. We shall present the standard derivation of the SVD method in this section, and present an information-theoretic justification for its use in Section III C and Section III D. It is advantageous to introduce some mathematical notion at this stage. For simplicity we shall present the results for the one-dimensional case only. Let us define the field (assumed scalar) that is scattered from the object as  $f(x)$ , the PSF of the imaging system as  $T(y, x)$  and the data samples as  $\mathbf{g}$ . It is important to note that  $\mathbf{g}$  is a vector of sample values at discrete positions  $y_i$  ( $i = 1, \dots, m$  for  $m$  data samples). The imaging equation then reads

$$g_i = \int_D dx T(y_i, x) f(x) + n_i \quad (3.1)$$

where the integral is over the domain  $D$  of support of the object, and  $n_i$  is the measurement noise that is introduced into the  $i$ th sample. The SVD proceeds by decomposing the operator  $T(y_i, x)$  as follows:

$$\int_D dx T(y_i, x) u^k(x) = (\lambda_k)^{\frac{1}{2}} v_i^k \quad (3.2)$$

$$\sum_{j=1}^m T(y_j, x) v_j^k = (\lambda_k)^{\frac{1}{2}} u^k(x) \quad (3.3)$$

where

$$\int_D dx u^{k*}(x) u^l(x) = \delta_{k,l} \quad (3.4)$$

and

$$\sum_{j=1}^m v_j^{k*} v_j^l = \delta_{k,l} \quad (3.5)$$

Thus the basis  $\mathbf{v}^k$  may serve as an orthonormal basis with which to decompose any particular image  $\mathbf{g}$ . If we ignore the effect of measurement noise, then each  $u^k(x)$  passes through the imaging system  $T(y_i, x)$  to give an attenuated  $\mathbf{v}^k$ . Furthermore if we assume that the one- and two-point noise statistics are given by

$$\langle n_i \rangle = 0 \quad (3.6)$$

$$\langle n_i n_j^* \rangle = N \delta_{i,j} \quad (3.7)$$

and that these specify all the statistical structure of the noise, then only those modes for which the inequality

$$\lambda_k \geq N \quad (3.8)$$

holds contain useful information on the form of the object. A data interpretation scheme may now be formulated as

$$f_{\text{rec}}(x) = \sum_k \frac{g'_k}{(\lambda_k)^{\frac{1}{2}}} u^k(k) \quad (3.9)$$

where the  $g'_k$  are the components of  $\mathbf{g}$  in the  $\mathbf{v}^k$  basis. The summation in Equation 3.9 should be taken only over those modes  $k$  that satisfy the inequality in Equation 3.8. This scheme gives a data interpretation (or reconstruction of  $f(x)$ ) that combines the prior knowledge of the support constraint  $D$  with the samples  $\mathbf{g}$ . It has been shown in simple cases that this gives rise to super-resolution [4, 6]. For the specific case of an imaging system that acts as a band-pass filter on  $f(x)$  and a support  $D$  that has the form of a single ‘top-hat’ function (i.e. the object is known to reside in a single well defined region), the degree of super-resolution increases as the length of the support is reduced and/or the level of image noise is reduced. This is entirely reasonable because the shorter the length of support the more possible reconstructions are being excluded from consideration, and the lower the level of image noise the more chance there is of a particular mode  $\mathbf{v}^k$  being ‘visible’ in the data.

However a major disadvantage of this SVD technique for our purposes is that the form of the prior knowledge is given as a support. We are concerned with super-resolving targets that are embedded in surrounding clutter, so  $f(x)$  is potentially non-zero for all  $x$ . This compels us to use a support of infinite extent, which amounts to not introducing any prior knowledge at all! The basic SVD technique must be extended in order to encompass our type of data interpretation problem. A partial solution to this problem has been presented by [5]. They extend the basic SVD technique by incorporating a ‘profile function’  $P(x)$  into the definition of the object. Thus they write the imaging equation as (compare with Equation 3.1)

$$g_i = \int_{-\infty}^{\infty} dx T(y_i, x) P(x) f(x) + n_i \quad (3.10)$$

This equation arises if an object (of infinite spatial extent) is illuminated non-uniformly. The  $P(x)$  factor then represents the amplitude of the illumination at position  $x$ , which gives rise to a modified SVD due to the softness of the effective support.

A possible alternative interpretation of Equation 3.10 is that the illumination is uniform, and that  $P(x)$  is associated with the object itself. Thus  $P(x)$  would encode prior knowledge of an estimate of the object profile [5]. Using this interpretation we could attempt to super-resolve targets that are embedded in clutter by defining a suitable  $P(x)$ .  $P(x)$  could then be decomposed into a weak component that has a slow spatial variation (representing clutter), plus localised strong components centred on each target. However the physical interpretation of the ‘soft’ support afforded by  $P(x)$  in Equation 3.10 is not as clear as that of the ‘hard’ support in Equation 3.1. This problem does not arise when  $P(x)$  is associated with the illumination, because it then represents a known physical quantity (amplitude of illumination). In order to derive the form of the soft support when  $P(x)$  is associated with the object itself, we shall adopt a more fundamental approach to data interpretation, namely Bayesian

inference, and we shall show how the SVD method (with profile function) may be derived.

### C. Bayesian object reconstruction

This section gives a review of the method used by [18] to reconstruct  $f(x)$ . We shall be using the language of probability density functions (PDF) extensively. We shall use the generic notation  $P[*]$  to denote the PDF over  $*$ . We formulate our prior knowledge of  $f(x)$  by using a PDF  $P[f(x)]$ ; this is the *a priori* PDF. Similarly we formulate our knowledge of the imaging process by using a conditional PDF  $P[g|f(x)]$ . We may then use Bayes rules to determine the *a posteriori* PDF  $P[f(x)|g]$ :

$$P[f(x)|g] = \frac{P[g|f(x)] P[f(x)]}{P[g]} \quad (3.11)$$

where

$$P[g] = \int [df(x)] P[g|f(x)] P[f(x)] \quad (3.12)$$

Thus the *a posteriori* PDF  $P[f(x)|g]$  combines the *a priori* PDF  $P[f(x)]$  (the prior knowledge) with  $P[g|f(x)]$  (the data). For a given joint set of prior knowledge and data we may drop the denominator factor  $P[g]$  in Equation 3.11 because it only affects the overall normalisation. We regard the *a posteriori* PDF as the basic construct from which all inferences about the data are to be drawn. We now wish to construct a model set of PDF which will prove to be adequate for the interpretation of target images (embedded in clutter). Furthermore we wish to demonstrate that the profile function method [5] may be derived as a special case of our PDF model.

We shall adopt a zero-mean gaussian model for the *a priori* PDF. This will not accurately represent the non-gaussian statistics that actually occur, but the model will suffice to specify the scale of the amplitude of  $f(x)$ . This amplitude scale can be identified with the profile function that was used to generalise the SVD.  $P[f(x)]$  is thus defined to be

$$P[f(x)] \propto \exp [-f^\dagger W^{-1} f] \quad (3.13)$$

where

$$f^\dagger W^{-1} f \equiv \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} dx dy f^*(x) W^{-1}(x, y) f(y) \quad (3.14)$$

This definition corresponds to the following one- and two-point statistics

$$\langle f(x) \rangle = 0 \quad (3.15)$$

$$\langle f^*(x) f(y) \rangle = W(x, y) \quad (3.16)$$

All other statistics may be expressed in terms of  $W(x, y)$  as well. We shall denote the imaging process by the following shorthand notation

$$g = T f + n \quad (3.17)$$

which in component form reads as

$$g_i = \int_{-\infty}^{\infty} dx T(y_i, x) f(x) + n_i \quad (3.18)$$

We shall assume that the effect of the measurement noise may be represented by a zero-mean gaussian PDF:

$$P[n] \propto \exp [-n^\dagger N^{-1} n] \quad (3.19)$$

which has the following one- and two-point statistics:

$$\langle n_i \rangle = 0 \quad (3.20)$$

$$\langle n_i^* n_j \rangle = N_{i,j} \quad (3.21)$$

Note that we have not assumed that the noise is white. The conditional PDF  $P[g|f(x)]$  may be obtained by replacing  $n$  by  $g - T f$  in Equation 3.19. When the Bayes rule is used we obtain the *a posteriori* PDF as

$$P[f(x)|g] \propto \exp \left[ - (g - T f)^\dagger N^{-1} (g - T f) - f^\dagger W^{-1} f \right] \quad (3.22)$$

This is the central result which we obtain by adopting the Bayesian approach to object reconstruction. We must rearrange the terms in order to obtain a more useful equation; thus

$$P[f(x)|g] \propto \exp \left[ - (f - \langle f \rangle)^\dagger (T^\dagger N^{-1} T + W^{-1}) (f - \langle f \rangle) \right] \quad (3.23)$$

where

$$\begin{aligned} \langle f \rangle &\equiv (T^\dagger N^{-1} T + W^{-1})^{-1} T^\dagger N^{-1} g \\ &= W T^\dagger (T W T^\dagger + N)^{-1} g \end{aligned} \quad (3.24)$$

As in the SVD method (Equation 3.9) we wish to present a single  $f_{\text{rec}}(x)$  as representative of our total state of knowledge of  $f(x)$ . A natural way in which we can make such a selection from the set of  $f(x)$  that are permitted by  $P[f(x)|g]$  is to select the average  $f(x)$  which is given in Equation 3.24. In full the various parts of Equation 3.24 read as

$$(TWT^\dagger + N)_{i,j} \equiv \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} dx' dx'' T(y_i, x') W(x', x'') T^*(y_j, x'') + N_{i,j} \quad (3.25)$$

$$(WT^\dagger)_i(x) \equiv \int_{-\infty}^{\infty} dx' W(x, x') T^*(y_i, x') \quad (3.26)$$

The matrix result in Equation 3.25 must be inverted before applying it to the data  $\mathbf{g}$ .

A particular point of interest is that the  $f_{\text{rec}}(x)$  that has been obtained above is the maximum *a posteriori* probability (MAP) reconstruction. This is because the PDF are all gaussian, and so their means are coincident with their modes. We may also obtain the MAP reconstruction  $f_{\text{MAP}}(x)$  by a minimisation procedure. The relevant objective function is given by (see Equation 3.22)

$$O[f(x)] \equiv (Tf - g)^\dagger N^{-1} (Tf - g) + f^\dagger W^{-1} f \quad (3.27)$$

which must be minimised with respect to  $f(x)$  to obtain  $f_{\text{MAP}}(x)$ . This is a generalisation of the weighted-minimum norm methods that have been studied previously [1, 2, 7, 8]. We believe that the Bayesian derivation of this result is more satisfying, because we have derived the ‘weight’  $W$  and the ‘regularisation parameter’  $N$  by appealing to the information content of the prior knowledge and the data (albeit expressed as simple gaussian PDF).

We may compare the reconstruction obtained in Equation 3.24 with that obtained by using the SVD by diagonalising the operator  $TWT^\dagger + N$ . For the purposes of this comparison we shall assume that the covariance in Equation 3.16 is  $\delta$ -correlated; thus

$$W(x, y) = |P(x)|^2 \delta(x - y) \quad (3.28)$$

where  $P(x)$  is a ‘profile function’ [5].  $|P(x)|^2$  (rather than  $P(x)$ ) appears in Equation 3.28 because  $W(x, y)$  measures the *two-point* statistics of the scattered field; therefore in our Bayesian object reconstruction scheme from complex data samples  $P(x)$  always appears in the combination  $|P(x)|^2$ .

Furthermore we shall assume that the noise covariance is given by

$$N_{i,j} = N \delta_{i,j} \quad (3.29)$$

Diagonalising  $TWT^\dagger$

$$T |P|^2 T^\dagger \mathbf{v}'^k = \lambda'_k \mathbf{v}'^k \quad (3.30)$$

where

$$\sum_{j=1}^m v'_j{}^{k*} v'_j{}^l = \delta_{k,l} \quad (3.31)$$

and defining the functions  $u'^k(x)$  by

$$u'^k(x) \equiv \frac{P^* T^\dagger v'^k}{(\lambda'_k + N)^{\frac{1}{2}}} \quad (3.32)$$

then  $f_{\text{rec}}(x)$  (derived from Equation 3.24) may be expressed as

$$f_{\text{rec}}(x) = P(x) \sum_{k=1}^m \frac{g'_k}{(\lambda'_k + N)^{\frac{1}{2}}} u'^k(x) \quad (3.33)$$

where the  $g'_k$  are the components of  $\mathbf{g}$  in the basis  $\mathbf{v}'^k$ . We may also obtain the pair of coupled equations

$$TP u'^k = (\lambda'_k)^{\frac{1}{2}} \left[ \frac{\lambda'_k}{\lambda'_k + N} \right]^{\frac{1}{2}} v'^k \quad (3.34)$$

$$P^* T^\dagger v'^k = (\lambda'_k)^{\frac{1}{2}} \left[ \frac{\lambda'_k + N}{\lambda'_k} \right]^{\frac{1}{2}} u'^k \quad (3.35)$$

and the normalisation condition

$$\int_{-\infty}^{\infty} dx u'^{k*}(x) u'^l(x) = \frac{\lambda'_k}{\lambda'_k + N} \delta_{k,l} \quad (3.36)$$

In the limit of a large signal-to-noise ratio these results are precisely the same as those obtained by using the profile function generalisation of the SVD [5]. The differences that arise due to the finite signal-to-noise ratio can be traced to the different forms of regularisation that are implicit in the two reconstruction schemes. Whilst these differences are probably not important, the Bayesian approach using PDF is the more desirable method because of the greater ease with which it is physically interpreted.

#### D. Information theory

Let us now examine the information content of  $P[f(x)|\mathbf{g}]$  as given in Equation 3.23. In order to keep the derivation simple we shall impose the restrictions that were given in Equation 3.27 and Equation 3.28. We must ‘diagonalise’  $P[f(x)|\mathbf{g}]$  in order to interpret it. For this purpose we shall replace  $\langle f \rangle$  by the expansion given in Equation 3.33 (recall that  $f_{\text{rec}} = \langle f \rangle$ ), and we shall similarly expand the full  $f$  as follows

$$f(x) = P(x) \sum_{k=1}^m \frac{f'_k}{(\lambda'_k + N)^{\frac{1}{2}}} u'^k(x) + f_c(x) \quad (3.37)$$

where the expansion coefficients are given by

$$f'_k \equiv \frac{\lambda'_k + N}{\lambda'_k} \sum_{j=1}^m v'_j{}^{k*} \int_{-\infty}^{\infty} dx T(y_j, x) f(x) \quad (3.38)$$

In Equation 3.38 we see that  $f$  is first of all imaged (using  $T$ ), and then projected onto the  $\mathbf{v}^k$  basis, so the expansion coefficient  $f'_k$  indicates the amplitude of the image of  $f$  in the  $k$ th data space mode. The components of  $f$  that do not project onto this basis lie in the orthogonal complement subspace and we shall denote their sum as  $f_c$ . These components are ‘invisible’ to the data space, so whatever  $\mathbf{g}$  is measured we cannot infer anything more about  $f_c$  than is already contained in the *a priori* PDF. The ‘visibility’ of each of the remaining components is given by the corresponding eigenvalue. For eigenvalues

that are less than  $O(N)$  the corresponding data space mode has insufficient signal energy present to dominate the noise, and so little can be inferred about the corresponding component of  $f$  than is already implied by  $P[f(x)]$ . An important point to be noted here is that the positions of the data samples can strongly influence the form of the eigenvalue spectrum, and it is worthwhile to optimise the sample positions in order to maximise the potential information content of the data [17, 19].

Inserting the expressions for  $f$  and  $\langle f \rangle$  into Equation 3.23 we obtain

$$P[f(x)|\mathbf{g}] \propto \exp \left[ f_c^\dagger |P|^{-2} f_c \right] \exp \left[ - \sum_{k=1}^m |f'_k - g'_k|^2 \left( \frac{1}{N} - \frac{1}{\lambda'_k + N} \right) \right] \quad (3.39)$$

where we have used the fact that  $f_c$  does not project onto the data space in order to eliminate some terms. The first factor in  $P[f(x)|\mathbf{g}]$  corresponds to a piece of the *a priori* PDF that is not changed in passing to the *a posteriori* PDF;  $f_c$  is invisible to the data space. The second factor in  $P[f(x)|\mathbf{g}]$  is dependent on the data; it expresses the influence that  $\mathbf{g}$  has on the *a posteriori* PDF. Each mode exerts an independent effect on  $P[f(x)|\mathbf{g}]$ , so each mode carries an independent piece of useful information about  $f$ .

The above information-theoretic result has been obtained without explicit recourse to information theory *per se*. We shall now present a derivation that is couched in the more general language of information theory [27, 28]. The results that we present below have been derived by Luttrell [17, 19].

Let us assume that the statistics of a signal are fully specified by the probability of occurrence  $P[x]$  of each signal state  $x$ . The case of correlated signals can be dealt with by constructing ‘blocks’ of signal states that are long compared with the correlation time, and using the joint probability  $P[x_1, x_2, \dots]$  to specify (almost) fully the statistics. The potential information content of the signal is given by its entropy  $H[x]$ , which is defined thus

$$H[x] \equiv - \int [dx] P[x] \log_2(P[x]) \quad (3.40)$$

where  $[dx] P[x]$  may be regarded as a measure that is used for averaging the logarithm. We shall use the generic notation  $H[*]$  to denote the entropy of  $*$ . The interpretation of  $H[x]$  is that it is the number of bits of useful information which could be communicated each time an  $x$  is selected (with a frequency determined by  $P[x]$ ). Thus after  $N$  such selections the number of distinct ‘messages’ is  $2^{NH}$ . This is the original Shannon interpretation of the role of entropy; he was primarily interested in its implications for communication theory. By measuring the statistics of an information source (specified by its PDF),

one can then calculate its entropy, which enables one to deduce the upper limit to the number of distinct messages of a given length  $N$  (where  $N$  is large). This upper limit implies the minimum communication band width which must be supplied in order to carry the messages faithfully; a very useful quantity to know. In practice of course the problem of transforming the messages into a form that does not waste band width is a non-trivial task which has given rise to the study of data compression and coding theory.

We shall now present an alternative interpretation of entropy which is expressed in terms that are related to the problem of information measurement in imaging systems [17, 19]. To be specific let us discuss the *a priori* PDF  $P[f(x)]$  in information theoretic terms.  $f(x)$  now takes on the role of a signal, and  $P[f(x)]$  fully specifies its statistics; this implies that realisations of  $f(x)$  are independent of each other. Such a situation arises if we consider an ensemble of  $N$  realisations of  $f(x)$ , each of which has been independently selected from  $P[f(x)]$ . The entropy  $H[f(x)]$  is defined as

$$H[f(x)] \equiv - \int [df(x)] P[f(x)] \log_2(P[f(x)]) \quad (3.41)$$

and  $2^{NH}$  now gives the number of distinct joint realisations from  $N$  samples of  $P[f(x)]$ . A large classroom of  $N$  students who are each making the same measurement once would be a situation in which  $2^{NH}$  would be useful. It could be used to estimate the number of distinct ways in which the joint results (of the whole classroom) could occur.

Having introduced our interpretation of entropy we may now use the other results of Shannon’s information theory. We have a need only for the expression for transinformation  $I[f(x), \mathbf{g}]$  (or rate of transmission, or

mutual information)

$$I[f(x), \mathbf{g}] \equiv H[f(x)] - H[f(x)|\mathbf{g}] \quad (3.42)$$

$$\equiv H[\mathbf{g}] - H[\mathbf{g}|f(x)] \quad (3.43)$$

$$\equiv H[f(x)] + H[\mathbf{g}] - H[f(x), \mathbf{g}] \quad (3.44)$$

where the conditional entropies  $H[f(x)|\mathbf{g}]$  and  $H[\mathbf{g}|f(x)]$  are averaged results over  $\mathbf{g}$  and  $f(x)$  respectively. These three expressions for  $I[f(x), \mathbf{g}]$  are inter-related by Bayes' rule.  $I[f(x), \mathbf{g}]$  measures the extent to which  $f(x)$  and  $\mathbf{g}$  are correlated with each other. Equation 3.42 specifies the extent to which the uncertainty in  $f(x)$  is reduced by measuring  $\mathbf{g}$ , Equation 3.43 specifies the extent to which knowledge of  $f(x)$  reduces the uncertainty as to what  $\mathbf{g}$  will be measured, and Equation 3.44 specifies the extent to which the sum of the separate uncertainties of  $f(x)$  and  $\mathbf{g}$  exceed their joint uncertainty. All these quantities are equivalent; the word 'trans-information' is suggested by Equation 3.42 and Equation 3.43, and the phrase 'mutual information' is suggested by Equation 3.44. An important property that  $I[f(x), \mathbf{g}]$  possesses is measure independence [19], so we have a consistent measure of information.

We may now use our earlier expressions for  $P[f(x)]$  (Equation 3.13),  $P[\mathbf{g}|f(x)]$  (Equation 3.19 with  $n \rightarrow g - T f$ ), and  $P[f(x)|\mathbf{g}]$  (Equation 3.23) to derive  $I[f(x), \mathbf{g}]$ . Starting from Equation 3.42 after some manipulation [19] we obtain

$$I[f(x), \mathbf{g}] = \log_2 \left[ \frac{\det(W)}{\det(T^\dagger N^{-1} T + W^{-1})^{-1}} \right] \quad (3.45)$$

and if we start from Equation 3.43 we obtain

$$I[f(x), \mathbf{g}] = \log_2 \left[ \frac{\det(T W T^\dagger + N)}{\det(N)} \right] \quad (3.46)$$

In Equation 3.45 the 'matrices'  $T^\dagger N^{-1} T$  and  $W$  are in fact functions of two continuous variables (not two discrete indices), so the determinants must be combined in order to avoid infinities occurring during intermediate steps of the calculation. This problem does not occur with the necessarily discrete 'matrix' forms of  $T W T^\dagger$  and  $N$  which appear in Equation 3.46. With this proviso in mind the results in Equation 3.45 and Equation 3.46 are equivalent.

We may interpret these results as follows. Firstly we must identify what the various 'matrices' are:

1.  $W$  = covariance of  $f(x)$  according to the *a priori* PDF (see Equation 3.13);
2.  $N$  = covariance of  $n$  according to the measurement noise PDF (see Equation 3.19);
3.  $T W T^\dagger + N$  = covariance of  $g$ , which may easily be derived from 1 and 2 and the imaging Equation 3.17;
4.  $(T^\dagger N^{-1} T + W^{-1})^{-1}$  = covariance of  $f(x)$  according to the *a posteriori* PDF (see Equation 3.23).

A determinant of a covariance 'matrix' measures the 'volume' of signal space that is likely to be occupied by the corresponding signal. The ratio of determinants in Equation 3.45 is the ratio of the volume of the covariance associated with *a priori* PDF to the volume of the covariance associated with the *a posteriori* PDF. The corresponding ratio in Equation 3.46 is the ratio of the volume of the covariance of the signal plus noise to the volume of the covariance of the measurement noise alone. Both of these quantities have an intuitive interpretation which is consistent with  $I[f(x), \mathbf{g}]$  being a measure of the useful information that is acquired by measuring  $\mathbf{g}$ . Thus Equation 3.45 measures how much the *a priori* PDF is 'squeezed' by  $\mathbf{g}$  when it is converted into an *a posteriori* PDF, and Equation 3.46 measures how much of  $\mathbf{g}$  is true signal. These two quantities are equivalent when measured according to Shannon's information theory. It is important to note that we have not assumed that  $W$  and/or  $N$  are  $\delta$ -correlated to derive the results presented in Equation 3.45 and Equation 3.46, so they will hold true for an arbitrary correlated gaussian PDF model. Furthermore it may easily be checked that non-zero mean PDF do not change this result.

In order to complete the comparison of the formal information-theoretic results with Equation 3.39, we must diagonalise the two expressions for  $I[f(x), \mathbf{g}]$  for the  $\delta$ -correlated model introduced in Equation 3.28 and Equation 3.29. This is a trivial task: the  $P u^k(x)$  diagonalise Equation 3.45, and the  $v^k$  diagonalise Equation 3.46. This is consistent with the result that we obtained in Equation 3.39. The diagonalised expressions for  $I[f(x), \mathbf{g}]$  are the same (of course) and are given by

$$I[f(x), \mathbf{g}] = \sum_{k=1}^m I_k \quad (3.47)$$

where

$$I_k \equiv \log_2 \left( \frac{\lambda'_k}{N+1} \right) \quad (3.48)$$

The consistency of the formal information-theoretic results and the factorisation of the *a posteriori* PDF result in Equation 3.39 which was demonstrated by transforming to the  $v^k$  and  $P u^k(x)$  bases (i.e. an SVD type of approach) provides a powerful argument in favour of the SVD, at least in cases where the use of gaussian PDF can be justified. However non-gaussian *a priori* PDF are commonplace (more so than non-gaussian measurement noise PDF), and in such cases an alternative argument must be advanced for the validity of an SVD type of approach. In our application where we wish to superresolve targets that are embedded in clutter we recognise that a gaussian *a priori* PDF is not a perfect model. However we may use it as a first approximation in order to give an indication of the scale of the amplitude of  $f(x)$ .

### E. Super-resolution results

We shall now use Equation 3.24 to calculate  $f_{\text{rec}}(x)$  for a simulated target embedded in clutter. For the purposes of numerical computation we shall simulate a two-dimensional imaging system whose point spread function is that of a square transparent aperture. We shall model a data acquisition system that measures complex ( $I$  and  $Q$ ) samples on a square lattice that has the Nyquist sampling frequency. We shall be simulating localised targets embedded in clutter, so we shall not need all the data. Instead we shall choose a  $9 \times 9$  array of data samples that contains the target of interest approximately at its centre. If the target is not too bright relative to the surrounding clutter, then there is very little target information lost by discarding the remainder of the data. A rule of thumb that may be used to determine the minimum size of the data array that should be used is that all of the target side-lobes that dominate the surrounding clutter should be contained within the chosen array. We shall discretise the two-dimensional object space coordinate so that there are 16 object lattice points per data lattice point (i.e. 16 per two-dimensional resolution cell). Furthermore we shall register the position of every fourth object lattice point (in each dimension) with a data lattice point, and we shall not place any object lattice points outside the bounds of the  $9 \times 9$  data array. This gives a  $33 \times 33$  ( $33 = 4 \times (9-1) + 1$ ) object array. It is essential to choose a fine enough discretisation of object space that no structure on a finer scale could be deduced from the data. Our choice of a factor of four (in each dimension) would thus permit a super-resolution factor of the order of four in each dimension to be attained before errors introduced by discretisation overwhelm the results. The conditions that are required for super-resolution of this order do not arise in SAR applications, so our discretisation of object space is sufficiently fine for our purposes.

Having set up the lattices we now require estimates of  $W$  and  $N$ . For simplicity we have restricted our attention to  $\delta$ -correlated models as in Equation 3.28 and Equation 3.29. The measurement noise level may be deduced from the properties of the SAR. In practice however (for Nyquist-sampled data at least) we may assume that the measurement noise is zero, because Nyquist-sampled clutter is essentially  $\delta$ -correlated and so appears substantially like a very large measurement error would appear. In this case we need not include the true measurement noise term because it would only increment by a very small amount an already large clutter term.

The target and clutter prior knowledge are introduced by specifying the function  $|P(x)|^2$ . Because  $|P(x)|^2$  plays the role of a cross section (per unit length) the task facing us is to specify our prejudices about what types of cross section are reasonable for target/clutter. As a first approximation we can estimate the cross section to be constant everywhere. It does not matter what value we choose because we have set  $N = 0$ , and there are no other scales involved in the reconstruction method of Equation

### 3.24.

With the choice  $N = 0$  and  $|P(x)|^2 = \text{constant}$  we may compute the reconstruction operator  $WT^\dagger(TWT^\dagger + N)^{-1}$  that is required in Equation 3.24, and use it to calculate  $f_{\text{rec}}(x)$  from  $\mathbf{g}$ . This operator needs to be computed only once because it does not depend on  $\mathbf{g}$ . Although it is not necessary we have computed the  $\mathbf{v}'^k$  and  $Pu'^k(x)$  bases for this case, and we present the first 81 object basis functions (in order of decreasing eigenvalue) in Figure 7. Note how as the eigenvalue decreases the number of nodal lines per unit length generally increases. This is because the finer structure that is associated with a larger number of nodal lines per unit length is attenuated more when it passes through the filter  $T$ . Despite this effect the eigenvalues all have approximately the same value because the data array is Nyquist-sampled and  $T$  is a square transparent aperture, and in fact the small differences are due solely to edge effects caused by using a finite-sized piece of lattice. It has been shown [18] that the case  $|P(x)|^2 = \text{constant}$  gives a reconstruction that is an interpolation of the data, which reduces to the usual ‘sinc interpolation’ formula when a rectangular transparent aperture with Nyquist sampling is used. The  $Pu'^k(x)$  that are displayed in Figure 7 may therefore be used to interpolate the data (in general) in order to obtain a better idea of what information is present in the data samples alone.

We shall use this preliminary interpolated form of the data to make the cross section estimates that are required in order to introduce prior knowledge. As we commented at the end of Section III C we may decompose this estimation problem into two components: (i) estimating the clutter cross section; and (ii) estimating the target cross section.

The clutter cross section has a slow spatial variation which we shall model as being constant, and whose value is estimated by taking a local average intensity (excluding any pixels with intensities exceeding a suitably chosen target threshold). The intensities of those pixels that were thresholded out in this process are used directly as estimates of the corresponding (local target) cross section. This process whereby an interpolated form of the data is used to generate ‘prior knowledge’, which will be used to obtain a more refined reconstruction, may seem incestuous. However it is important to realise that the prior knowledge resides not in the data, but in the ‘rules and regulations’ that are used to interpret the data. In our case we are applying the rule that locally bright regions of the data are exceedingly likely to have derived from local concentrations of cross section, rather than any of the myriad of other possibilities. This rule is not a fundamental constraint on the form of objects; instead it has been derived from empirical observation of typical SAR images.

We may now present as a sequence of pictures (Figure 8(a)-(d)) the ideas that we have developed so far. In Figure 8(a) we show a high-resolution image of a point target that is embedded in clutter. There are clear side-lobes

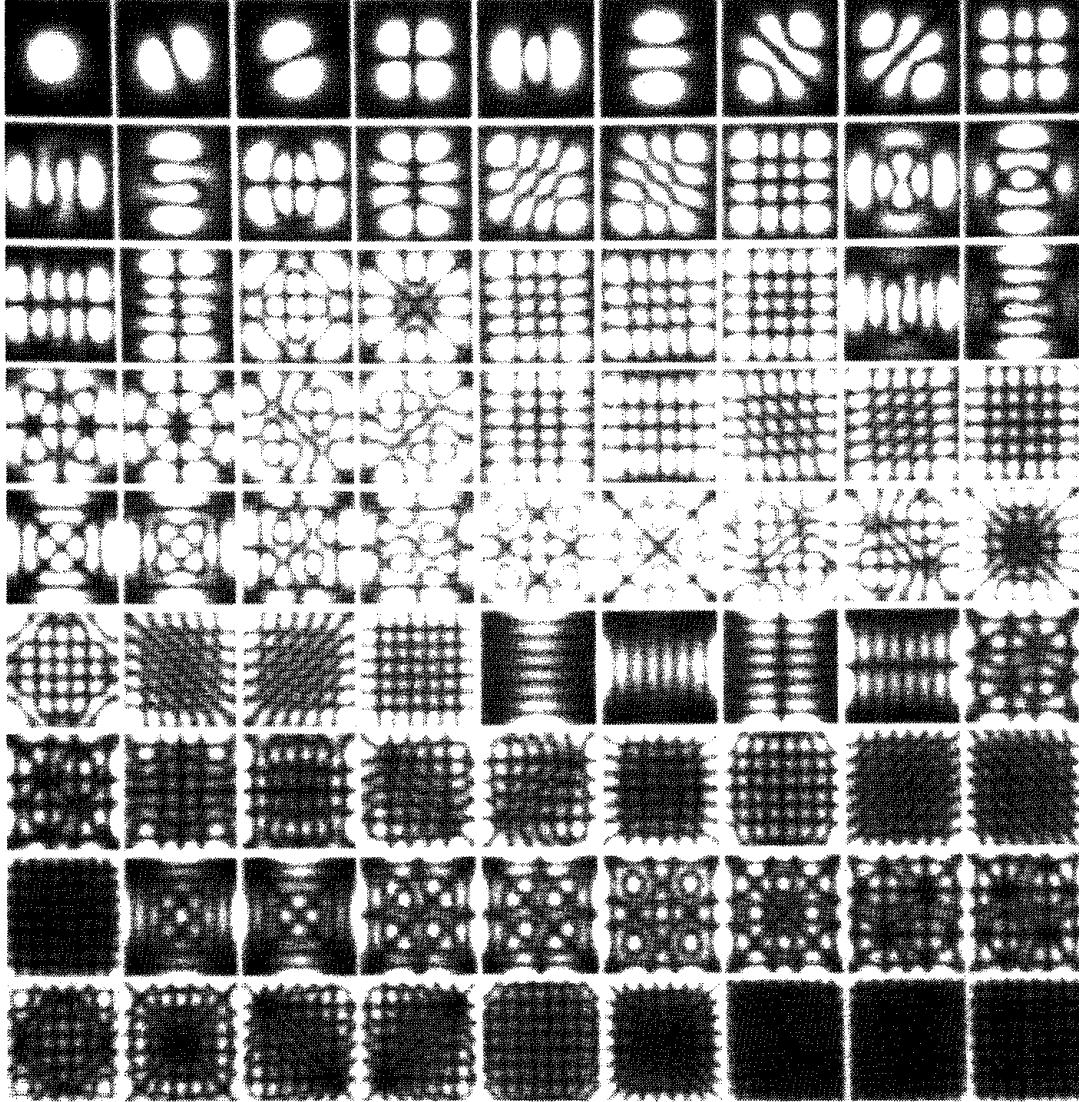


Figure 7: A montage of the 81 object space functions associated with a  $9 \times 9$  square lattice of Nyquist-sampled data.

present, which could be used to deduce that the target is indeed composed of a single scatterer. In Figure 8(b) we show the same target/clutter combination imaged at a resolution four times poorer (in each dimension); the SNR is 21 dB in Figure 8(b). Now the side-lobes are less distinct because there is a greater amount of clutter cross section present in each resolution cell, or equivalently because the target image has been smeared over a greater region of the image plane. The target structure (i.e. single scatterer) would therefore be more difficult to ascertain at the lowest resolution. In Figure 8(a) and Figure 8(b) we have interpolated the Nyquist-sampled data to give a smooth image. We may give a rudimentary demonstration of super-resolution if we apply the technique that we outlined in the previous paragraph to the interpolated low-resolution data shown in Figure 8(b).

Figure 8(c) depicts the estimated cross section, which shows some evidence of ‘false alarms’ where target side-lobes have been incorrectly identified as minor targets. These faults could be rectified by using a more sophisticated cross section estimation procedure, but we shall not present such results here. Figure 8(d) shows the reconstruction that is given by Equation 3.24 with  $N = 0$  and  $W(x, y) = |P(x)|^2 \delta(x - y)$ . The central peak of the reconstruction is clearly narrower and higher than the corresponding peak in the original image (Figure 8b)). The surrounding clutter has not been affected significantly by the reconstruction process, except where false alarms were introduced during cross section estimation. This simple reconstruction method will super-resolve the simplest type of target image, whilst not introducing an unacceptable amount of spurious detail into the surround-

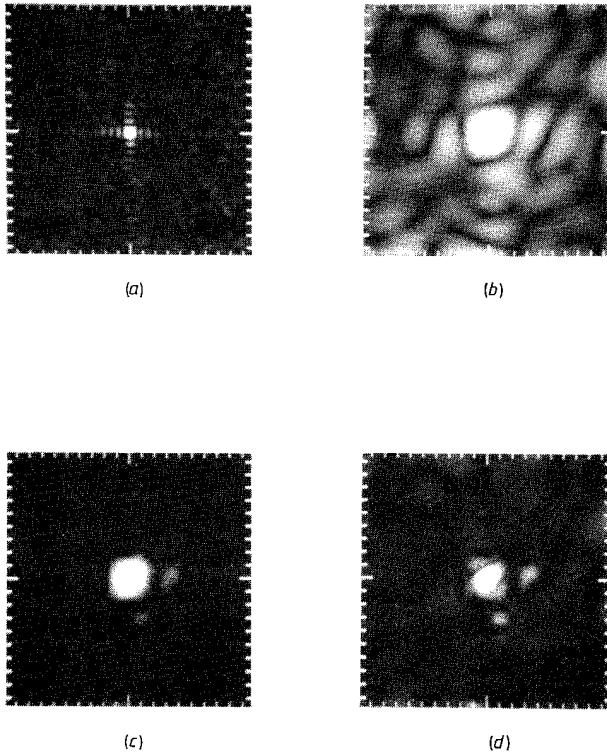


Figure 8: (a) The high-resolution image of a point target in clutter. (b) The low-resolution image of a point target in clutter. (c) The estimated cross section obtained from Figure 8(b). (d) The super-resolved reconstruction of Figure 8(b), obtained using the cross section estimate in Figure 8(c).

ing clutter. Even in this simple example the degree of super-resolution varies from point to point in the reconstruction, so we cannot assign a global super-resolution factor. This problem of defining a super-resolution factor becomes more acute as the structure of the object becomes more complicated, so we shall not attempt to give a universal definition of super-resolution factor. The universal definition given by [6] relies on the fact that their support is not dependent on the data (unlike ours), and so they may estimate the number of visible degrees of freedom from the signal-to-noise ratio. However even this definition gives only the potential super-resolution factor, because in practice information about the object is not recorded in all the visible degrees of freedom. A simple example of this effect is the fact that two in-phase point targets are more difficult to resolve than two anti-phase point targets when the observed signal-to-noise ratio is adjusted to be the same in both cases.

In Figure 9 we show the first 16 functions  $P u'^k(x)$ , which correspond to the super-resolution example given above. As in Figure 7 they are presented in order of decreasing eigenvalue. Note how the first few modes concentrate almost exclusively on a small region that is contained within the central peak of Figure 8(c), whereas there is no such concentration evident in the modes in

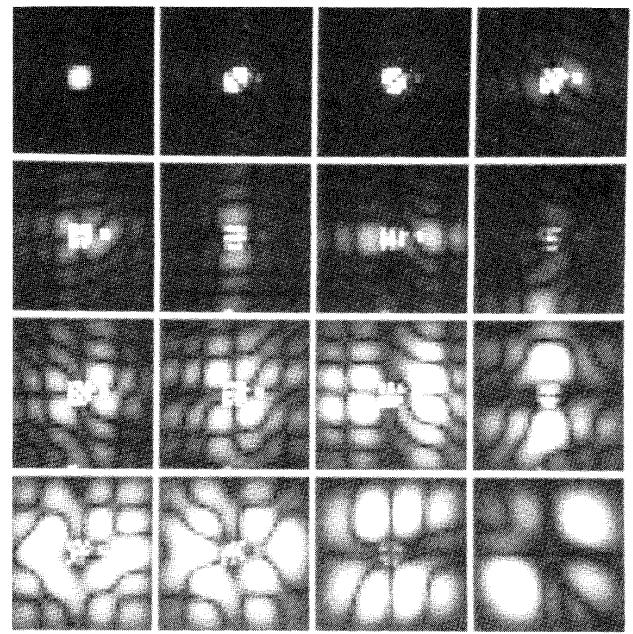


Figure 9: A montage of the first 16 object space functions associated with the estimated cross section in Figure 8(c).

Figure 7. This is how super-resolution is achieved: a small region of the object has concentrated into it a large amount of the information that is present in the data (see Equation 3.39). We have not displayed the remainder of the modes because they concentrate their attention mostly on the surrounding clutter region.

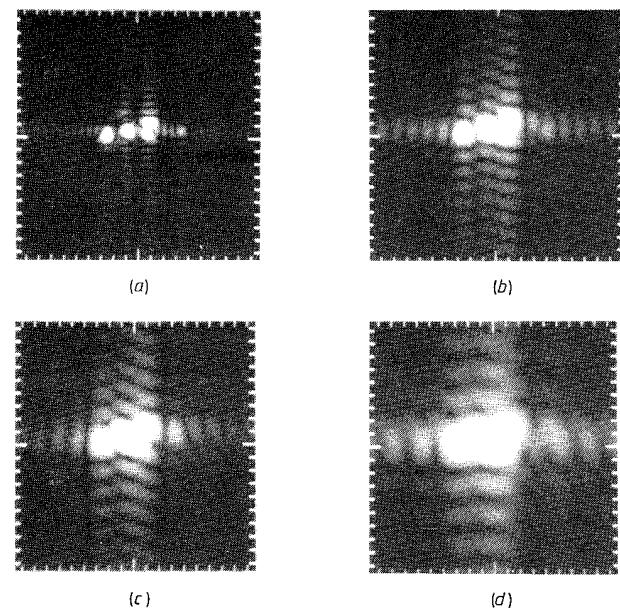


Figure 10: Four different resolution images of a simulated target without clutter present. The resolution ratios are (a) : (b) : (c) : (d) = 8 : 4 : 3 : 2.

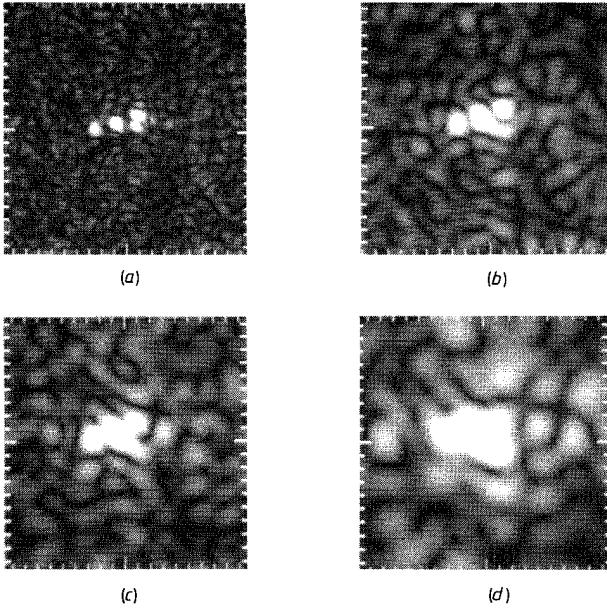


Figure 11: Four images corresponding to those given in Figure 10(a) to Figure 10(d), but with the simulated target embedded in clutter.

In Figure 10 we show interpolated images at different resolutions of a simulated target without surrounding clutter, and in Figure 11 we show the same images with surrounding clutter present. Figure 10(a) (Figure 11(a)) has the highest resolution of all, Figure 10(b), (Figure 11(b)) has a resolution two times poorer than Figure 10(a) (Figure 11(a)), Figure 10(c) (Figure 11(c)) has a resolution  $\frac{8}{3}$  times poorer than Figure 10(a) (Figure 11(a)), and Figure 10(d) (Figure 11(d)) has a resolution four times poorer than Figure 10(a) (Figure 11(a)). Figure 10 demonstrates the obvious loss of target information that occurs as the resolution becomes poorer. Figure 11 demonstrates that the effect of the surrounding clutter is to destroy more target information as the resolution becomes poorer.

We shall now super-resolve the target image in Figure 11(d); this has a SNR of 20 dB and it is the worst case shown. In Figure 12(a) we show the cross section that is estimated from the interpolated image in Figure 11(d). The clutter background has been well separated from the target, but there are two false alarms where the clutter has broken through the threshold. Figure 12(b) shows the reconstruction that is obtained. If we compare this with the original image in Figure 11(d), we observe that the target has more detail present. Furthermore if we compare Figure 12(b) with the higher-resolution version of the original image in Figure 11(c) (1.5 times higher resolution), we see that the target component is approximately the same; we have obtained a super-resolution factor of 1.5 (in each dimension). On the other hand the clutter component of the reconstruction in Figure 12(b) is substantially the same as that in the original image in Figure 11(d). The only differences lie around the false

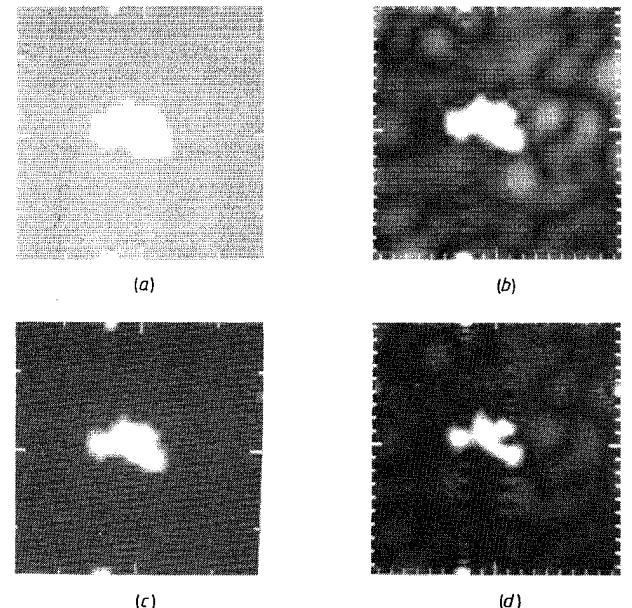


Figure 12: (a) The estimated cross section obtained from Figure 11(d). (b) The super-resolved reconstruction of Figure 11(d) using the cross section estimate in Figure 12(a). (c) The estimated cross section obtained from Figure 12(b). (d) The super-resolved reconstruction of Figure 12(b) obtained using the cross section estimate in Figure 12(c).

alarms; these could be removed by using a more sophisticated cross section estimation procedure.

So far we have incorporated a limited amount of prior knowledge about target structure. We have introduced the ‘fact’ that a bright peak in the interpolated image derives from a localised region of greater cross section. Thus the brightness of the peak causes a threshold to be exceeded, and we go on to claim that all such peaks derive from targets. We cannot be 100% sure that this interpretation of bright peaks is correct, but it is certainly true that the brighter the peak is (relative to the background) the more likely it is to be correct; we therefore set our threshold high enough that the false alarm rate is reduced to an acceptable level. The procedure for setting the threshold level may be derived from a suitable clutter model (see Section II).

If we have set the threshold sufficiently high that we are highly confident that the data interpretation afforded by the super-resolved reconstruction (in Figure 12(b)) is correct, then we may use the reconstruction as if it were the image itself. The reconstruction is a collation of the low-resolution data and low-grade prior knowledge on the form of typical targets. We may now embark upon another stage of object reconstruction by basing our estimate of the cross section on the reconstruction already obtained. Such a cross section estimate, which is based upon the reconstruction in Figure 12(b), is given in Figure 12(c). When Figure 12(c) is compared with Figure 12(a) it is apparent that finer structure is now

being included in the estimate. The corresponding reconstruction (using the estimate in Figure 12(c)) is shown in Figure 12(d). This shows more detail than the first reconstruction in Figure 12(b), and it has a comparable resolution to the image in Figure 11(b) (two times the resolution). Our confidence in the correctness of the second reconstruction is lower than that for the first reconstruction, because we assumed that the first reconstruction could be used as a super-resolved image (of the target). Clearly we must halt this iterative reconstruction procedure when the required super-resolution is obtained, or when our confidence drops to the lowest acceptable value (whichever is the sooner). A paper that gives a formal derivation of this iterated approach to super-resolution by appealing to Bayesian reconstruction principles is in preparation [20].

#### IV. CONCLUSIONS

The task of interpreting SAR images is very difficult. There are two basic reasons for this: the image content

has a very complex structure, and there are a large number of data. It would be exceedingly useful to be able to summarise the content of a SAR image in terms of a few significant features; conventionally this amounts to detecting ‘targets’ in a ‘clutter’ background. In Section II we presented a model of SAR terrain clutter that could be used to define a target detection threshold. Any target detections would then be fed to the super-resolution technique that is presented in Section III. The combination of these two techniques (and others that we have not reviewed here) could be used to compress drastically the storage requirements of SAR data, and to provide a means whereby SAR data may be interpreted.

- 
- [1] J B Abbiss, C de Mol, and H S Dhadwal, *Regularised iterative and non-iterative procedures for object restoration from experimental data*, Optica Acta **30** (1983), no. 1, 107–124.
  - [2] J B Abbiss, M Defrise, C de Mol, and H S Dhadwal, *Regularised iterative and non-iterative procedures for object restoration in the presence of noise: an error analysis*, Journal of the Optical Society of America **73** (1983), no. 11, 1470–1475.
  - [3] M Abramowitz and I A Stegun, *Handbook of mathematical functions*, Dover, New York, 1964.
  - [4] M Bertero, P Brianzi, P Parker, and E R Pike, *Resolution in diffraction-limited imaging, a singular value analysis III. The effect of sampling and truncation of the data*, Optica Acta **31** (1984), no. 2, 181–201.
  - [5] M Bertero, C de Mol, E R Pike, and J G Walker, *Resolution in diffraction-limited imaging, a singular value analysis IV. The case of uncertain localisation or non-uniform illumination of the object*, Optica Acta **31** (1984), no. 8, 923–946.
  - [6] M Bertero and E R Pike, *Resolution in diffraction limited imaging, a singular value analysis I. The case of coherent illumination*, Optica Acta **29** (1982), no. 6, 727–746.
  - [7] C L Byrne, R M Fitzgerald, M A Fiddy, T J Hall, and A M Darling, *Image restoration and resolution enhancement*, Journal of the Optical Society of America **73** (1983), no. 11, 1481–1487.
  - [8] A M Darling, T J Hall, and M A Fiddy, *Stable non-iterative object reconstruction from incomplete data using prior knowledge*, Journal of the Optical Society of America **73** (1983), no. 11, 1466–1469.
  - [9] \*\*\*INCOMPLETE\*\*\*S C Giess, R J Miller, and C J Oliver, \*\*\*incomplete\*\*\*, Remote Sensing Society and British Pattern Recognition Association Workshop on Image Understanding in Remote Sensing (London), 1985.
  - [10] E Jakeman, *On the statistics of K-distributed noise*, Journal of Physics A: Mathematical and General **13** (1980), no. 1, 31–48.
  - [11] E Jakeman, E R Pike, and P N Pusey, *Photon correlation study of stellar scintillation*, Nature **263** (1976), no. 5574, 215–217.
  - [12] E Jakeman and P N Pusey, *Non-Gaussian fluctuations in electromagnetic radiation scattered by random phase screen I. Theory*, Journal of Physics A: Mathematical and General **8** (1975), no. 3, 369–410.
  - [13] ———, *A model for non-Rayleigh sea echo*, IEEE Transactions on Antennas and Propagation **24** (1976), no. 6, 806–814.
  - [14] ———, *RADAR-77*, ch. Statistics of non-Rayleigh microwave sea echo, pp. 105–109, IEE, London, 1977.
  - [15] ———, *Significance of K-distributions in scattering experiments*, Physical Review Letters **40** (1978), no. 9, 546–550.
  - [16] J-S Lee, *Digital image enhancement and noise filtering by use of local statistics*, IEEE Transactions on Pattern Analysis and Machine Intelligence **2** (1980), no. 2, 165–168.
  - [17] S P Luttrell, *A new method of sample optimisation*, Optica Acta **32** (1985), no. 3, 255–257.
  - [18] ———, *Prior knowledge and object reconstruction using the best linear estimate technique*, Optica Acta **32** (1985), no. 6, 703–716.
  - [19] ———, *The use of transinformation in the design of data sampling schemes for inverse problems*, Inverse Problems **1** (1985), no. 3, 199–218.
  - [20] ———, *A Bayesian derivation of an iterative autofocus/super-resolution algorithm*, Inverse Problems **6** (1990), no. 6, 975–996.
  - [21] C J Oliver, *Fundamental properties of high-resolution sideways-looking radar*, IEE Proceedings F **129** (1982),

- no. 6, 385–402.
- [22] ———, *A model for non-Rayleigh scattering statistics*, Optica Acta **31** (1984), no. 6, 701–722.
  - [23] ———, *Correlated K-distributed clutter models*, Optica Acta **32** (1985), no. 12, 1515–1547.
  - [24] ———, *A model for non-Rayleigh scattering statistics*, Proceedings of the IMA Conference on Wave Propagation and Scattering (Oxford) (B J Uscinski, ed.), Clarendon Press, 1986, p. 155.
  - [25] C J Oliver and R J Tough, *On the simulation of correlated K-distributed random clutter*, Optica Acta **33** (1986), no. 3, 223–250.
  - [26] G Parry, P N Pusey, E Jakeman, and J G McWhirter, *Focussing by a random phase screen*, Optics Communications **22** (1977), no. 2, 195–201.
  - [27] C E Shannon, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 3, 379–423.
  - [28] ———, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 6, 623–656.
  - [29] M Skolnik, *Introduction to radar systems*, McGraw-Hill, New York, 1981.
  - [30] K D Ward, *Compound representation of high resolution sea clutter*, Electronics Letters **17** (1981), no. 16, 561–565.
  - [31] ———, *RADAR-82*, ch. A radar sea clutter model and its application to performance assessment, pp. 203–207, IEE, London, 1982.

# An Optimisation of the Metropolis Algorithm for Multibit Markov Random Fields \*

S P Luttrell

*Royal Signals and Radar Establishment, St Andrews Road, Great Malvern, Worcs, WR14 3PS, UK*

We present a version of the Metropolis algorithm which is suitable for rapidly updating multibit Markov random fields. The fundamental objects on which the algorithm operates are the constituent binary digits of the field variables. A speed enhancement factor of  $\mathcal{O}(\frac{2^N}{N})$  (for an  $N$ -bit representation) is obtained in comparison with a conventional Metropolis algorithm.

## I. LETTER

The purpose of this Letter is to explain a method of sampling efficiently from complicated *a posteriori* PDFs. This method has applications in solving inverse problems.

Let us consider the generic measurement

$$g_i = T_i(\mathbf{f}) \quad i = 1, 2, \dots, n \quad (1.1)$$

where  $\mathbf{f}$  is the state of the system under observation,  $T_i$  is a mapping (possibly non-linear and usually noisy) which produces the  $i$ th data measurement  $g_i$ . We shall assume that our *a priori* knowledge of  $\mathbf{f}$  is expressed as an *a priori* PDF  $P[\mathbf{f}]$  over system states  $\mathbf{f}$  and that the properties of  $T_i$  ( $i = 1, \dots, n$ ) are expressed as a conditional transfer PDF  $P[\mathbf{g}|\mathbf{f}]$  over data states  $\mathbf{g}$ . Using Bayes' rule we may construct the *a posteriori* PDF  $P[\mathbf{f}|\mathbf{g}]$  as

$$P[\mathbf{f}|\mathbf{g}] = \frac{P[\mathbf{g}|\mathbf{f}] P[\mathbf{f}]}{P[\mathbf{g}]} \quad (1.2)$$

where  $P[\mathbf{g}]$  is the total probability of obtaining data  $\mathbf{g}$ . Bayes' rule thus provides a rigorous connection between the probabilistic (i.e. information theoretic) formulations of the 'direct' and 'inverse' problems [4]. The principal drawback to this method is that explicit results may be obtained only for the simplest types of PDF (e.g. gaussian).

However, we may lift this restriction by resorting to numerical methods and using the Metropolis algorithm [5]. A Markov chain of samples from a PDF  $P[\mathbf{x}]$  is generated as follows:

$$\mathbf{x}^{(k+1)} = S \mathbf{x}^{(k)} \quad (1.3)$$

where  $\mathbf{x}^{(k)}$  is the  $k$ th sample and  $S$  is a transition matrix. The probability of each possible transition is given by the appropriate element of a suitable stochastic matrix, i.e. one which generates a Markov chain of  $\mathbf{x}^{(k)}$  with a limiting distribution given by  $P[\mathbf{x}]$ . The Metropolis algorithm provides one specification for a suitable stochastic

matrix. The types of system for which the Metropolis algorithm is particularly appropriate are Markov random fields (MRFs), for which  $P[\mathbf{x}]$  has the special form [3]

$$P[\mathbf{x}] = \frac{1}{Z} \exp \left( - \sum_c U_c(\mathbf{x}) \right) \quad (1.4)$$

where  $U_c(\mathbf{x})$  is a potential which depends only on a subset of the components of  $\mathbf{x}$ , and  $Z$  is the partition function. If the current state is  $\mathbf{x} = \mathbf{x}_1$  and the possible next state is  $\mathbf{x} = \mathbf{x}_2$ , then the Metropolis algorithm will make a transition from  $\mathbf{x}_1$  to  $\mathbf{x}_2$  under the following conditions: if  $P[\mathbf{x}_2] > P[\mathbf{x}_1]$  then update  $\mathbf{x} \rightarrow \mathbf{x}_2$ , if  $P[\mathbf{x}_2] \leq P[\mathbf{x}_1]$  then update  $\mathbf{x} \rightarrow \mathbf{x}_2$  with probability  $\frac{P[\mathbf{x}_2]}{P[\mathbf{x}_1]}$ , otherwise leave  $\mathbf{x} = \mathbf{x}_1$ . The convenience of a MRF in this respect derives from the fact that all potentials  $U_c(\mathbf{x})$  which have the same value for  $\mathbf{x} = \mathbf{x}_1$  and  $\mathbf{x} = \mathbf{x}_2$  will cancel in the Metropolis update prescription.

The Metropolis algorithm has been used for solving inverse imaging problems using optimisation by simulated annealing (OSA) methods [2]. However, an issue which has not been examined in any detail is the performance of the Metropolis algorithm when  $\mathbf{x}$  is continuous valued; here the task of generating a suitably distributed random number in order to update  $\mathbf{x}$  can be computationally very expensive.

We can avoid such computational problems by updating not  $\mathbf{x}$  itself, but rather the components of a binary representation of  $\mathbf{x}$ . For this purpose we shall transform  $\mathbf{x}$  as

$$y_j \equiv y_j(\mathbf{x}) \quad 0 \leq y_j < 1 \quad (1.5)$$

in order to bring all the variables into the open interval  $[0, 1)$ . This entails a corresponding change of probability measure to

$$P'[\mathbf{y}] = \frac{\partial(\mathbf{x})}{\partial(\mathbf{y})} P[\mathbf{x}] \quad (1.6)$$

For our purposes the transformation in Equation 1.5 must be invertible in order that  $\mathbf{x}$  may be recovered from  $\mathbf{y}$ . Now define a binary representation of  $y_j$  as

$$y_j \equiv \sum_{r=1}^{\infty} 2^{-r} b_r^{(j)} \quad b_r^{(j)} \in \{0, 1\} \quad (1.7)$$

where clearly there is a unique binary decomposition for each possible value which  $y_j$  may take in the interval

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This letter appeared in *Inverse Problems*, 1986, vol. 2, no. 2, pp. L15-L17. Received 12 December 1985. © 1986 Controller, Her Majesty's Stationery Office, London.

$[0, 1)$ ; the  $b_r^{(j)}$  are then the ‘binary components’ of  $y_j$ . The corresponding transformation of measure is

$$\prod_j dy_j \longrightarrow \prod_j \prod_{r=1}^{\infty} db_r^{(j)} \quad (1.8)$$

where  $db_r^{(j)}$  can take only the values 0 and 1. This expresses the fact that a uniform distribution over the  $b_r^{(j)}$  corresponds to a uniform distribution over the  $y_j$ . The PDF  $P'[\mathbf{y}]$  may now be transformed into a PDF over the  $b_r^{(j)}$  by substituting the expansion in Equation 1.7 into  $P'[\mathbf{y}]$ . In practice the expansion in Equation 1.7 must be truncated, which leads to a representation of  $y_j$  in terms of a finite number of bits.

The Metropolis algorithm may then be applied directly to the constituent bits of  $\mathbf{y}$ , and  $\mathbf{x}$  may be recovered at each stage by inverting Equation 1.5. The decision about the number of bits to use (i.e. how many terms to retain in Equation 1.7) must depend upon the sensitivity of  $P'[\mathbf{y}]$  to small changes in the  $y_j$ ; clearly if we choose the transformation in Equation 1.5 intelligently we can reduce such sensitivity. The speed advantage of the binary components update scheme when compared with the conventional update scheme is  $\mathcal{O}\left(\frac{2^N}{N}\right)$  - for an  $N$ -bit representation. This factor arises because  $N$  2-state binary components  $b_r^{(j)}$  replace one  $2^N$ -state variable  $y_j$ .

Note that we have assumed that the conventional scheme does not have its speed already enhanced by some alternative technique, such as direct table search for generating the  $\mathbf{x}$ .

A caveat in all numerical work which involves sampling from a PDF by generating a Markov chain of samples is that we must beware of the fact that the samples are correlated. This does not affect our results provided that the Markov chain has a length which is many times its longest correlation time (measured in updates), and that we measure only quantities which are insensitive to the artificial Markovian correlations (i.e. physical quantities).

We anticipate that other related update algorithms may be discovered in due course. This is because all we have achieved is a decomposition of a (complicated) multivalued field into some elementary components - its constituent bits. There is every reason to suppose that other decompositions should be possible, and that some of these will follow naturally from fundamental analyses of what ‘drives’ multivalued field variables. An interesting example of a more general binary decomposition is the ‘Boltzmann Machine’ [1].

In summary, using the Metropolis algorithm to sample from  $P[\mathbf{x}]$  where  $\mathbf{x}$  is continuous is more efficient if a binary decomposition of  $\mathbf{x}$  is used.

- [1] D H Ackley, G E Hinton, and T J Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Science **9** (1985), no. 2, 147–169.
- [2] S Geman and D Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence **6** (1984), no. 6, 721–741.
- [3] R Kindermann and J L Snell, *Markov random fields and their applications*, Contemporary Mathematics, vol. 1, American Mathematical Society, Providence, 1980.
- [4] S P Luttrell, *The use of transinformation in the design of data sampling schemes for inverse problems*, Inverse Problems **1** (1985), no. 3, 199–218.
- [5] N Metropolis, A W Rosenbluth, M N Rosenbluth, A H Teller, and E Teller, *Equations of state calculations by fast computing machines*, Journal of Chemical Physics **21** (1953), no. 6, 1087–1092.

# The Use of Markov Random Field Models to Derive Sampling Schemes for Inverse Texture Problems \*

S P Luttrell

Royal Signals and Radar Establishment, St Andrews Road, Malvern, Worcs WR14 3PS, UK

We advocate the use of Markov random field (MRF) models to describe texture properties generally. For homogeneous textures we derive a sampling scheme that preserves the information content of the data whilst reducing their dimensionality considerably. We derive a refinement of this sampling scheme where residual redundancy is removed by a more careful selection of what is sampled. We relate our results to the grey level co-occurrence method of texture classification and to the pattern recognition device that is known as WISARD.

## I. INTRODUCTION

In this paper we shall be concerned with the modelling and sampling of textures. A comprehensive review of modern texture modelling techniques is given in reference [9] where the following loose definition of what constitutes a texture is given: ‘Texture could be defined as a structure composed of a large number of more or less ordered similar elements or patterns without one of these drawing special attention. So a global unitary impression is offered to the observer.’ We shall not attempt to refine this definition.

In practice, modelling and sampling are precursors to solving an inverse problem; the sample values are processed (i.e. inverted) within the context of the given model in order to extract information relating to the underlying source of the sample values. It is common to talk of ‘inverse source’ and ‘inverse scattering’ problems. We shall extend this terminology to include ‘inverse texture’ problems, in order to describe cases where the main aim of processing the sample values is to extract information relating to which texture (or class of textures) gave rise to the sample values.

The techniques which have been used to model textures divide into two broad classes: statistical and structural [9]. Statistical methods define texture properties according to a set of non-deterministic (or stochastic) rules, whereas structural methods define texture properties according to a set of deterministic rules or grammar (e.g. placement rules for elements of texture). Clearly hybrid schemes which embody both stochastic and deterministic aspects can be constructed.

We shall be concerned with statistical texture models in this paper; these can always be replaced by equivalent probability density function (PDF) models. All statistical texture models demand that images be analysed by using a set of statistical estimators chosen according to the form of the texture model. Frequently only the

estimators themselves are specified without any explicit mention of the underlying model. In this paper we shall adopt the viewpoint that it is highly desirable to use an underlying model explicitly when designing estimators.

In order to restrict the generality of the PDF that we might specify, we impose the restriction that it must factorise into the product of a number of ‘partial PDFs’, where each partial PDF would depend on only a limited number of the texture sample values. This is an exact representation when it can be argued that there is an independent underlying random variable generating the partial PDF of each such subset of sample values, for then the overall PDF is the product of all independent partial PDFs. This product representation is very flexible because we can refine any approximate model by, for instance, extending the dependences of the partial PDFs to include other sample values, or by including further partial PDF factors. This process could be continued to the extreme case where one subset is in fact the set of all sample values; thus the overall PDF could be represented exactly whatever its form. We envisage that such models would be built using iterated ‘hypothesis and test’ cycles, with the set of partial PDFs being steadily refined.

To achieve such a decomposition we shall use Markov random fields (MRF) to model textures [4–6]. The general MRF modelling framework is exceedingly flexible, and so many different types of texture may be modelled. Other advantages of the MRF modelling framework include its suitability for mapping onto either a general purpose computer or special purpose hardware for simulation studies [11, 16, 20], thus avoiding the need to calculate explicit results; frequently a very laborious process, if indeed it is at all possible.

In Section II we summarise the results from MRF theory that we shall need later on. Section III and Section IV contain the main results of this paper. In Section III we shall use Bayesian methods to derive an ideal sampling scheme for homogeneous textures (modelled as MRFs), and in Section IV we shall refine the results of Section III by removing sample redundancy. In Section V we shall comment upon the relationship of our results to the grey level co-occurrence method of texture classification [10], and also the relationship to the pattern recognition device that is known as WISARD [1].

Our contribution to MRF texture model theory is the

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This paper appeared in *Inverse Problems*, 1987, vol. 3, no. 2, pp. 289–300. Received 2 July 1986, in final form 1 September 1986. © 1987 Controller, Her Majesty’s Stationery Office, London.

sampling schemes that we have derived, plus the proofs of their information-preserving properties.

## II. MARKOV RANDOM FIELD MODELS

Let us denote a sample position by  $\mathbf{x}$  and the corresponding sample value by  $T(\mathbf{x})$ .  $T(\mathbf{x})$  is usually scalar and  $\mathbf{x}$  is usually permitted to take one of a discrete set of values (e.g. the pixel positions in an image), but neither of these assumptions is necessary in order to develop an MRF model. In particular, we shall often treat  $\mathbf{x}$  as if it were continuously variable. We shall denote the probability density function (PDF) of occurrence of a particular state  $T(\mathbf{x})$  of a texture by  $P[T(\mathbf{x})]$ . In practice  $T(\mathbf{x})$  is represented by using a finite number of bits per pixel, so  $P[T(\mathbf{x})]$  is in fact a probability (not a PDF). However, this distinction is not important for our purposes, so we shall use the language of PDFs.

Models have two distinct ingredients in general: variables and interactions amongst variables. The mathematical object that is used to represent such a structure is a ‘graph’ [3], and it consists of a set of sites or nodes (corresponding to variables) and a set of links or arcs (corresponding to interactions), each of which joins a pair of sites. The particular type of model that we shall build on such a graph is usually called a Markov random field (MRF) texture model [4–6].

Various treatments of MRFs on graphs may be found in the literature [2, 8, 12, 17], but for completeness we shall summarise the main results here. Let  $S \equiv \{i_1, i_2, \dots, i_n\}$  be a set of sites and  $G \equiv \{g_j, j \in S\}$  be a neighbourhood system for  $S$ , where  $g_j$  is the set of sites which are linked to site  $j$ : the corresponding graph is then denoted by  $\{S, G\}$ . A clique  $C$  of  $S$  is a subset of sites which are all mutually linked. Using texture model notation, a MRF model on a graph  $\{S, G\}$  is obtained when all the conditional PDFs relating to  $T(\mathbf{x})$  obey the constraint

$$\begin{aligned} P[T(\mathbf{x} = \mathbf{x}_j) | T(\mathbf{x} = \mathbf{x}_k), k \neq j] &= \\ P[T(\mathbf{x} = \mathbf{x}_j) | T(\mathbf{x} = \mathbf{x}_k), k \in g_j] \end{aligned} \quad (2.1)$$

In Equation 2.1 pixel  $\mathbf{x}_j$  of the data corresponds to site  $j$  of  $S$ , and pairs of interacting pixels correspond to sites of  $S$  that are neighbours in  $G$ , and the conditional PDFs in Equation 2.1 are called the local characteristics of the texture. We shall use the notation  $\mathbf{x}^{(C)}$  to denote those  $\mathbf{x}_j$  with  $j \in C$  and  $T(\mathbf{x}^{(C)})$  to denote the part of the whole texture state  $T(\mathbf{x})$  that occupies the pixels in  $C$ . We shall also assume that all proper subcliques of each clique  $C$  are excluded from the set of all cliques  $\{C\}$  of the graph  $\{S, G\}$ ; this has the effect of retaining in  $\{C\}$  only the largest cliques. This restriction on  $\{C\}$  will apply throughout this paper. Our choice of symbols to use for suffices is somewhat cavalier, but we are obliged to adopt such an approach in order not to overburden our notation with nested suffices.

The most convenient way to express  $P[T(\mathbf{x})]$  is as a Gibbs distribution. This relies upon using the

Hammersley-Clifford theorem [2, 12, 17] which may be stated as follows.

*Theorem.* Let  $G$  be a neighbourhood system.  $T(\mathbf{x})$  is a MRF with respect to  $G$  iff  $P[T(\mathbf{x})]$  is a Gibbs distribution with respect to  $G$ :

$$P[T(\mathbf{x})] \equiv \frac{1}{Z} \exp(-U[T(\mathbf{x})]) \quad (2.2)$$

where  $U[T(\mathbf{x})]$  is a potential given by

$$U[T(\mathbf{x})] \equiv \sum_C U_C [T(\mathbf{x}^{(C)})] \quad (2.3)$$

where the sum is over  $\{C\}$ ,  $U_C [T(\mathbf{x}^{(C)})]$  is a clique potential and  $Z$  is the normalisation factor (partition function) for the PDF. We make further comment below on the concept of a potential.

An alternative way of expressing the properties of a MRF is in terms of the set of local characteristics in Equation 2.1. However, this scheme is fraught with difficulties when the structure of the graph  $\{S, G\}$  is complicated, because  $\{S, G\}$  then enforces certain interrelationships amongst the local characteristics [2]. It is better to use the Gibbs distribution form because it automatically generates these interrelationships, though we pay the price for this convenience by specifying  $P[T(\mathbf{x})]$  in terms of a set of potentials that are only indirectly measurable.

In order to develop a physical intuition for these alternative ways of specifying a MRF, we shall describe a closely analogous situation which exists in statistical mechanics. Consider a system that consists of a number of interacting classical particles. The state probability of clusters of particles is measurable by an experimenter who operates directly upon the observable particles. A theorist will attempt to derive the joint state probability of all the particles as a Gibbs distribution constructed from the boundary conditions and a set of underlying particle interactions expressed as a Hamiltonian. Determining the form of the Hamiltonian from experimental observations involves solving an extremely difficult inverse problem; however, the economy of using it to capture the properties of a physical system more than repays the effort which is involved in deducing it in the first place. This analogy demonstrates the existence of a very strong precedent in the physical sciences for describing state probability distributions in terms of a set of underlying potentials rather than in terms of directly observable properties.

In order to complete the suite of alternative ways of expressing the properties of MRFs, we may write the Gibbs distribution in Equation 2.2 and Equation 2.3 as a product of probability factors. We shall denote the probability factor for clique  $C$  by  $p_C [T(\mathbf{x}^{(C)})]$ , where

$$p_C [T(\mathbf{x}^{(C)})] \equiv \exp[-U_C [T(\mathbf{x}^{(C)})]] \quad (2.4)$$

and

$$P[T(\mathbf{x})] \equiv \frac{1}{Z} \prod_C p_C [T(\mathbf{x}^{(C)})] \quad (2.5)$$

This decomposition of  $P[T(\mathbf{x})]$  in terms of the  $p_C[T(\mathbf{x}^{(C)})]$  makes it unnecessary to introduce the clique potentials  $U_C[T(\mathbf{x}^{(C)})]$  explicitly. For situations (such as texture modelling) where there may be no obvious physical interpretation of  $U_C[T(\mathbf{x}^{(C)})]$  this alternative, but equivalent, formulation is more satisfying. Loosely speaking each clique state  $T(\mathbf{x}^{(C)})$  is assigned a probability factor  $p_C[T(\mathbf{x}^{(C)})]$  (which is not directly measurable) which represents the probability measure assigned to that state.

We shall denote the (directly measurable) probability with which  $T(\mathbf{x}^{(C)})$  occurs when the interactions of all the other cliques  $C'$  ( $C' \neq C$ ) are taken into account by  $P_C[T(\mathbf{x}^{(C)})]$ . This is given by

$$P_C[T(\mathbf{x}^{(C)})] \equiv \int \left[ \prod_{j \in S-C} dT(\mathbf{x}_j) \right] P(T(\mathbf{x})) \quad (2.6)$$

This may be rearranged to give

$$\begin{aligned} P_C[T(\mathbf{x}^{(C)})] &= \\ \frac{1}{Z} p_C[T(\mathbf{x}^{(C)})] \int \left[ \prod_{j \in S-C} dT(\mathbf{x}_j) \right] \prod_{C' \neq C} p_{C'}[T(\mathbf{x}^{(C')})] &\quad (2.7) \end{aligned}$$

Of course  $P_C[T(\mathbf{x}^{(C)})] \neq p_C[T(\mathbf{x}^{(C)})]$  because the ‘bare’ interactions within clique  $C$  are ‘dressed’ by interactions with the rest of the cliques as shown in Equation 2.7.

There are other very good reasons for using MRF models. The statistical properties of a texture may then be investigated either numerically by using the Metropolis algorithm [16], or theoretically by using cluster expansion techniques [21]. A MRF model may also be mapped directly into a special purpose computer designed specifically to run the Metropolis algorithm very rapidly [11]. A derivation of the Metropolis algorithm and some associated results on convergence are derived in the appendix of reference [8].

The use of the Metropolis algorithm to generate a sequence of realisations of  $T(\mathbf{x})$  is summarised as follows. The aim of the algorithm is to generate a sequence of realisations of  $T(\mathbf{x})$  by progressively altering  $T(\mathbf{x})$  so as to move about the space of all  $T(\mathbf{x})$  in such a way that the frequency with which a particular  $T(\mathbf{x})$  occurs is proportional to  $P[T(\mathbf{x})]$ . Let the current state be  $T(\mathbf{x}) = T_1(\mathbf{x})$ , and the possible next state be  $T(\mathbf{x}) = T_2(\mathbf{x})$ . The Metropolis algorithm states that a transition from  $T_1(\mathbf{x})$  to  $T_2(\mathbf{x})$  is made under the following conditions: if  $P[T_2(\mathbf{x})] > P[T_1(\mathbf{x})]$  then update  $T(\mathbf{x}) \rightarrow T_2(\mathbf{x})$ , if  $P[T_2(\mathbf{x})] \leq P[T_1(\mathbf{x})]$  then update  $T(\mathbf{x}) \rightarrow T_2(\mathbf{x})$  with probability  $\frac{P[T_2(\mathbf{x})]}{P[T_1(\mathbf{x})]}$ , otherwise leave  $T(\mathbf{x}) = T_1(\mathbf{x})$ . This algorithm will generate a sequence (Markov chain) of realisations of  $T(\mathbf{x})$  which is easily proved [16] to sample each  $T(\mathbf{x})$  with a probability that is proportional to  $P[T(\mathbf{x})]$ , as required. The

main caveat in using this algorithm to investigate the statistical properties of  $T(\mathbf{x})$  (as defined by  $P[T(\mathbf{x})]$ ) is that a single Metropolis update of  $T(\mathbf{x})$  does not provide a new and independent realisation of  $T(\mathbf{x})$ . Thus statistical averages do not have error bars that decrease like  $(N)^{\frac{1}{2}}$  ( $N$  = number of updates), rather they decrease like  $(\frac{N}{c})^{\frac{1}{2}}$  where  $c$  is the correlation time (measured in updates) of the sequence of  $T(\mathbf{x})$  generated by the Metropolis algorithm.

### III. SAMPLING FOR SUFFICIENT STATISTICS

We shall now derive an efficient scheme for sampling texture data. Firstly we shall summarise the Bayesian analysis of data derived from one of  $b$  texture classes  $t_j$  ( $j = 1, \dots, b$ ). Define the *a priori* probability that  $t_j$  exists as  $q[t_j]$ , and define the PDF that data  $T(\mathbf{x})$  are then derived from texture  $t_j$  as  $P[T(\mathbf{x})|t_j]$ . Then the *a posteriori* probability is

$$P[t_j|T(\mathbf{x})] = \frac{P[T(\mathbf{x})|t_j] q[t_j]}{P[T(\mathbf{x})]} \quad (3.1)$$

where we have used Bayes’ rule, and  $P[T(\mathbf{x})]$  is given by

$$P[T(\mathbf{x})] = \sum_{j=1}^b P[T(\mathbf{x})|t_j] q[t_j] \quad (3.2)$$

It is important to realise that the information (as defined by Shannon [18, 19]) that is encoded by  $P[t_j|T(\mathbf{x})]$  contains both prior knowledge (the  $q[t_j]$ ) and data constraint information (the  $P[T(\mathbf{x})|t_j]$ ), and that nothing more can be said about  $P[t_j|T(\mathbf{x})]$  [13–15]. No processing of  $T(\mathbf{x})$  can increase the *a posteriori* information; one can at best preserve the information, and usually one destroys information during processing.

In practice, texture classification does not proceed along the ideal lines described above because of the large data set size ( $10^3$  to  $10^6$  pixels is currently fashionable). In fact a vector of statistics  $\mathbf{N}$  is first extracted from the data, and then classification is attempted using  $\mathbf{N}$  alone. Ideally  $\mathbf{N}$  captures all the information-bearing statistical structure of the data  $T(\mathbf{x})$ , in which case the *a posteriori* distribution  $P[t_j|\mathbf{N}]$  is identical to  $P[t_j|T(\mathbf{x})]$ . Such an ideal set of statistics is referred to as a ‘sufficient’ set of statistics [7]. The first step in a good data analysis is the design of a powerful set of statistics for data reduction, so that the information of interest in  $T(\mathbf{x})$  is preserved.

In the case where  $P[T(\mathbf{x})|t_j]$  is modelled as a MRF (using a Gibbs distribution) we shall derive an  $\mathbf{N}$  that is a sufficient set of statistics. Define the graph  $\{S, G\}$  as follows:

$$S \equiv S_1 = S_2 = S_3 = \dots S_b \quad (3.3)$$

$$G \equiv G_1 \cup G_2 \cup G_3 \cup \dots G_b \quad (3.4)$$

where we assume that the sites  $S_j$  for each texture  $t_j$  are the same, and the union of the neighbourhoods is taken

so that all the textures' statistical properties may be described by the same graph structure. Using Equation 2.2

$$P[T(\mathbf{x})|t_j] = \frac{1}{Z} \exp \left\{ -U^{(j)}[T(\mathbf{x})] \right\} \quad (3.5)$$

where

$$U^{(j)}[T(\mathbf{x})] \equiv \sum_C U_C^{(j)} [T(\mathbf{x}^{(C)})] \quad (3.6)$$

where it is understood that all proper subcliques have been removed from  $\{C\}$ .

We shall now simplify the notation by replacing functions by vectors as follows:

$$U_C^{(j)} [T(\mathbf{x}^{(C)})] \longrightarrow \mathbf{G}_C^{(j)} \cdot \mathbf{n}_C \quad (3.7)$$

This replacement is justified by noting that  $U_C^{(j)}[T(\mathbf{x}^{(C)})]$  is a mapping from the state  $T(\mathbf{x}^{(C)})$  to the potential  $U_C^{(j)}$ . Such a mapping can be broken down into two simple stages as follows. Firstly map  $T(\mathbf{x}^{(C)})$  onto a standard notation, chosen to be one of the set of unit vectors represented by  $(0, 0, 0, \dots, 0, 1, 0, \dots, 0, 0, 0)$ . We arrange that the number of unit vectors is equal to the number of states that  $T(\mathbf{x}^{(C)})$  may occupy. Secondly form the scalar product of this unit vector and a previously defined vector  $\mathbf{G}_C^{(j)}$ , consisting of components equal to the corresponding potentials  $U_C^{(j)}$ . Together these two stages of mapping yield the overall mapping  $T(\mathbf{x}^{(C)}) \longrightarrow U_C^{(j)}$ , whilst at the same time reducing the notational burden by replacing the function notation by a vector notation. Equation 3.5 can now be written as

$$P[T(\mathbf{x})|t_j] = \frac{1}{Z_j} \exp \left( - \sum_C \mathbf{G}_C^{(j)} \cdot \mathbf{n}_C \right) \quad (3.8)$$

In order to make further progress we shall now introduce the symmetries of the textures  $t_j$  which identify which  $\mathbf{G}_C^{(j)}$  are equal for a given  $t_j$ . We shall assume here that the statistical properties of all the textures are translation invariant (i.e. the textures are homogeneous), so the clique structure is repeated at each pixel position. We shall denote the translation operator (for translating site labels) generically by the symbol  $L$ , so  $\{C\}$  can be symbolically decomposed thus:

$$\{C\} = \{L\{P\} | \text{all } L \text{ in the symmetry group}\} \quad (3.9)$$

where  $\{P\}$  is the set of cliques (with proper subcliques removed) that is associated with a single pixel. The set  $\{P\}$  is the largest set of cliques that can be constructed such that  $LC_j \neq C_k$  is true for all pairs of cliques  $C_j$  and  $C_k$  in  $\{P\}$  and for all  $L$  in the symmetry group. The set  $\{L\{P\} | \text{all } L \text{ in the symmetry group}\}$  is then clearly equal to  $\{C\}$  by construction.  $P[T(\mathbf{x})|t_j]$  can therefore be written as

$$P[T(\mathbf{x})|t_j] = \frac{1}{Z_j} \exp \left( - \sum_P \mathbf{G}_P^{(j)} \cdot \mathbf{N}_P \right) \quad (3.10)$$

where

$$\mathbf{N}_P \equiv \sum_{C=L\{P\}, \text{ all } L \text{ in the symmetry group}} \mathbf{n}_C \quad (3.11)$$

$\mathbf{N}_P$  gathers together all those  $\mathbf{n}_C$  that are related by a translation  $L$ , and so the components of  $\mathbf{N}_P$  effectively define a 'histogram' (or frequency of occurrence) of each clique state in  $\{P\}$  in the homogeneous texture. Clearly this reduction can be performed in an analogous way for any other symmetry.

The notation can be simplified further by incorporating the summation in Equation 3.10 into the inner product notation thus:

$$\mathbf{G}^j \equiv (G_1^{(j)}, G_2^{(j)}, \dots) \quad (3.12)$$

and

$$\mathbf{N} \equiv (\mathbf{N}_1, \mathbf{N}_2, \dots) \quad (3.13)$$

where the cliques in  $\{P\}$  have been arbitrarily indexed  $1, 2, 3, \dots$ . Then

$$\mathbf{G}^j \cdot \mathbf{N} = \sum_P \mathbf{G}_P^{(j)} \cdot \mathbf{N}_P \quad (3.14)$$

so that

$$P[T(\mathbf{x})|t_j] = \frac{1}{Z_j} \exp (-\mathbf{G}^j \cdot \mathbf{N}) \quad (3.15)$$

The PDF over  $\mathbf{N}$  is then given by

$$P[\mathbf{N}|t_j] = \int [dT(\mathbf{x})] P[T(\mathbf{x})|t_j] \delta[\mathbf{N}[T(\mathbf{x})] - \mathbf{N}] \quad (3.16)$$

where  $\mathbf{N}[T(\mathbf{x})]$  is the function that performs the overall mapping  $T(\mathbf{x}) \longrightarrow \mathbf{N}$ . The way in which we have used the  $\delta$ -function notation is a special case of the identity

$$P[a(\mathbf{z}) = \alpha] \equiv \int [d\mathbf{z}] P[\mathbf{z}] \delta[a(\mathbf{z}) - \alpha] \quad (3.17)$$

Because the  $T(\mathbf{x})$  dependence of  $P[T(\mathbf{x})|t_j]$  is determined solely by  $\mathbf{x}[T(\mathbf{x})]$ , which is held constant by the  $\delta$ -function in Equation 3.16, we can perform the integration yielding

$$\begin{aligned} P[\mathbf{N}|t_j] &= Z_0[\mathbf{N}] P[T(\mathbf{x})|t_j] \\ &= \frac{Z_0[\mathbf{N}]}{Z_j} \exp (-\mathbf{G}^j \cdot \mathbf{N}) \end{aligned} \quad (3.18)$$

where  $Z_0[\mathbf{N}]$  is the volume of integration (which does not depend on  $t_j$ ). In the trivial case where  $P[T(\mathbf{x})|t_j]$  is constant (which occurs when class  $t_j$  is uniformly distributed random noise) then  $P[\mathbf{N}|t_j]$  is simply proportional to  $Z_0[\mathbf{N}]$ .

The result in Equation 3.18 proves that  $\mathbf{N}$  is a sufficient set of statistics because, apart from a known texture-independent factor  $Z_0[\mathbf{N}]$ , the PDF  $P[\mathbf{N}|t_j]$  is

the same as  $P[T(\mathbf{x})|t_j]$ . Although the inverse mapping  $\mathbf{N} \rightarrow T(\mathbf{x})$  cannot be performed uniquely, the inverse mapping  $P[\mathbf{N}|t_j] \rightarrow P[T(\mathbf{x})|t_j]$  is unique, so the mapping  $T(\mathbf{x}) \rightarrow \mathbf{N}$  preserves all information concerning the  $t_j$  that is contained in  $T(\mathbf{x})$  (see Equation 3.1). Using the language of references [13–15] transinformation (or mutual information) is conserved under the mapping  $T(\mathbf{x}) \rightarrow \mathbf{N}$ ; this is a necessary property for  $\mathbf{N}$  to be a sufficient set of statistics.

It is important to recognise what conditions must be satisfied in order that Equation 3.18 be valid; they are given in Equation 3.3 and Equation 3.4. The most restrictive condition is Equation 3.4 because it requires that all the neighbourhoods  $G_j$  be known in advance. Even if both these conditions are met in full the result in Equation 3.18 is not of any practical use unless the dimensionality of  $\mathbf{N}$  can be reduced by introducing symmetries as in Equation 3.9. The translational symmetry that we used is very widely applicable, and should not restrict the utility of Equation 3.18 much.

In practice, we might not know the  $G_j$  in advance of processing the data. In such cases the above prescription for dimensionality reduction cannot be implemented directly. However, we may instead build up neighbourhood structures *ab initio* by using selfconsistency. Thus for a particular  $G_j$  we measure  $P[\mathbf{N}|t_j]$  and we hypothesise that  $\mathbf{N}$  comprises a sufficient set of statistics. Using Equation 3.18 this gives  $P[T(\mathbf{x})|t_j]$ , so that synthetic realisations of texture  $t_j$  may accordingly be generated by using the Metropolis algorithm. The hypothesis that  $\mathbf{N}$  is a sufficient set of statistics may now be checked by comparing further (as yet unmeasured) statistics derived both from the original image and from the synthetic image. Where disagreement is found, corresponding cliques may be introduced to enlarge  $G_j$ . We have only touched upon the possibilities for automatic generation and verification of  $G_j$  as the correct neighbourhood structure for texture  $t_j$ ; a detailed account will be presented in a future publication.

#### IV. ELIMINATION OF SAMPLE REDUNDANCY

In Section III we showed that the clique state occupancy  $\mathbf{N}$  is a sufficient set of statistics for retaining all the

information in  $T(\mathbf{x})$  relevant to the problem of texture classification. In this section we shall derive a method of reducing the dimensionality of  $\mathbf{N}$  whilst still retaining all of this information. This procedure eliminates all residual sample redundancy.

Recall that ultimately  $\mathbf{N}$  derives from the  $\mathbf{n}_C$  in Equation 3.7. The  $\mathbf{n}_C$  are individual clique state occupancies which are summed together in equivalence classes in Equation 3.11; this is the dimension-reducing step in the overall transformation  $T(\mathbf{x}) \rightarrow \mathbf{N}$ . Further dimension reduction turns out to be possible because cliques are not only related by symmetry (to form equivalence classes), but they are also related because of the possibility that  $C_j \cap C_k \neq \emptyset (j \neq k)$ , i.e. some cliques overlap each other. Symmetry enabled us to sum the  $\mathbf{n}_C$  as shown in Equation 3.11, whilst clique overlap will enable us to interrelate the states  $\mathbf{n}_C$  of different cliques. In order to evaluate the effect of clique overlap systematically it is necessary to decompose each clique into subcliques as we shall show below.

The clique state occupation vector  $\mathbf{n}_C$  that we introduced in Equation 3.7 has the general form  $(0, 0, \dots, 0, 1, 0, \dots, 0, 0)$ , where only one component is non-zero. This notation was adequate for the purposes of Section III, but we need to use a different representation in order to display clique state interdependencies (i.e. redundancy) explicitly. Clearly cliques overlap only partially, so a notation in which clique states are decomposed into partial states would assist the analysis.

We shall present the new representation without proof, and demonstrate that it has the required properties afterwards. Define the state  $T(\mathbf{x}^{(C)})$  of an  $m$ -point clique  $C$  as follows:

$$T(\mathbf{x}^{(C)}) \equiv (T_1^{(C)}, T_2^{(C)}, \dots, T_m^{(C)}) \quad (4.1)$$

where we have arbitrarily chosen to label the clique sites as  $1, 2, 3, \dots, m$ . The components  $T_j^{(C)}$  are defined as

$$T_j^{(C)} \in X \equiv \{0, 1, 2, \dots, L - 1\} \quad (4.2)$$

Thus each ‘pixel’ can take one of  $L$  states which are arbitrarily labelled as shown in Equation 4.2. Now define a transformation of  $T(\mathbf{x}^{(C)})$  into state occupancy as follows:

$$\begin{aligned}
& \text{if } (T(\mathbf{x}_j) = r \neq 0) \\
& \quad \text{then } n_{j,r}^{(1)} = 1 \quad (1 \leq j \leq m, \\
& \quad \text{else } n_{j,r}^{(1)} = 0 \quad (1 \leq r \leq L-1) \\
& \text{if } (T(\mathbf{x}_j) = r \neq 0) \text{ and } (T(\mathbf{x}_k) = s \neq 0) \\
& \quad \text{then } n_{j,k,r,s}^{(2)} = 1 \quad (1 \leq j < k \leq m, \\
& \quad \text{else } n_{j,k,r,s}^{(2)} = 0 \quad (1 \leq r \leq L-1, 1 \leq s \leq L-1) \\
& \text{if } (T(\mathbf{x}_j) = r \neq 0) \text{ and } (T(\mathbf{x}_k) = s \neq 0) \text{ and } (T(\mathbf{x}_l) = t \neq 0) \\
& \quad \text{then } n_{j,k,l,r,s,t}^{(3)} = 1 \quad (1 \leq j < k < l \leq m, \\
& \quad \text{else } n_{j,k,l,r,s,t}^{(3)} = 0 \quad (1 \leq r \leq L-1, 1 \leq s \leq L-1, 1 \leq t \leq L-1)
\end{aligned} \tag{4.3}$$

etc, where we have omitted the clique suffix  $C$  for clarity. The elements of each  $n^{(p)}$  are a set of matrix elements that record the state occupancies of the  $p$ -point subcliques of clique  $C$ . The total number of matrix elements is easily verified to be  $L^m - 1$ , where the missing state is  $\mathbf{x}^{(C)} = \mathbf{0}$  which need not be explicitly recorded in the state occupancy representation because its presence can be inferred when all the matrix elements in Equation 4.3 are zero. Clearly this state could also have been omitted in the representation that was used in Section III. We observe in passing that it is not important which element of  $X$  in Equation 4.2 is selected as the ‘odd one out’ in Equation 4.3, nor even that it is the same choice for different sites of the graph. The clique occupancy vector  $\mathbf{n}_C$  will now be replaced with its new representation thus:

$$\mathbf{n}_C \rightarrow \mathbf{n}'_C \equiv \left\{ \mathbf{n}_C^{(1)}, \mathbf{n}_C^{(2)}, \dots, \mathbf{n}_C^{(m)} \right\} \tag{4.4}$$

where  $\mathbf{n}_C^{(p)}$  represents (as a vector) the matrix elements corresponding to the  $p$ -point state occupancies ( $1 \leq p \leq m$ ) as defined in Equation 4.3 of an  $m$ -point clique  $C$ .

$\mathbf{n}'_C$  could now be used in place of  $\mathbf{n}_C$  in the derivation of Section III from Equation 3.7 onwards. However we must first verify that Equation 4.3 is invertible so that the clique state may be recovered from  $\mathbf{n}'_C$ . This will ensure that we can construct an arbitrary potential  $U_C[T(\mathbf{x}^{(C)})]$  by using an expression of the form  $\mathbf{G}_C^{(j)} \cdot \mathbf{n}'_C$ , as required. A simple test for invertibility is to check that the matrix made up row-wise of all the possible  $\mathbf{n}'_C$  vectors is full rank. For the case of the  $\mathbf{n}_C$  vectors of Section III this is trivial because such a matrix is the identity matrix (or a row and column permutation thereof). It is simplest to give an example to show how the  $\mathbf{n}'_C$  case is checked. Thus let  $m = 3$  in Equation 4.1 and  $L = 2$  in Equation 4.2, then the matrix is:

	1**	*1*	***1	*11	1*1	11*	111
100	1	0	0	0	0	0	0
010	0	1	0	0	0	0	0
001	0	0	1	0	0	0	0
011	0	1	1	1	0	0	0
101	1	0	1	0	1	0	0
110	1	1	0	0	0	1	0
111	1	1	1	1	1	1	1

where all the seven ( $= 2^3 - 1$ ) states are listed at the left-hand side, the columns are correspondingly labelled according to the value(s) of  $r, s, t \dots$  in Equation 4.3 and the asterisks denote a ‘don’t care’ state. We have outlined the block diagonal structure because it may be used to eliminate all the off-diagonal elements; this proves that the matrix is full rank. Such a matrix structure is preserved for all  $m$  and  $L$ , so in general Equation 4.3 provides an invertible representation.

We have shown that  $\mathbf{n}'_C$  will lead to a sufficient set of statistics (just as  $\mathbf{n}_C$  did in Section III). Now the task is to use the subclique structure, of the new representation to eliminate redundancy. For the special case of a pair of overlapping cliques  $C$  (with  $m$  points) and  $D$  (with  $n$  points) with states denoted by  $\mathbf{x}^{(C)}$  and  $\mathbf{x}^{(D)}$  respectively, denote the occupancy vectors by

$$\begin{aligned}
\mathbf{n}'_C &\equiv \left( \mathbf{n}_C^{(1)}, \mathbf{n}_C^{(2)}, \dots, \mathbf{n}_C^{(m)} \right) \\
\mathbf{n}'_D &\equiv \left( \mathbf{n}_D^{(1)}, \mathbf{n}_D^{(2)}, \dots, \mathbf{n}_D^{(n)} \right)
\end{aligned} \tag{4.5}$$

$C \cap D \neq \emptyset$ , so some component vectors of  $\mathbf{n}'_C$  and  $\mathbf{n}'_D$  are common to both cliques although they occupy different positions in the overall vectors  $\mathbf{n}'_C$  and  $\mathbf{n}'_D$ . Such duplication of component vectors will lead to redundancy in the components of the analogue of  $\mathbf{N}_P$  in Equation 3.11. Note that the source of this redundancy is not at all obvious in the representation that was used in Section III. Sample redundancy may now be removed by ensuring that each component vector in Equation 4.4 is allowed to appear in only a single  $\mathbf{n}'_C$ . There is no other source of redundancy (due to clique overlap), so the resultant state occupation representation will be termed ‘minimal’.

A simple prescription that leads to a minimal representation is given by:

1. identify all cliques and all proper subcliques (i.e. all the possible cliques of the graph  $\{S, G\}$ );

2. enumerate the state of each such  $p$ -point clique according to the prescription:

---


$$\begin{aligned} & \text{if } (T(\mathbf{x}_j) = r \neq 0) \text{ and } (T(\mathbf{x}_k) = s \neq 0) \text{ and } (T(\mathbf{x}_l) = t \neq 0) \text{ and } \dots \\ & \text{then } n_{j,k,l,\dots,r,s,t,\dots}^{(p)} = 1 \quad \left( \begin{array}{c} 1 \leq j < k < l < \dots \leq m, \\ 1 \leq r \leq L-1, 1 \leq s \leq L-1, 1 \leq t \leq L-1, \dots \end{array} \right) \\ & \text{else } n_{j,k,l,\dots,r,s,t,\dots}^{(p)} = 0 \end{aligned}$$


---

This provides a complete set of  $\mathbf{n}^{(p)}$  vectors (as in Equation 4.4) with no redundancy.

Symmetries may be introduced as in Section III by summing together the  $\mathbf{n}^{(p)}$  vectors that fall into equivalence classes under the symmetries. This produces the analogue of the  $\mathbf{N}_P$  in Equation 3.11 calculated in the new state occupancy representation, and therefore is devoid of redundancy. The remainder of the derivation following Equation 3.11 may be carried through with no changes. The new representation leads to a set of statistics that may be termed ‘necessary and sufficient statistics’, because although there may be other minimal state occupation representations there is certainly no further redundancy that can be removed from our scheme (hence the term ‘necessary’). This sampling scheme is to homogeneous MRFs as Nyquist sampling is to band-limited signals [13, 14].

## V. APPLICATIONS

Both the ‘sufficient’ and the ‘necessary and sufficient’ statistics results that we obtained in Section III and Section IV respectively are widely applicable. They are of most use where the data set is a large array of data (e.g. an image) that consists of a few large homogeneous segments, for then the greatest advantage comes from dimension reduction due to translational symmetry. Such data sets are typically obtained from remote sensing equipment.

A special case of the sampling scheme that we have developed has appeared in the literature under a different guise; it has been called the grey level co-occurrence matrix method (GLCM) [9, 10]. In terms of our notation the GLCM is the sampling scheme that would be obtained using the method of Section III if the following conditions were met:

1. the texture is homogeneous
2. only two-pixel cliques are required.

In the GLCM two-pixel clique state occupancies are measured (as in Section III) for various choices of pixel sepa-

Table I:  $D_{\text{red}}$  and  $D_{\min}$ .

$L$	$D_{\text{red}}$	$D_{\min}$
2	8	5
3	36	18
4	64	39

ration. Because the underlying clique structure is not actually known the choice of two-pixel cliques is somewhat arbitrary, and is therefore not guaranteed to provide a sufficient set of statistics. Usually co-occurrence matrices that involve neighbouring pixels (or near neighbours) only are computed in the GLCM, so any long range two-pixel clique structure is ignored.

Furthermore we can categorically state that the set of statistics that is measured by the GLCM is redundant because the effects of clique overlap have been ignored. The extent of this redundancy is easy to check. For instance, when a nearest-neighbour scheme is used, the clique structure (with all proper subcliques removed) consists of four different types of two-point cliques: north to south, east to west, north-east to south-west and north-west to south-east. For pixels with  $L$  states (see Equation 4.2) the dimensionality  $D_{\text{red}}$  of  $\mathbf{N}$  (see Equation 3.11) is  $4L^2$  using the redundant scheme of Section III which is equivalent to the GLCM. However, using the minimal scheme of Section IV the dimensionality  $D_{\min}$  is  $4L^2 - 7L + 3$  ( $(L-1)$  from one-point cliques and  $4(L-1)^2$  from two-point cliques). Table I compares these dimensionalities.

Here, the advantages of the minimal scheme are obvious for small values of  $L$ , although  $D_{\min} \sim D_{\text{red}}$  for large  $L$ .

In addition to the GLCM the sampling scheme that is used in WISARD [1] is also related to our results. In the extreme case where no prior knowledge of the graph structure  $\{S, G\}$  is available and also there are no symmetries to assist the data reduction, one has no choice but to adopt a non-committal random sampling scheme. The clique size should be as large as possible within the limitations of the available memory in order to minimise the

possible information loss, but at the same time the clique members should be chosen at random. This scheme has had a significant degree of success in practical pattern recognition problems.

Clearly the GLCM and WISARD lie at two extremes: the GLCM is (currently) concerned with short-range statistics but WISARD is (in the case described above) concerned with long-range statistics on average. Our results show how these two sampling schemes may be unified within a single coherent mathematical framework.

An immediate application of our sampling results would be to make rigorous the use of the GLCM method in image analysis. Furthermore we have indicated how the method could be extended by use of a ‘hypothesis and test’ technique whereby statistical differences between the original image and a synthetically generated image allow the structure of the underlying model to be systematically built up.

## VI. CONCLUSIONS

Expressing stochastic models as MRFs on graphs leads to a systematic expansion of the PDF  $P[T(\mathbf{x})]$  in terms of clique factors. We used this decomposition together with symmetries of the model to derive a sampling scheme that measures sufficient statistics of the MRF, i.e. all information is preserved whilst the dimensionality of the data is reduced. The relevant statistics are the clique state occupation frequencies (where all proper subcliques have been removed), measured over equivalence classes of cliques generated by the symmetries (see Equation 3.7 and com-

ments thereafter). We observed that these statistics are in general redundant, so that some further dimensionality reduction is possible. We derived a minimal (necessary and sufficient) set of statistics that is to symmetric MRFs as Nyquist sampling is to band-limited signals. The minimal statistics are obtained by including all proper subcliques as well in the set of cliques, and at the same time ignoring any (sub)clique state that contains the state 0 at any site (see Equation 4.3 and comments thereafter). Symmetries are then used as above.

These results provide a theoretical underpinning for the grey level co-occurrence matrix method (GLCM) of data reduction and the sampling scheme that is used in the WISARD pattern recognition device. It is satisfying that a novel information theoretic result generalises pre-existing sampling techniques.

The greatest benefit of our results is an understanding of where the statistical information in a symmetric MRF resides. A corollary of this is that our set of statistics is all that needs to be measured in order to determine the underlying potentials of the MRF, because there is no other information present to be measured. Furthermore, the number of independent parameters that are required to describe all the potentials is equal to the dimensionality of our minimal statistic representation. If a greater number of parameters is used then only a subspace of the relevant dimension will be ‘visible’ in the data statistics, and the remaining linear combination will remain ‘invisible’. This is a simple demonstration that the inverse problem of determining the potentials (or the Hamiltonian) by observing sufficient statistics is ill-conditioned.

- 
- [1] I Aleksander and T J Stonham, *Guide to pattern recognition using random access memories*, Computers and Digital Techniques **2** (1979), no. 1, 29–40.
  - [2] J Besag, *Spatial interaction and the statistical analysis of lattice systems*, Journal of the Royal Statistical Society: Series B **36** (1974), no. 2, 192–236.
  - [3] B Carré, *Graphs and networks*, Clarendon, Oxford, 1979.
  - [4] R Chellappa and S Chatterjee, *Classification of textures using Gaussian Markov random fields*, IEEE Transactions on Acoustics, Speech, and Signal Processing **33** (1985), no. 4, 959–963.
  - [5] R Chellappa, S Chatterjee, and R Bagdazian, *Texture synthesis and coding using Gaussian Markov random field models*, IEEE Transactions on Systems, Man, and Cybernetics **15** (1985), no. 2, 298–303.
  - [6] G R Cross and A K Jain, *Markov random field texture models*, IEEE Transactions on Pattern Analysis and Machine Intelligence **5** (1983), no. 1, 25–39.
  - [7] M H DeGroot, *Optimal statistical decisions*, McGraw-Hill, New York, 1970.
  - [8] S Geman and D Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence **6** (1984), no. 6, 721–741.
  - [9] L Van Gool, P Dewaele, and A Oosterlinck, *Texture analysis anno 1983*, Computer Vision, Graphics, and Image Processing **29** (1985), no. 3, 336–357.
  - [10] R M Haralick, K Shanmugam, and I Dinstein, *Textural features for image classification*, IEEE Transactions on Systems, Man, and Cybernetics **3** (1973), no. 6, 610–621.
  - [11] H J Hilhorst, A F Bakker, C Bruin, A Compagner, and A Hoogland, *Special purpose computers in physics*, Journal of Statistical Physics **34** (1984), no. 5–6, 987–999.
  - [12] R Kindermann and J L Snell, *Markov random fields and their applications*, Contemporary Mathematics, vol. 1, American Mathematical Society, Providence, 1980.
  - [13] S P Luttrell, *A new method of sample optimisation*, Optica Acta **32** (1985), no. 3, 255–257.
  - [14] ———, *The use of transinformation in the design of data sampling schemes for inverse problems*, Inverse Problems **1** (1985), no. 3, 199–218.
  - [15] S P Luttrell and C J Oliver, *Prior knowledge in synthetic aperture radar processing*, Journal of Physics D: Applied Physics **19** (1986), no. 3, 333–356.
  - [16] N Metropolis, A W Rosenbluth, M N Rosenbluth, A H Teller, and E Teller, *Equations of state calculations by fast computing machines*, Journal of Chemical Physics **21** (1953), no. 6, 1087–1092.

- [17] C J Preston, *Gibbs states on countable sets*, University Press, Cambridge, 1974.
- [18] C E Shannon, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 3, 379–423.
- [19] ———, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 6, 623–656.
- [20] S Wolfram, *Special section on computing at the frontiers of theoretical physics*, Communications of the ACM **28** (1985), no. 4, 352–353.
- [21] M Wortis, *Phase transitions and critical phenomena*, vol. 3, ch. Linked cluster expansion, pp. 113–180, Academic, New York, 1974.



# The Role of Prior Knowledge in Coherent Image Processing \*

S. P. LUTTRELL and C. J. OLIVER

Royal Signals and Radar Establishment, St Andrews Road, Great Malvern, Worcester WR14 3PS, U.K.

Models that encode prior knowledge about a scene provide a means for interpreting image data from that scene in more detail than would otherwise be so. Information about both background clutter and target characteristics should be included in this prior knowledge. We demonstrate the use of a generalised noise model to represent a variety of naturally occurring random terrain clutter textures observed in high-resolution synthetic aperture radar (SAR) images. In addition a similar approach is adopted for the simulation of such textures. Having established the background properties we next introduce prior knowledge about any target within the scene and exploit this in achieving a cross-section reconstruction having improved resolution compared with the original image. Examples of such a super-resolution method based on singular value decomposition are demonstrated and the limits of the technique are indicated.

## I. INTRODUCTION

In this paper we shall be concerned with aspects of the interpretation of synthetic aperture radar (SAR) images. In common with all image interpretation processes we shall imbue the raw-image data with some ‘meaning’ based on relevant past experience in a similar context. This ‘prior knowledge’ is cast in the form of a model that underlies our image interpretation.

In many radar applications we are concerned with detecting the presence of ‘targets’ against background ‘clutter’. This statement assumes more information about the scene beyond that found in the image alone. We have introduced prior knowledge that (typically) targets are small bright blobs embedded in a weak clutter background. To detect the presence of such targets we must develop an understanding of the properties of both targets and clutter. In this paper we describe techniques to increase our understanding of both classes by the application of prior knowledge.

## II. CLUTTER MODELS AS PRIOR KNOWLEDGE

In Figure 1 we show a SAR image of a region of fields and woods, obtained with the RSRE X-band airborne system. It is apparent that the intensity in the image shows strong speckle behaviour. This is a well-known phenomenon in coherent imaging and arises where the total intensity is made up by the interference between large numbers of randomly positioned scatterers within each resolution cell. The presence of this speckle over the entire image suggests that a physical model for scattering and imaging of the surface might describe each resolution

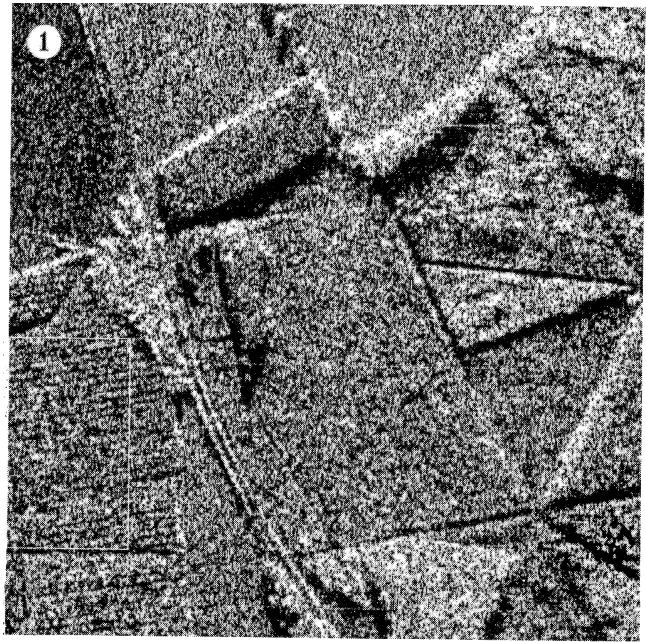


Figure 1: SAR image of a region of fields and woods.

cell as containing many randomly positioned scatterers. No attempt is made to represent the configuration within the cell. The physical processes of scattering and imaging can then be analysed theoretically. Where the cross section is spatially non-varying the image statistics should correspond to classical Rayleigh speckle. In Table I we compare the higher-order singlepoint moments of the detected intensity distribution for a region of open field, for which the cross section is expected to be essentially constant, with the theoretical values for pure speckle.

It is apparent that the statistics are indeed consistent with such a model. In general the detected intensity of each pixel of an image will then take the form of a convolution of the underlying cross-section fluctuation from pixel to pixel with a negative exponential speckle contribution. This process is clearly visible qualitatively in Figure 1. One would therefore expect that the intensity probability distribution will no longer be negative

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 11, 2019.

This appeared in *Philosophical Transactions of the Royal Society of London Series A*, 1988, vol. 324, no. 1579, pp. 397-407.

S P Luttrell’s contribution to this paper was the work on resolution enhancement.

Table I: Comparison of the intensity moments for the field texture.

moment	data	model	
		mean	SD
second	1.970	2.000	0.014
third	5.68	6.00	0.14
fourth	21.3	24.0	1.3

exponential but will have larger higher-order moments indicating the presence of excess fluctuations.

Having proposed a physical scattering model consistent with the observed speckle it is next necessary to provide a model for the cross-section fluctuations. In general, no physical description of the surface appears feasible so we are forced to consider a phenomenological model for the surface cross-section fluctuations. The aim of such a model is to represent the surface in such a way that the scattered radiation is consistent with the properties of the observed image; there is no attempt to reproduce the exact surface. We shall show that natural clutter textures can be treated as homogenous noise processes. In terms of Figure 1, therefore, we are concerned that the model should be capable of reproducing the statistical and correlation properties of clutter textures such as that shown enclosed by the white rectangle.

One such model that has been proposed is that the surface cross section,  $\sigma$ , should be represented by a random process with a gamma-distributed probability-density function (PDF)

$$p(\sigma) = \frac{1}{\langle \sigma \rangle} \left( \frac{\sigma}{\langle \sigma \rangle} \right)^{\nu-1} \frac{1}{\Gamma(\nu)} \exp \left[ -\frac{\sigma}{\langle \sigma \rangle} \right] \quad (2.1)$$

where  $\Gamma(\nu)$  is the gamma function of order  $\nu$  [4, 8–10]. This is the continuum analogue of the discrete negative-binomial distribution. On scattering the observed intensity PDF would then be the convolution of the gamma-distribution with the negative exponential distribution, namely the  $K$ -distribution given by

$$p(I) = \frac{1}{\langle I \rangle} \left( \frac{I}{\langle I \rangle} \right)^{\frac{1}{2}(\nu-1)} \frac{1}{\Gamma(\nu)} K_{\nu-1} \left[ 2 \left( \frac{I}{\langle I \rangle} \right)^{\frac{1}{2}} \right] \quad (2.2)$$

where  $K_{\nu-1}$  is the modified Bessel function ( $K$  distribution) of order  $\nu - 1$ . From Equation 2.1 it is apparent that this noise model is a generalisation of thermal noise (the case when  $\nu = 1$ ). The order parameter  $\nu$  serves to control the contrast and higher-order moments of the distribution. Such a noise model would result from a physical model in which the local cross section, or density of scatterers, depended on a birth-death-migration process [5]. Thermal noise describes a process in which only migration (diffusion) occurs. This noise process is the result of including an additional source and sink of scatterers. A variety of evidence from scattering experiments at both microwave and optical wavelengths has

Table II: Comparison of the intensity moments for the wooded texture denoted in Figure 1.

moment	data	full model		simple model		simulation	
		mean	SD	mean	SD	mean	SD
second	2.64	2.64	0.04	2.64	0.06	2.55	0.11
third	13.1	13.1	0.7	13.0	1.4	14	3
fourth	100	108	10	102	32	160	90

indicated that such a model is consistent with the data [6, 7, 15, 17, 18, 20, 22–24] though no unambiguous physical justification is possible.

Before embarking on a detailed comparison of the statistics it is essential to consider the correlation properties within the texture which often convey more information than the contrast fluctuations. Gamma-distributed noise can be regarded as the resultant of a random walk in  $\nu$  dimensions of a complex gaussian variable, i.e. thermal noise. Thus all the results for classical thermal noise can be generalised to the gamma-distributed equivalent by exploiting the appropriate factorisation properties [15]. For distributions with high contrast it is necessary to use fractional values of the order parameter  $\nu$ . Provided that the model truly represents the texture, it then contains fundamentally all the information available about that texture because gaussian processes are described entirely in terms of their mean and covariance. Various simple theoretical models of this kind have been proposed and compared with image data [15–17, 20].

Let us next apply this approach to the texture of the plantation of young trees shown in Figure 1 enclosed by a white rectangle. The first stage is to derive a reasonable model for the surface autocorrelation function (ACF) which, when imaged, gives an intensity ACF that closely represents the observed image result. For the texture shown the model requires three components for an approximate correspondence. A method also exists for obtaining even closer agreement by regarding the ACF of the original data as a perturbation of the model ACF [19]. Having obtained the form of the surface ACF the higher-order moments can then be calculated following the methods established earlier [15–17]. A comparison of the observed single-point moments with the predicted values for the first four moments is given in Table II. In addition the predicted values for a simple  $K$ -distribution having the observed second moment are included.

Both theoretical predictions are consistent with the original data within the theoretical errors included in the table. Strictly speaking an integrated  $K$ -distribution is not itself  $K$ -distributed so the simple model should not fit as well. However, for distributions with lower contrast than thermal noise the difference is often insignificant. The simple model, of course, is not capable of predicting how the statistics vary with the imaging resolution; for this the full theory is required.

We have so far demonstrated that random textures in coherent images can be represented in terms of a cor-

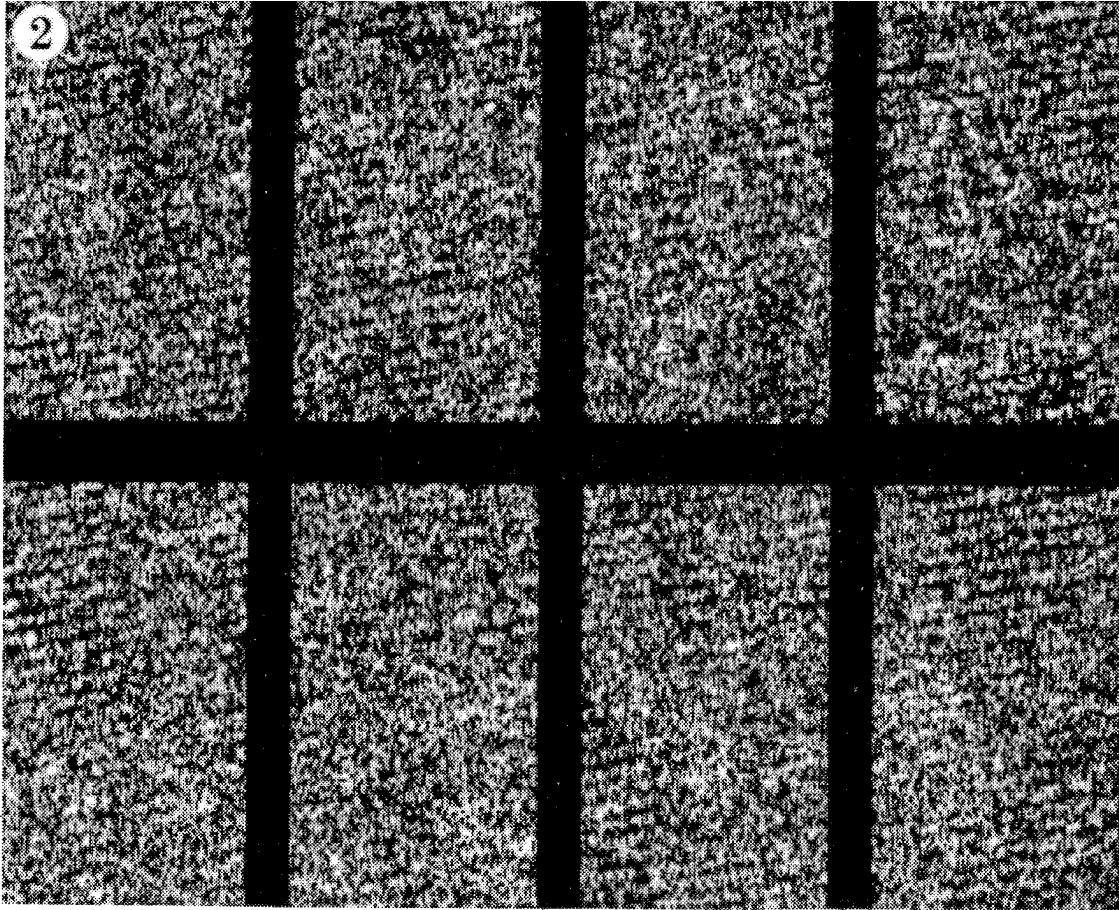


Figure 2: Comparison of original (top right) and several simulated textures for the region indicated in Figure 1.

related gamma-distributed surface cross section, which when imaged yields  $K$ -distributed correlated intensity fluctuations. Although only a single example is included here many other clutter textures in both radar and sonar images have been demonstrated to be capable of representation with the same fundamental model.

As a final test of the model it is possible to simulate textures having the same statistical properties as the original image by using a linear filter method [17, 21]. This method seeks to reproduce the two-point statistics, the covariance function, of the original image. The surface is first simulated by passing uncorrelated gamma-distributed noise through a linear filter. The filter weights are chosen so that their ACF is determined by the surface ACF [21]. The complex scattered field is then simulated by speckling the cross section with a random zero-mean gaussian variable for both the in-phase (I) and quadrature (Q) field components. This field is then convolved with the imaging response function of the system. A set of such simulations is compared with the original texture (top right) in Figure 2. Visually the agreement is excellent. As a further test the mean value and standard deviation of the higher-order single-point statistics of the simulations is included in the last two columns of

Table II. Again the agreement is within the statistics. This ability to simulate textures is of great importance in deriving statistical estimates of, for example, target detection probabilities obtained with various algorithms against such a clutter background.

The highest level of prior knowledge, the *contextual model*, describes the relation of defined textures within a scene. In terms of Figure 1, therefore, it would contain map information on the position of the boundaries of the woods and fields, statistical information about the correlation properties and order parameter of the texture in each region, plus other information, such as tree height, ground slope and aircraft altitude, to enable one to predict shadowing and other geometrical effects. In principle, the combination of all three stages in the model should enable us to predict coherent images that are statistically equivalent to a SAR image of the same region.

Once the model for the clutter is established it can be used in the detection of targets against this clutter background. The first stage in the process is to form a composite clutter model for the area created from combinations of map data, previous SAR images of the same area and other sensor data. This clutter model (prior knowledge) then defines the predicted statistical proper-

ties that are expected at each point in the SAR image. To detect ‘targets’ we then start with the assumption that the scene is entirely described in terms of the clutter map and search for anomalies which we define as targets. These anomalies may not be simply related to the image intensity but may depend on higher-order statistical properties of the image. We might, for instance, wish to detect changes of texture correlation length where the mean intensity and contrast remain the same. Whatever criterion is selected for the discrimination the clutter model is used to define a ‘target detection threshold map’ for that measure. Target detection then occurs when the value of the measure exceeds this threshold.

### III. TARGET RESOLUTION-ENHANCEMENT FROM PRIOR KNOWLEDGE

Once a target has been detected the next stage is to seek to classify that target. In many examples of SAR image the resolution of the system is of the same order as the size of targets of interest making any type of geometrical classification based on the distribution of cross section within the target essentially impossible. The second part of this paper now concentrates on the use of prior knowledge about targets to improve the effective resolution of the sensor. This is achieved by introducing a model to represent the target. There are therefore two key issues that have to be considered: (i) what is an appropriate form for the prior knowledge model and (ii) how is the model applied to yield the maximum additional information about the target? Before addressing these issues directly let us consider a particular example of a prior-knowledge model and examine the implications. The Rayleigh resolution criterion states that two point sources are just resolved when the first null of one point spread function (PSF) lies on the main lobe of the second. However, in the ‘monopulse’ technique of radar processing strong prior knowledge about the target is introduced, namely that it consists of two scatterers. We can then inspect the detailed shape of the image to determine the amplitudes, phases and positions of the two scatterers with greatly enhanced resolution, limited only by the noise in the system, not the PSF directly. This type of approach is common in many experimental fields where one defines a model to represent the process under study and then determines the parameters of that model. The approach is effective provided that the prior knowledge introduced in the model is in fact correct. It is therefore advisable to use as weak a form of prior knowledge as possible because there is less chance of introducing false information. Let us now consider what general properties of targets might be appropriate as prior knowledge in the context of SAR images.

Obviously it would be possible to generalise the monopulse technique by assuming that one had a number of point sources within the target and searching for the best fit as that number is varied. As the number in-

creases the ambiguity introduced by the noise means that a large number of alternative solutions exist and one is forced to introduce some arbitrary criterion for choosing a particular solution, for example, that with the smallest number of scatterers. This form of prior knowledge is generally speaking too strong because we cannot be sure that targets consist of point-like scatterers. A weaker form of prior knowledge that we can be confident about is that targets are of limited spatial extent. Furthermore we may assert that the target’s position and dimensions are well represented by the original image. The fundamental premise of all radar target detection algorithms is that a target exists where a large return is recorded. This form of prior knowledge is based on the same assumption and may be viewed with the same level of confidence. It is obviously true that pathological scatterer configurations can violate the assumption and so invalidate the result. One could only have complete confidence in the prior knowledge if one knew precisely what the target was, in which case there is no more information to be gained from the image. This concentration of the target cross section into a localised continuum form makes no assertions about the scattering centres within the target itself; this information will be derived by applying the prior knowledge to the image data.

In applying resolution-enhancement methods on an SAR image a small region enclosing a detected target will be selected. Thus the prior knowledge must combine background and target models. For the total model we assert that it is made up of a background region of uniform cross section containing many random scatterers per resolution cell. Imposed on this background at a position and with dimensions determined from the original image is a localised target within which the scatterer distribution is unknown. Comparative strengths of target and background can be deduced from the image data.

Let us now examine how this prior knowledge may be exploited in the reconstruction of a target at an enhanced resolution. We shall follow the methods previously proposed by [12] and [14] based on a bayesian reconstruction scheme. Suppose that our object function,  $f$  ( $= f(x)$ , a continuous complex variable denoting the scattering amplitude and phase), is imaged through a response function  $T$  to give a complex image vector  $\mathbf{g}$ . The imaging process can be denoted by the shorthand notation

$$\mathbf{g} = T f + \mathbf{n} \quad (3.1)$$

where the vector  $\mathbf{n}$  is the noise introduced in the process. Bayesian reconstruction then provides a means for relating the *a posteriori* PDF,  $P[f(x)|\mathbf{g}]$  (the conditional probability of a particular object given the image data), to the prior knowledge of the object, expressed in terms of the *a priori* PDF  $P[f(x)]$  (the predicted probability of a particular object) and the imaging process via the expression

$$P[f(x)|\mathbf{g}] \propto P[\mathbf{g}|f(x)] P[f(x)] \quad (3.2)$$

where  $P[\mathbf{g}|f(x)]$  describes the imaging properties in

terms of the conditional probability of the image given a particular object. Note that a one-dimensional form is used for simplicity. This *a posteriori* PDF is the basic construct from which inferences about the data, in particular the target reconstruction, are to be drawn.

To progress we need to make assumptions about the PDFs of the object and the noise. Let us assume that the object distribution, the *a priori* PDF, is a zero-mean gaussian. This will not accurately represent the non-gaussian statistics that actually occur but will suffice to specify the scale of the amplitude of  $f(x)$ . In principle we cannot specify what the statistics of the object should be without knowing in advance what the target is. Thus the gaussian assumption is a reasonable compromise. The properties of the object are then all contained in its covariance matrix,  $W$ . In the imaging process in the absence of noise the conditional PDF  $P[\mathbf{g}|f(x)]$  will be a delta function because the mapping is determined precisely by the imaging function  $T$ . When noise is introduced it broadens the conditional PDF. Let us assume that the measurement noise is also a zero-mean gaussian process specified completely in terms of its covariance matrix,  $N$ . Note that it is not necessary to assume that the noise is uncorrelated; correlated noise can be treated correctly. It can then be shown [12, 14] that the *a posteriori* PDF can be expressed in the form

$$P[f(x)|\mathbf{g}] \propto \exp \left[ - (f - \langle f \rangle)^\dagger (T^\dagger N^{-1} T + W^{-1}) (f - \langle f \rangle) \right] \quad (3.3)$$

where ( $\dagger$  denotes hermitian conjugate)

$$\langle f \rangle = W T^\dagger (T W T^\dagger + N)^{-1} \mathbf{g} \quad (3.4)$$

This *a posteriori* PDF contains the available information about the object. In practice we require to select some particular member of this distribution as representing the ‘best’ reconstruction. A logical choice would be to choose the average value of  $f(x)$ , defined in Equation 3.4.

Let us now make the simplifying assumptions that both the object and noise covariances are delta-correlated. The ‘weight’  $W$  is now a real function which can be related to a profile function  $P(x)$  such that

$$W(x_1, x_2) = |P(x_1)|^2 \delta(x_1 - x_2) \quad (3.5)$$

It can be shown [12, 14] that the object reconstruction may then be expressed as

$$f_{\text{rec}}(x) = P(x) \sum_{k=1}^m \frac{g_k}{(\lambda_k + N)^{\frac{1}{2}}} u^k(x) \quad (3.6)$$

where  $g_k$  are the components of the image  $\mathbf{g}$  in the image basis vectors and  $u^k(x)$  are the equivalent eigenvectors in object space. These object and image eigenvectors correspond to singular value decomposition (SVD) of the imaging process and are characterised by the eigenvalues  $\lambda_k$ . If the signal to noise ratio is large enough the results

of this bayesian reconstruction method are identical to those obtained elsewhere using an SVD technique [1–3].

Let us next examine the *a posteriori* distribution in more detail to determine where the dominant information is contained. It can be shown that any object can be expanded in terms of object space vectors in the form

$$f(x) = P(x) \sum_{k=1}^m \frac{f_k}{(\lambda_k + N)^{\frac{1}{2}}} u^k(x) + f_c(x) \quad (3.7)$$

where the expansion coefficients  $f_k$  are the components of the image of the object  $f$  in the image space. The second term in Equation 3.7,  $f_c$ , denotes the sum of the components of  $f$  that do not project onto the image space basis. Because these components have no contribution in the image (data) space they can be said to be ‘invisible’ to the data. Thus we cannot infer anything about these components from a measurement of the image  $\mathbf{g}$ . All that we can say about them is contained in the form of the *a priori* PDF that we have defined.

If we next consider the implication of the first term in Equation 3.7 we see that the contribution to the reconstructed object of each of the vectors  $f_k$  is characterised by an eigenvalue  $\lambda_k$ , which defines the visibility of that contribution. Where these eigenvalues are smaller than the noise  $N$  the corresponding data have insufficient energy to dominate the noise and so cannot convey information about the object reconstruction. In this situation also the only information available is that introduced in the prior knowledge. Only those contributions  $f_k$  whose eigenvalues exceed the noise provide additional information beyond that represented by the prior knowledge. A more detailed discussion of the information content of these images, expressed in terms of mutual information, is to be found in [11] and [13] and [14].

If we next substitute for  $f$  and  $\langle f \rangle$  ( $= f_{\text{rec}}$ ) from Equation 3.6 and Equation 3.7 into Equation 3.3 we obtain for the *a posteriori* PDF

$$P[f(x)|\mathbf{g}] \propto \exp[f_c^\dagger |P|^{-2} f_c] \exp \left[ - \sum_{k=1}^m |f_k - g_k|^2 \left( \frac{1}{N} - \frac{1}{\lambda_k + N} \right) \right] \quad (3.8)$$

The first factor corresponds to a piece of the *a priori* PDF that is not changed in passing to the *a posteriori* PDF, since  $f_c$  is invisible in the data space. The second factor depends on the data and expresses the influence that the data,  $\mathbf{g}$ , have on the *a posteriori* PDF. Each mode carries an independent piece of useful information about the object, which vanishes as the noise increases.

Let us now illustrate the process by using Equation 3.6 to calculate reconstructed objects for a simulated target embedded in clutter. We shall simulate a two-dimensional imaging system whose PSF is that for a square aperture. The data acquisition system measures complex field Nyquist-spaced samples (I and Q) on a

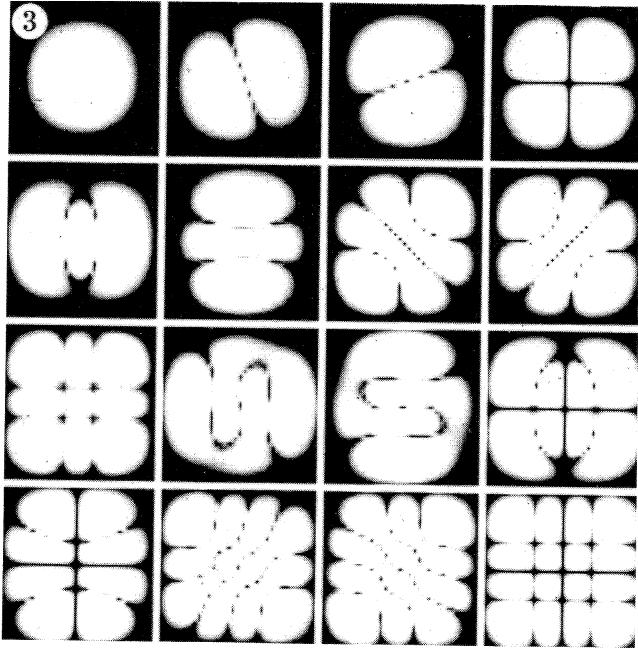


Figure 3: Stages in the first iteration of the reconstruction process for the simulated target. Original image (left), prior knowledge weight (centre), reconstruction (right); logarithmic display.

square lattice. Because the target is localised we shall consider an image space of  $9 \times 9$  samples. This must be sufficiently large that no sidelobes of the target are visible outside the sampled image. In the object space each image sample is subdivided into  $4 \times 4$  subsamples without lying outside the bounds of the  $9 \times 9$  array. This sampling is deemed adequate to provide at least Nyquist sampling on the enhanced-resolution object reconstructions. For the sake of this demonstration a typical small target is simulated which leads to the observed image shown on the left of Figure 3.

Having defined the sampling spaces we next require estimates of  $W$  and  $N$ . In typical SAR images the measurement noise and the background clutter speckle are both essentially delta-correlated. Furthermore the noise is negligible compared with the clutter speckle fluctuations and may be ignored. If we also assume that the target is delta-correlated then the target and clutter prior knowledge are expressed in terms of the profile weighting function  $W(x) = |P(x)|^2$  which has the form of a cross section. Having determined the weighting function we next compute the reconstructed object  $f_{\text{rec}}(x)$  from the image  $\mathbf{g}$  as given in Equation 3.6. In the analysis it can be shown [12] that the reconstruction vectors in the object space include the profile function  $P$ . It is instructive to examine the consequence of introducing no prior knowledge into the reconstruction. This is equivalent to setting  $P$  constant; we are totally non-committal about where the scatterers are located within the image. In Figure 4, we show the first 16 of the 81 possible ob-

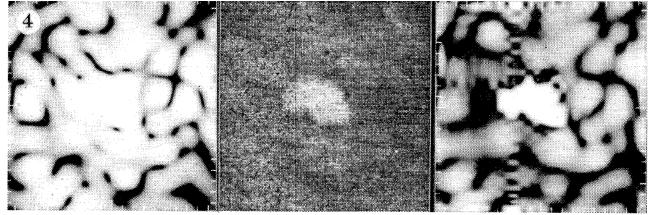


Figure 4: First 16 of the weighted object eigenvectors when no prior knowledge is introduced; logarithmic display.

ject space eigenvectors arranged in order of decreasing eigenvalues. Note that as the eigenvalue decreases the spatial frequency content of the eigenvectors increases. In general higher spatial frequency terms are more attenuated by the imaging process leading to the ordering of the functions. However, for a uniform weighting function all eigenvalues are approximately equal so that all 81 are required to perform the reconstruction. As one might expect, a reconstruction based on this prior knowledge is essentially the same as the original image and can be shown [12] to provide an interpolated form of the data. We may use this interpolated form to make cross-section estimates for the combined clutter and target prior-knowledge weighting function.

Initially we use threshold detection to separate those pixels that may be defined as ‘target’ from those we define as ‘clutter’. The mean intensity of those pixels that lie below the threshold is taken as the background cross-section estimator whereas the intensity of the pixels that exceed the threshold is taken as the estimate of the local target cross section. This process whereby the data is used to generate ‘prior knowledge’ may seem incestuous. It is, however, important to appreciate that the prior knowledge does not reside in the image data but in the rules that are used in separating the image into the two components and setting up the weighting function. In particular we are asserting, in common with nearly all other radar systems, that locally bright pixels are extremely likely to have been derived from local concentrations of cross section rather than from the myriad of other possible configurations that would give the identical image. This rule is not a fundamental constraint so much as a consequence of empirical observation of typical SAR images.

Let us now apply these methods to a particular simulated image, shown on the left of Figure 3. This image has been formed from an array of four dominant scatterers. However, the resolution is so poor that the image shows merely some evidence of asymmetry. The signal to background ratio is 20 dB. Performing threshold detection we derive the prior knowledge cross-section weighting function shown in the centre of the figure. Because the method uses threshold detection there is a statistical probability that pixels which might actually belong to the background class will emerge as targets, and vice versa. Two examples of the former effect can be observed. This

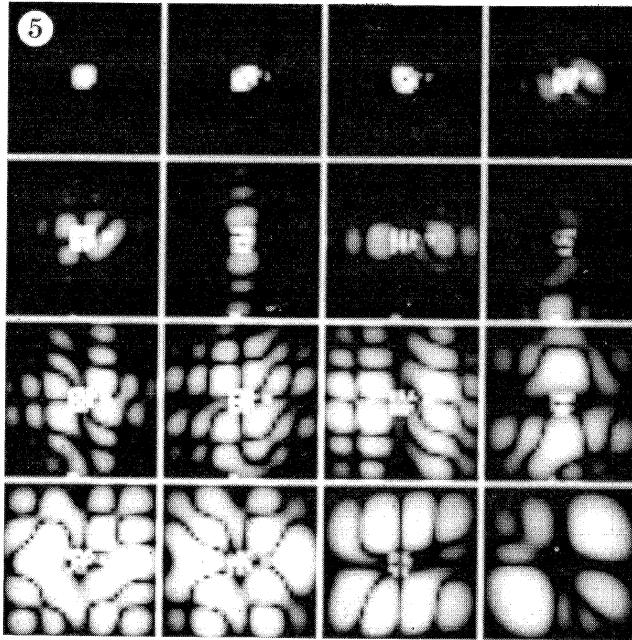


Figure 5: First 16 of the weighted object eigenvectors when localised target prior knowledge is introduced; logarithmic display.

weighting function now takes the form of the *a priori* PDF in the bayesian reconstruction scheme. The resulting reconstructed object is shown on the right of Figure 3. Comparison with the original image reveals that the apparent resolution in the image has indeed been improved. The degree of enhancement depends strongly on the nature of the data and it is not possible to define a global enhancement factor. Some understanding of how the enhancement is achieved may be gained from studying the object space eigenvectors for the case where a localised target leads to a strongly peaked weighting function. The first 16 of these modes are shown in Figure 5. Again they are presented in order of decreasing eigenvalue. Note how the first few modes concentrate almost exclusively on a strongly localised region contained within the central peak of the data. No such effect was observed with the uniform weight (Figure 4). Unlike the uniform example also, the eigenvalues of the modes decrease comparatively rapidly as the order increases. Thus a small region of the object space contains a large amount of the information present in the data. Only a comparatively small amount of the information, contained in the weak higher-order modes, is used to reconstruct the background clutter. This is the means whereby resolution enhancement is achieved in this approach.

Having applied the resolution-enhancement technique once it is tempting to attempt to apply it again. There are two main objections to such a step. Firstly, one has to vindicate the choice of the prior knowledge in an iterative procedure. Regardless of the true nature of the target this type of processing will tend to concentrate

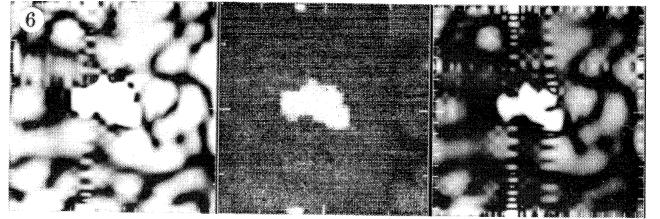


Figure 6: Stages in the second iteration of the reconstruction process for the simulated target. First reconstruction (left), prior knowledge weight (centre), reconstruction (right); logarithmic display.

the object cross section into a number of localised scatterers. If the image were of an object such as a tree then this progressive concentration of the cross section would be inappropriate since the scatterers are distributed in a comparatively diffuse manner. On the other hand, man-made objects, such as vehicles, can usually be represented in terms of a small number of point scatterers. Thus, if it were possible to introduce further prior knowledge that the object in question is indeed man-made, we may proceed with further iterations of the reconstruction process treating the output of the first reconstruction as if it were itself an image.

However, proceeding with iterated reconstruction introduces a second limiting factor into consideration. The presence of the random clutter speckle within the image introduces a noise component into the reconstruction so that, eventually, the result is dominated by the noise. It is clearly essential to iterate the process only for as long as this noise component permits. In this instance a second iteration can be justified. Exactly the same procedure is adopted as before with the previous reconstruction treated as the new image (Figure 6, left).

The weighting function, shown in the centre of Figure 6 is derived in the same way as before. Note that the threshold detection algorithm still passes the two false alarm targets detected in the original data. The reconstruction shows still more resolution enhancement and even serves to split the original image into four dominant scatterers (Figure 6, right).

Let us conclude by comparing the reconstruction with an image of the original object where the resolution is increased by a factor of four in each direction. In Figure 7 reconstructed images after two iterations with (top right) and without (top left) a clutter background are compared with equivalent high-resolution images (bottom).

The example on the left without background gives a very high-quality reconstruction, closely similar to the image, because the only limiting factor is the noise introduced by rounding errors in the computer. The example on the right, with an initial target to background ratio of 20 dB, shows the distortion that has been introduced by the random speckle. Nevertheless the information content of the reconstruction has been considerably enhanced by the introduction of prior knowledge when

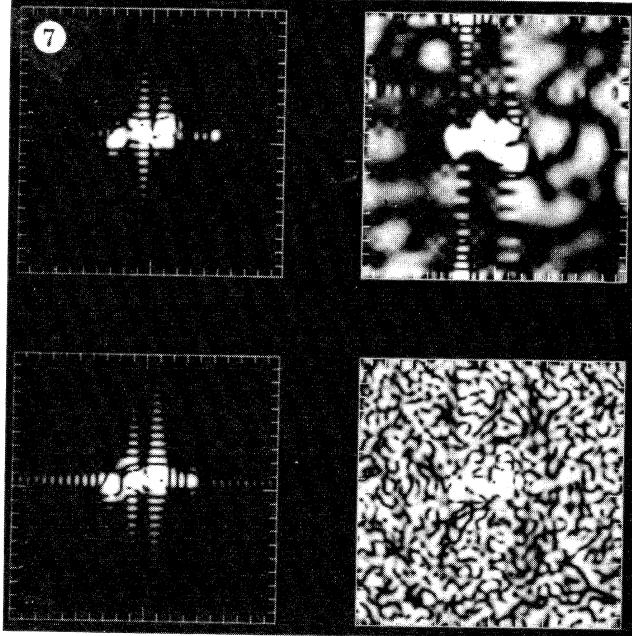


Figure 7: Comparison of the final reconstructions with high resolution images of the same object. Reconstruction without background (top left), reconstruction with background (top right), high-resolution image without background (bottom left), high-resolution image with background (bottom right); logarithmic display.

compared with the original image (left, Figure 3). Note

also that the process has suppressed the background clutter compared to the peaks of the reconstructed target. Although this reconstruction may not in itself be sufficient to classify the target it does represent a significant step in that direction. This has been achieved entirely by post-processing the original data and does not require any modification of the existing SAR sensor.

#### IV. CONCLUSIONS

The task of interpreting SAR images is very difficult. In addition to the problems associated with the complexity of the image and the sheer quantity of data, which it shares with many other sensor systems, there are the further limitations of speckle and low resolution. It would be extremely useful to be able to summarise the content of a SAR image in terms of a few significant features, ‘targets’, set against a ‘clutter’ background. One can then concentrate one’s efforts on the targets. In the first main section of the paper we presented a model of SAR terrain that could be used to define a target detection threshold and to suggest suitable detection algorithms. Ensuing target detections, which are essentially a result of the first stage of the reconstruction process, can then be studied in more detail including, in particular, resolution enhancement as outlined in the second main section of the paper. Both steps are vital tools in the total problem of SAR image interpretation.

- 
- [1] M Bertero, P Brianzi, P Parker, and E R Pike, *Resolution in diffraction-limited imaging, a singular value analysis III. The effect of sampling and truncation of the data*, Optica Acta **31** (1984), no. 2, 181–201.
  - [2] M Bertero, C de Mol, E R Pike, and J G Walker, *Resolution in diffraction-limited imaging, a singular value analysis IV. The case of uncertain localisation or non-uniform illumination of the object*, Optica Acta **31** (1984), no. 8, 923–946.
  - [3] M Bertero and E R Pike, *Resolution in diffraction limited imaging, a singular value analysis I. The case of coherent illumination*, Optica Acta **29** (1982), no. 6, 727–746.
  - [4] E Jakeman, *On the statistics of K-distributed noise*, Journal of Physics A: Mathematical and General **13** (1980), no. 1, 31–48.
  - [5] ———, *Statistics of integrated Gamma-Lorentzian intensity fluctuations*, Optica Acta **27** (1980), no. 6, 735–741.
  - [6] E Jakeman, E R Pike, and P N Pusey, *Photon correlation study of stellar scintillation*, Nature **263** (1976), no. 5574, 215–217.
  - [7] E Jakeman and P N Pusey, *Non-Gaussian fluctuations in electromagnetic radiation scattered by random phase screen I. Theory*, Journal of Physics A: Mathematical and General **8** (1975), no. 3, 369–410.
  - [8] ———, *A model for non-Rayleigh sea echo*, IEEE Transactions on Antennas and Propagation **24** (1976), no. 6, 806–814.
  - [9] ———, *RADAR-77*, ch. Statistics of non-Rayleigh microwave sea echo, pp. 105–109, IEE, London, 1977.
  - [10] ———, *Significance of K-distributions in scattering experiments*, Physical Review Letters **40** (1978), no. 9, 546–550.
  - [11] S P Luttrell, *A new method of sample optimisation*, Optica Acta **32** (1985), no. 3, 255–257.
  - [12] ———, *Prior knowledge and object reconstruction using the best linear estimate technique*, Optica Acta **32** (1985), no. 6, 703–716.
  - [13] ———, *The use of transinformation in the design of data sampling schemes for inverse problems*, Inverse Problems **1** (1985), no. 3, 199–218.
  - [14] S P Luttrell and C J Oliver, *Prior knowledge in synthetic aperture radar processing*, Journal of Physics D: Applied Physics **19** (1986), no. 3, 333–356.
  - [15] C J Oliver, *A model for non-Rayleigh scattering statistics*, Optica Acta **31** (1984), no. 6, 701–722.
  - [16] ———, *Correlated K-distributed clutter models*, Optica Acta **32** (1985), no. 12, 1515–1547.
  - [17] ———, *The interpretation and simulation of clutter textures in coherent images*, Inverse Problems **2** (1986), no. 4, 481–518.
  - [18] ———, *A model for non-Rayleigh scattering statistics*,

- Proceedings of the IMA Conference on Wave Propagation and Scattering (Oxford) (B J Uscinski, ed.), Clarendon Press, 1986, p. 155.
- [19] \_\_\_\_\_, *On the simulation of coherent clutter textures with arbitrary spectra*, Inverse Problems **3** (1987), no. 3, 463–475.
- [20] \_\_\_\_\_, *Correlated clutter in coherent imaging*, Proceedings of the IMA conference on applications of mathematics to remote sensing (Oxford), Clarendon Press, 1989, pp. 125–141.
- [21] C J Oliver and R J Tough, *On the simulation of correlated K-distributed random clutter*, Optica Acta **33** (1986), no. 3, 223–250.
- [22] G Parry, P N Pusey, E Jakeman, and J G McWhirter, *Focussing by a random phase screen*, Optics Communications **22** (1977), no. 2, 195–201.
- [23] K D Ward, *Compound representation of high resolution sea clutter*, Electronics Letters **17** (1981), no. 16, 561–565.
- [24] \_\_\_\_\_, *RADAR-82*, ch. A radar sea clutter model and its application to performance assessment, pp. 203–207, IEE, London, 1982.



# A Maximum Entropy Approach to Sampling Function Design \*

S P Luttrell

Royal Signals and Radar Establishment, St Andrews Rd, Malvern, Worcestershire WR14 3PS, UK

We use the maximum entropy principle to infer the information content of sampling functions of data sets. This may be used as a quantitative measure of the suitability of sampling schemes for use in inverse problems. We present a practical numerical means of evaluating the integrals which appear in the analysis, and we demonstrate how the method is applied to the special case of homogeneous texture analysis.

## I. INTRODUCTION

Inverse problems divide naturally into two stages: the choice and measurement of suitable data, followed by its inversion to recover some parameters of interest. There is usually an intermediate stage where the raw data set is reduced (or compressed) into a more amenable form without too much loss of information relevant to solving the particular inverse problem of interest. Data reduction is important when the raw data set consists of a large number of measurements (such as the pixels of an image), because the overall storage and communication requirements of the inversion algorithm may then be reduced. A trivial example of this is the compression of an oversampled image.

We shall use information theoretic techniques [3, 11] to suggest a method of designing useful data compression schemes. This allows us to derive adaptively a set of sampling functions of the data which provides the information which is necessary for solving the inverse problem. Our approach may be viewed as a nonlinear version of the well known singular value decomposition (SVD) signal processing technique [2, 4, 10]. Strictly speaking, linear techniques (such as the SVD) can manipulate only the first and second-order statistical properties (mean and covariance) of the data [4, 10]. Therefore they impose an implicit gaussian assumption on the probability density function (PDF) which is used to describe the system being analysed. On the other hand, non-gaussian PDFs require higher order statistical properties for their full description, so linear signal processing techniques can fail. For instance radar clutter is usually non-gaussian, so solving inverse problems such as target detection and location in the presence of clutter can require knowledge of the properties of high-order clutter statistics (e.g. high-order moments). The  $K$ -distributed sea clutter model is an excellent example of this approach [13].

The sampling functions which we obtain may be used to construct a maximum entropy approximation to the PDF of the data which are being compressed. We con-

struct this PDF completely from a set of training data, in contrast with the usual approach where a PDF model is invoked (sometimes on physical grounds) with only a few parameters left free to be determined. In a previous paper [9] we presented some results on the sampling of a data set which is known *a priori* to be a Markov random field (MRF), and hence described by a Gibbs distribution. We now make no such assumption, so our new results apply to data sets with an arbitrary statistical structure. The sampling functions must retain those parts of the data sets which are sensitive to the parameters of interest; any other information is not relevant to solving the inverse problem of recovering these parameters. The optimal form of data compression must match the particular inverse problem which ultimately is to be solved. Some examples of this approach in the context of sample position (rather than sample function) optimisation in the SVD approach are described in [5]. Conventionally data compression, as applied to images, seeks to preserve the information which is required in order to recover the original image. However, this is not our goal, for we wish to recover only a set of parameters which serve as an interpretation of the image - this allows a far greater degree of data compression. In image processing terminology the compressed image should contain the information which is necessary for image classification but not image reconstruction.

Following the information theoretic approach we regard a data set as a sample  $\mathbf{x}$  from a probability density function  $P(\mathbf{x}|c = c_0)$  over the ensemble of data sets  $\{\mathbf{x} : c = c_0\}$  which could arise under a particular set of controlled conditions  $c = c_0$ . The compressed data set also has an associated PDF, which is related to  $P(\mathbf{x}|c)$  by the transformation which is induced by the sampling functions. In this treatment data noise of all forms (additive, multiplicative, quantisation, etc) are regarded as part of the data itself, and so  $P(\mathbf{x}|c)$  is actually a composite of the true (but unknown) PDF and the noise PDF. In the Bayesian approach to solving the inverse problem of recovering  $c$  from  $\mathbf{x}$  we need to use the *a posteriori* PDF  $P(c|\mathbf{x})$  which is obtained by using Bayes' theorem. The  $\mathbf{x}$  dependence of  $P(c|\mathbf{x})$  is determined entirely by  $P(\mathbf{x}|c)$  which already includes any effects of data noise. So, when we are interested only in the information about  $c$  which is contained in  $\mathbf{x}$ , an appropriate data compression needs only to preserve information about the composite PDF  $P(\mathbf{x}|c)$ . For instance, in image analysis  $c$  might specify

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This paper appeared in *Inverse Problems*, 1988, vol. 4, no. 3, pp. 829-841. Received 9 November 1987, in final form 18 January 1988.  
© Controller, Her Majesty's Stationery Office, London, 1988.

that a particular object, a face for instance, is present in the scene, and that its position and orientation are unimportant. The ensemble of  $\mathbf{x}$  from which we estimate  $P(\mathbf{x}|c)$  then consists of all images of that particular object at all positions and orientations. In remote sensing image analysis an analogous problem occurs when  $c$  specifies a single type of terrain (e.g. wood, town, pasture, etc). The corresponding image then has a characteristic texture, and the ensemble of images  $\mathbf{x}$  which corresponds to  $c$  defines  $P(\mathbf{x}|c)$ . When the characteristic length scale of the texture is much smaller than the size of an image we may use a single image to represent the full ensemble, although this naturally leads to greater uncertainties in the estimated  $P(\mathbf{x}|c)$ . In both these examples the optimum choice of sampling functions for data compression will be determined by the parameters  $c$  which we wish to recover.

In Section II we review the maximum entropy principle as a vehicle for reconstructing a PDF from a limited set of information which is expressed in the form of constraints. This technique is both simple and non-committal; it recovers exactly what information is present in the constraints, and nothing more. This property is essential for our purposes because we can then systematically test the sampling functions for their information preserving properties. In Section III we measure the quality of the PDF reconstruction by defining a relative entropy, and we show how to calculate the change in relative entropy which occurs when the sampling functions are changed in various ways. These results may be used in the design of optimisation methods for obtaining the best possible sampling functions. In Section IV we present a practical numerical scheme for computing the contribution to the relative entropy from a sampling function, and we express the results in a form which is suitable for application to sparse data. In Section V we demonstrate how our technique may be applied to the particular problem of sampling textured images; this is of interest in remote sensing applications. The sampling functions which we obtain may be used to recover an MRF model of the texture (if it is required).

Our principal new results are contained in Section III where we present a framework for sampling scheme design. These results extend and unify those found in [6–9].

## II. THE MAXIMUM ENTROPY PRINCIPLE

We could consider the set (or support) of all values that  $c$  is likely to take, and derive a sampling function scheme which is sufficient to preserve the information which is necessary for solving the inverse problem of recovering  $c$  subject to this support. However, we shall address instead the simpler problem of constructing the PDF  $P(\mathbf{x}|c)$  which derives from a single  $c$ , so we shall replace the notation  $P(\mathbf{x}|c)$  by the notation  $P(\mathbf{x})$ . We shall denote the compressed data as  $\mathbf{s}$ , and the sampling functions which are used to produce the compressed data as

$\mathbf{s}(\mathbf{x})$ . The components of  $\mathbf{s}$  are not usually in one-to-one correspondence with the components of  $\mathbf{x}$ , and usually  $\mathbf{s}$  has far fewer components than  $\mathbf{x}$  (as required of a data compression scheme).

The only requirement of our data compression scheme is that  $P(\mathbf{x})$  may be recovered from observations of  $\mathbf{s}(= \mathbf{s}(\mathbf{x}))$ . At this stage we have not determined which particular functions  $\mathbf{s}(\mathbf{x})$  are suitable, but there are some simple inverse problems where  $\mathbf{s}(\mathbf{x})$  may be calculated in closed form. For instance, if we wish to determine only the scattering cross section of a homogeneous scene which is uniformly illuminated with coherent radiation it is usually sufficient to measure the sum of the modulus squared values of the complex image samples. More generally, the full structure of  $P(\mathbf{x})$  will be a complicated function of  $\mathbf{x}$  which we have to estimate from a set of examples of  $\mathbf{x}$ .

In order to derive the form of  $P(\mathbf{x})$  which we can reconstruct from  $\mathbf{s}$  we need to consider the PDF  $p(\mathbf{s})$  over  $\mathbf{s}$ , which is given by

$$p(\mathbf{s}) = \int d\mathbf{x} P(\mathbf{x}) \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \quad (2.1)$$

where the Dirac delta function forces the  $\mathbf{x}$  integration to extract that part of the mass of  $P(\mathbf{x})$  which lies on the surface  $\mathbf{s}(\mathbf{x}) = \mathbf{s}$ . Now suppose that all we know about  $P(\mathbf{x})$  is expressed indirectly as an estimate of  $p(\mathbf{s})$  which was determined by observing some examples of  $\mathbf{x}$ . We shall pose an inverse problem, namely: reconstruct the best estimate of  $P(\mathbf{x})$  given the estimate of  $p(\mathbf{s})$ . As usual with inverse problems we must select one out of a number of feasible reconstructions, and so we must define a cost function which discriminates amongst the various feasible reconstructions in order to select a unique best candidate.

For PDF reconstruction we may use the maximum entropy principle [3] to select a reconstruction  $Q(\mathbf{x})$  of  $P(\mathbf{x})$ . This principle has the advantage of imposing only that structure on  $Q(\mathbf{x})$  which is positively demanded by the data  $p(\mathbf{s})$ , but otherwise leaving  $Q(\mathbf{x})$  completely non-committal. More fundamentally, this is achieved by selecting that  $Q(\mathbf{x})$  which maximises the number of ways in which the constraints may be satisfied.  $Q(\mathbf{x})$  may be obtained as the solution to an optimisation problem specified by the cost function

$$C \equiv - \int d\mathbf{x} Q(\mathbf{x}) \log(Q(\mathbf{x})) - \int d\mathbf{s} \lambda(\mathbf{s}) \left( \int d\mathbf{x} Q(\mathbf{x}) \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) - p(\mathbf{s}) \right) \quad (2.2)$$

where the first term is the entropy of  $Q(\mathbf{x})$  and the second term introduces the required constraints by using a Lagrange multiplier function  $\lambda(\mathbf{s})$ . Note that our constraints are more general than those used in [3], but the principle is nevertheless the same. Functionally differentiating with respect to  $Q(\mathbf{x})$  yields

$$\begin{aligned}
\frac{\delta C}{\delta Q(\mathbf{x})} &= -\frac{\delta}{\delta Q(\mathbf{x})} \int d\mathbf{y} Q(\mathbf{y}) \log(Q(\mathbf{y})) - \int d\mathbf{s} \lambda(\mathbf{s}) \frac{\delta}{\delta Q(\mathbf{x})} \int d\mathbf{y} Q(\mathbf{y}) \delta(\mathbf{s} - \mathbf{s}(\mathbf{y})) \\
&= -\int d\mathbf{y} \delta(\mathbf{x} - \mathbf{y}) (\log(Q(\mathbf{y})) + 1) - \int d\mathbf{s} \lambda(\mathbf{s}) \int d\mathbf{y} \delta(\mathbf{x} - \mathbf{y}) \delta(\mathbf{s} - \mathbf{s}(\mathbf{y})) \\
&= -\log(Q(\mathbf{x})) - 1 - \lambda(\mathbf{s}(\mathbf{x}))
\end{aligned} \tag{2.3}$$

where we have used  $\frac{\delta Q(\mathbf{y})}{\delta Q(\mathbf{x})} = \delta(\mathbf{x} - \mathbf{y})$ . Setting  $\frac{\delta C}{\delta Q(\mathbf{x})}$  to zero gives

$$Q(\mathbf{x}) = \exp(-1 - \lambda(\mathbf{s}(\mathbf{x}))) \tag{2.4}$$

i.e.  $Q(\mathbf{x})$  is a function of only the statistic  $\mathbf{s}(\mathbf{x})$ . We may determine the function  $\lambda(\mathbf{s})$  by ensuring that  $Q(\mathbf{x})$  satisfies Equation 2.1 with  $Q(\mathbf{x})$  substituted for  $P(\mathbf{x})$  (this introduces the required constraint on  $Q(\mathbf{x})$ ), which gives

$$\begin{aligned}
p(\mathbf{s}) &= \int d\mathbf{x} \exp(-1 - \lambda(\mathbf{s}(\mathbf{x}))) \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \\
&= \exp(-1 - \lambda(\mathbf{s})) \int d\mathbf{x} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x}))
\end{aligned} \tag{2.5}$$

whence

$$Q(\mathbf{x}) = \frac{p(\mathbf{s}(\mathbf{x}))}{Z(\mathbf{s}(\mathbf{x}))} \tag{2.6}$$

where we have defined

$$Z(\mathbf{s}) \equiv \int d\mathbf{x} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \tag{2.7}$$

$Z(\mathbf{s})$  is the PDF over  $\mathbf{s}$  which would arise from a uniform PDF over  $\mathbf{x}$ , scaled up by the total number of states  $\mathbf{x}$  (compare Equation 2.7 with Equation 2.1), and  $Z(\mathbf{s}(\mathbf{x}))$  is the value of  $Z(\mathbf{s})$  at the point  $\mathbf{s} = \mathbf{s}(\mathbf{x})$ .  $Z(\mathbf{s})$  is a highly singular, but convenient, function to introduce into our calculations because it removes the need to count states explicitly. All measurable quantities are devoid of such singularities, so using  $Z(\mathbf{s})$  may be regarded as nothing more than a mathematical trick.

The form of the expression for  $Q(\mathbf{x})$  in Equation 2.6 has a very appealing interpretation.  $\mathbf{s}(\mathbf{x})$  maps  $\mathbf{x}$  to some  $\mathbf{s}$ , and it also defines a surface of constant  $\mathbf{s}(\mathbf{x}) = \mathbf{s}$  in  $\mathbf{x}$  space on which  $\mathbf{x}$  happens to lie. The total probability (as defined by  $P(\mathbf{x})$ ) that  $\mathbf{x}$  maps to between the surfaces  $\mathbf{s}(\mathbf{x}) = \mathbf{s}$  and  $\mathbf{s}(\mathbf{x}) = \mathbf{s} + \delta\mathbf{s}$  is given by  $p(\mathbf{s}(\mathbf{x})) \delta\mathbf{s}$ . Similarly the total number of states  $\mathbf{x}$  which map to between these two surfaces is  $Z(\mathbf{s}(\mathbf{x})) \delta\mathbf{s}$ . A non-committal estimate of  $P(\mathbf{x})$  must assign the same PDF to each such point  $\mathbf{x}$ , so  $Q(\mathbf{x})$  must be the ratio of  $p(\mathbf{s}(\mathbf{x})) \delta\mathbf{s}$  to  $Z(\mathbf{s}(\mathbf{x})) \delta\mathbf{s}$ , as we obtained in Equation 2.6. So we see that the maximum entropy method provides a consistent and non-committal estimate of  $P(\mathbf{x})$  under the imposed constraints. Furthermore, a unique solution  $Q(\mathbf{x})$  to the constrained maximum entropy problem posed in Equation 2.2 is guaranteed, as we have shown by explicit construction leading to Equation 2.6.

We may immediately deduce from the structure of  $Q(\mathbf{x})$  that  $P(\mathbf{x})$  will be reconstructed exactly if it is a function of only  $\mathbf{s}(\mathbf{x})$  in the first place; in such a case  $\mathbf{s}(\mathbf{x})$  is a sufficient set of statistics for  $P(\mathbf{x})$ . In [9] we showed how the *a priori* knowledge that  $P(\mathbf{x})$  has an MRF structure could be used to aid the search for a suitable  $\mathbf{s}(\mathbf{x})$ . However, in more general cases we do not know in advance what form of  $\mathbf{s}(\mathbf{x})$  is suitable, so we must derive an adaptive scheme for discovering an  $\mathbf{s}(\mathbf{x})$  which is sufficient to obtain an information-preserving compressed representation  $p(\mathbf{s})$  of  $P(\mathbf{x})$ . Ideally the dimensionality of  $\mathbf{s}(\mathbf{x})$  should be as small as possible consistent with retaining as much information about  $P(\mathbf{x})$  as possible. Furthermore, some quantitative measure of the information content of each component of  $\mathbf{s}(\mathbf{x})$  is desirable in order to evaluate the cost versus the benefit of measuring each component in the first place. These last two considerations are decisive in choosing an appropriate scheme for discovering  $\mathbf{s}(\mathbf{x})$ .

### III. RELATIVE ENTROPY AS A MEASURE OF THE QUALITY OF STATISTICS

If the sampling functions  $\mathbf{s}(\mathbf{x})$  are perturbed then the maximum entropy estimate  $Q(\mathbf{x})$  of  $P(\mathbf{x})$  will change. In particular, there will be some  $\mathbf{s}(\mathbf{x})$  which give rise to reconstructions  $Q(\mathbf{x})$  which are in some sense better estimates of  $P(\mathbf{x})$  than are other  $Q(\mathbf{x})$ . An information theoretic measure of the difference between  $P(\mathbf{x})$  and  $Q(\mathbf{x})$  is the relative entropy  $S$  [3] defined as

$$\begin{aligned}
S &\equiv - \int d\mathbf{x} P(\mathbf{x}) \log \left( \frac{P(\mathbf{x})}{Q(\mathbf{x})} \right) \\
&\leq 0
\end{aligned} \tag{3.1}$$

$S$  has the following interpretation in terms of generalised Bernoulli trials [12]. Suppose a PDF  $Q(\mathbf{x})$  is defined, and suppose that a large number  $n$  of samples  $\mathbf{x}$  are drawn independently from  $Q(\mathbf{x})$ . Defining the observed PDF of the samples as  $P(\mathbf{x})$  we obtain (see Section IV for a detailed derivation)  $\exp(nS)$  as the probability that the  $n$  trials have the observed distribution of outcomes. Thus the smaller  $\exp(nS)$  is (or, equivalently, the more negative  $S$  is) the more surprised we are that  $P(\mathbf{x})$  is genuinely the distribution of outcomes of independent samples from  $Q(\mathbf{x})$ .  $S$  may also be written as an integral

over  $\mathbf{s}$ , as follows:

$$\begin{aligned} S &= S_0 + \int d\mathbf{x} P(\mathbf{x}) \log(Q(\mathbf{x})) \\ &= S_0 + \int d\mathbf{x} P(\mathbf{x}) \log\left(\frac{p(\mathbf{s}(\mathbf{x}))}{Z(\mathbf{s}(\mathbf{x}))}\right) \\ &= S_0 + \int d\mathbf{x} P(\mathbf{x}) \int d\mathbf{s} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \log\left(\frac{p(\mathbf{s})}{Z(\mathbf{s})}\right) \\ &= S_0 + \int d\mathbf{s} p(\mathbf{s}) \log\left(\frac{p(\mathbf{s})}{Z(\mathbf{s})}\right) \end{aligned} \quad (3.2)$$


---

where  $S_0$  depends only on  $P(\mathbf{x})$ .

We may obtain the variation of  $S$  with respect to changes in  $\mathbf{s}(\mathbf{x})$  by evaluating the functional derivative  $\frac{\delta S}{\delta \mathbf{s}(\mathbf{x})}$ . This derivative depends on  $\frac{\delta p(\mathbf{s})}{\delta \mathbf{s}(\mathbf{x})}$  and  $\frac{\delta Z(\mathbf{s})}{\delta \mathbf{s}(\mathbf{x})}$ , which are given by

$$\begin{aligned} \frac{\delta p(\mathbf{s})}{\delta \mathbf{s}(\mathbf{x})} &= \int d\mathbf{y} P(\mathbf{y}) \frac{\delta}{\delta \mathbf{s}(\mathbf{x})} \delta(\mathbf{s} - \mathbf{s}(\mathbf{y})) \\ &= \int d\mathbf{y} P(\mathbf{y}) \frac{1}{2\pi} \int d\mathbf{k} \frac{\delta}{\delta \mathbf{s}(\mathbf{x})} \exp[i\mathbf{k} \cdot (\mathbf{s} - \mathbf{s}(\mathbf{y}))] \\ &= - \int d\mathbf{y} P(\mathbf{y}) \frac{1}{2\pi} \int d\mathbf{k} i\mathbf{k} \delta(\mathbf{x} - \mathbf{y}) \exp[i\mathbf{k} \cdot (\mathbf{s} - \mathbf{s}(\mathbf{y}))] \\ &= - \int d\mathbf{y} P(\mathbf{y}) \delta(\mathbf{x} - \mathbf{y}) \frac{1}{2\pi} \int d\mathbf{k} \nabla_{\mathbf{s}} \exp[i\mathbf{k} \cdot (\mathbf{s} - \mathbf{s}(\mathbf{y}))] \\ &= -P(\mathbf{x}) \nabla_{\mathbf{s}} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \end{aligned} \quad (3.3)$$

and similarly

$$\frac{\delta Z(\mathbf{s})}{\delta \mathbf{s}(\mathbf{x})} = -\nabla_{\mathbf{s}} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \quad (3.4)$$

where  $\nabla_{\mathbf{s}}$  is the gradient operator with respect to  $\mathbf{s}$ . Equation 3.3 and Equation 3.4 involve gradients of delta functions which are singular quantities. However this should not be surprising because functional differentiation itself is a singular operation. Thus we obtain from Equation 3.2

$$\begin{aligned} \frac{\delta S}{\delta \mathbf{s}(\mathbf{x})} &= \int d\mathbf{s} \frac{\delta p(\mathbf{s})}{\delta \mathbf{s}(\mathbf{x})} \log\left(\frac{p(\mathbf{s})}{Z(\mathbf{s})}\right) + \int d\mathbf{s} \frac{\delta p(\mathbf{s})}{\delta \mathbf{s}(\mathbf{x})} - \int d\mathbf{s} \frac{p(\mathbf{s})}{Z(\mathbf{s})} \frac{\delta Z(\mathbf{s})}{\delta \mathbf{s}(\mathbf{x})} \\ &= -P(\mathbf{x}) \int d\mathbf{s} \log\left(\frac{p(\mathbf{s})}{Z(\mathbf{s})}\right) \nabla_{\mathbf{s}} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) + \int d\mathbf{s} \frac{p(\mathbf{s})}{Z(\mathbf{s})} \nabla_{\mathbf{s}} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \\ &= P(\mathbf{x}) \nabla_{\mathbf{s}} \log\left(\frac{p(\mathbf{s})}{Z(\mathbf{s})}\right) \Big|_{\mathbf{s}=\mathbf{s}(\mathbf{x})} - \nabla_{\mathbf{s}} \frac{p(\mathbf{s})}{Z(\mathbf{s})} \Big|_{\mathbf{s}=\mathbf{s}(\mathbf{x})} \\ &= (P(\mathbf{x}) - Q(\mathbf{x})) \nabla_{\mathbf{s}} \log\left(\frac{p(\mathbf{s})}{Z(\mathbf{s})}\right) \Big|_{\mathbf{s}=\mathbf{s}(\mathbf{x})} \end{aligned} \quad (3.5)$$


---

where we use Equation 3.3 and Equation 3.4 and the fact that the functional derivative of the constant  $\int d\mathbf{s} p(\mathbf{s})$  is zero in the first stage of the simplification, we integrate by parts in the second stage, and we use Equation 2.6 to introduce  $Q(\mathbf{x})$  in the third stage. The surface terms which arise from the integration by parts do not contribute provided that we do not attempt to vary  $\mathbf{s}(\mathbf{x})$  on the surface. Alternatively, the surface terms do not contribute anyway if  $P(\mathbf{s}) = 0$  at those  $\mathbf{x}$  which map to the surface in  $\mathbf{s}$  space. In practical applications we use a discrete  $\mathbf{s}$  variable where  $p(\mathbf{s})$  is replaced by a histogram

with a finite total number of counts in each histogram bin, so the surface terms are guaranteed to be zero provided that we include ‘guard bins’ containing zero counts. As expected  $\frac{\delta S}{\delta \mathbf{s}(\mathbf{x})} = 0$  when  $Q(\mathbf{x}) = P(\mathbf{x})$ , in which case  $P(\mathbf{x})$  is a function of only  $\mathbf{s}(\mathbf{x})$ . We may use  $\frac{\delta S}{\delta \mathbf{s}(\mathbf{x})}$  to obtain the first-order estimate of the total change of relative entropy  $\Delta S$  due to a change  $\delta \mathbf{s}(\mathbf{x})$  in  $\mathbf{s}(\mathbf{x})$  in the form

$$\Delta S \simeq \int d\mathbf{x} \delta \mathbf{s}(\mathbf{x}) \cdot \frac{\delta S}{\delta \mathbf{s}(\mathbf{x})} \quad (3.6)$$

This type of perturbation would occur when we wish to refine an existing set of sampling functions in order to match them better to the task of preserving information about  $P(\mathbf{x})$ . Setting  $\delta\mathbf{s}(\mathbf{x}) = \varepsilon \mathbf{t}(\mathbf{x})$  in Equation 3.6 and using Equation 3.5 yields

$$\begin{aligned}\Delta S &\simeq \int d\mathbf{x} \varepsilon \mathbf{t}(\mathbf{x}) \cdot \int d\mathbf{s} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \\ &\quad \times (P(\mathbf{x}) - Q(\mathbf{x})) \nabla_{\mathbf{s}} \log \frac{p(\mathbf{s})}{Z(\mathbf{s})} \\ &= \varepsilon \int d\mathbf{x} d\mathbf{s} d\mathbf{t} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \delta(\mathbf{t} - \mathbf{t}(\mathbf{x})) \\ &\quad \times (P(\mathbf{x}) - Q(\mathbf{x})) \mathbf{t}(\mathbf{x}) \cdot \nabla_{\mathbf{s}} \log \frac{p(\mathbf{s})}{Z(\mathbf{s})} \quad (3.7)\end{aligned}$$

where we introduce delta functions in order to write  $\Delta S$  in a form which is easier to manipulate. Integrating Equation 3.7 with respect to  $\mathbf{x}$  yields

$$\frac{\partial S}{\partial \varepsilon} = \int d\mathbf{s} dt (p(\mathbf{s}, \mathbf{t}) - q(\mathbf{s}, \mathbf{t})) \mathbf{t} \cdot \nabla_{\mathbf{s}} \log \frac{p(\mathbf{s})}{Z(\mathbf{s})} \quad (3.8)$$

where we use  $\Delta S \simeq \varepsilon \frac{\partial S}{\partial \varepsilon}$ , and we define

$$\begin{aligned}p(\mathbf{s}, \mathbf{t}) &\equiv \int d\mathbf{x} P(\mathbf{x}) \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \delta(\mathbf{t} - \mathbf{t}(\mathbf{x})) \\ q(\mathbf{s}, \mathbf{t}) &\equiv \int d\mathbf{x} Q(\mathbf{x}) \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \delta(\mathbf{t} - \mathbf{t}(\mathbf{x})) \quad (3.9)\end{aligned}$$

This expression for  $p(\mathbf{s}, \mathbf{t})$  should be compared with Equation 2.1.  $p(\mathbf{s}, \mathbf{t})$  and  $q(\mathbf{s}, \mathbf{t})$  are the compressed versions of  $P(\mathbf{x})$  and  $Q(\mathbf{x})$  when the vector of sampling functions  $(\mathbf{s}(\mathbf{x}), \mathbf{t}(\mathbf{x}))$  is used. However, note that  $Q(\mathbf{x})$  is determined by measurements of  $\mathbf{s}(\mathbf{x})$  alone (see Equation 2.6).

Alternatively we may calculate the entropy change  $\Delta S$  which accompanies a transformation  $\mathbf{s}(\mathbf{x}) \rightarrow (\mathbf{s}(\mathbf{x}), \mathbf{t}(\mathbf{x}))$ . This is different from the above transformation because now we do not merely perturb  $\mathbf{s}(\mathbf{x})$ , rather we introduce some new dimensions (via  $\mathbf{t}(\mathbf{x})$ ) into the space of sampling functions of  $\mathbf{x}$ . This type of change would occur when we explore the effect of measuring some novel statistics  $\mathbf{t}(\mathbf{x})$  in addition to those statistics  $\mathbf{s}(\mathbf{x})$  which we have already measured. From Equation 3.2 we obtain

---


$$\begin{aligned}\Delta S &= \int d\mathbf{s} dt p(\mathbf{s}, \mathbf{t}) \log \left( \frac{p(\mathbf{s}, \mathbf{t})}{Z(\mathbf{s}, \mathbf{t})} \right) - \int d\mathbf{s} p(\mathbf{s}) \log \left( \frac{p(\mathbf{s})}{Z(\mathbf{s})} \right) \\ &= \int d\mathbf{s} dt p(\mathbf{s}, \mathbf{t}) \log \left( \frac{Z(\mathbf{s})}{p(\mathbf{s})} \frac{p(\mathbf{s}, \mathbf{t})}{Z(\mathbf{s}, \mathbf{t})} \right) \\ &\geq 0 \quad (3.10)\end{aligned}$$


---

where

$$Z(\mathbf{s}, \mathbf{t}) \equiv \int d\mathbf{x} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \delta(\mathbf{t} - \mathbf{t}(\mathbf{x})) \quad (3.11)$$

which should be compared with Equation 2.7. We may simplify Equation 3.10 further by combining Equation 2.6 and Equation 3.9 to obtain

$$q(\mathbf{s}, \mathbf{t}) = \frac{p(\mathbf{s}) Z(\mathbf{s}, \mathbf{t})}{Z(\mathbf{s})} \quad (3.12)$$

whence

$$\Delta S = \int d\mathbf{s} dt p(\mathbf{s}, \mathbf{t}) \log \left( \frac{p(\mathbf{s}, \mathbf{t})}{q(\mathbf{s}, \mathbf{t})} \right) \quad (3.13)$$

Equation 3.8 and Equation 3.13 give  $\Delta S$  for dimensionality preserving and dimensionality increasing changes to the statistic  $\mathbf{s}(\mathbf{x})$  respectively: these are the basic relations which we can use to find a suitable  $\mathbf{s}(\mathbf{x})$ . In both cases  $\Delta S$  depends on a comparison of  $p(\mathbf{s}, \mathbf{t})$  which derives from  $P(\mathbf{x})$ , and  $q(\mathbf{s}, \mathbf{t})$  which derives from  $Q(\mathbf{x})$ .

#### IV. NUMERICAL METHODS

In this section we shall consider the second type of variation of  $\mathbf{s}(\mathbf{x})$  (i.e. dimensionality increasing transformations), for which  $\Delta S$  is expressed in Equation 3.13. There are two basic problems to contend with. Firstly  $q(\mathbf{s}, \mathbf{t})$  must be estimated from  $p(\mathbf{s})$  by sampling from  $Q(\mathbf{x})$  and transforming the  $\mathbf{x}$  samples into the corresponding  $(\mathbf{s}, \mathbf{t})$  samples. Also the relative entropy expression in Equation 3.13 should be reformulated using the language of generalised Bernoulli trials [12] to take account of the small number of observations of  $(\mathbf{s}, \mathbf{t})$  which are available for estimating  $p(\mathbf{s}, \mathbf{t})$ . Throughout this section we

shall assume that  $\mathbf{s}$  and  $\mathbf{t}$  have been discretised into cells in  $p(\mathbf{s}, \mathbf{t})$  space, so all PDFs become histograms.

The expression for  $Q(\mathbf{x})$  which is given in Equation 2.6 involves a  $Z(\mathbf{s}(\mathbf{x}))$  factor which must either be calculated explicitly or estimated numerically. It is possible to obtain  $Z(\mathbf{s})$  explicitly in simple cases, but we concentrate on obtaining a numerical estimate because of the greater flexibility which such a scheme affords. In order to achieve this we take advantage of the identification of  $Z(\mathbf{s})$  with a PDF (see Equation 2.7), so we use a Monte Carlo simulation to generate samples from  $Z(\mathbf{s})$ . An estimate of  $Z(\mathbf{s})$  must count the number of states  $\mathbf{x}$  which map to each cell in  $\mathbf{s}$  space ( $\mathbf{s}$  cell) modulo an undetermined constant factor which does not affect Monte Carlo simulations. For our purposes we need to estimate  $Z(\mathbf{s})$  only for those  $\mathbf{s}$  cells (the occupied  $\mathbf{s}$  cells) which are occupied by non-zero probability  $p(\mathbf{s})$ . Such an estimate is obtained by performing a Monte Carlo sampling of  $\mathbf{x}$  subject only to the constraint that  $\mathbf{s}(\mathbf{x})$  lies in an occupied  $\mathbf{s}$  cell. The randomness of this update scheme ensures that the PDF over the occupied regions of  $\mathbf{x}$  is uniform as required, and the constraint ensures that we do not waste Monte Carlo updates in estimating  $Z(\mathbf{s})$  for  $\mathbf{s}$  cells which have no  $p(\mathbf{s})$  counts. The frequency with which an  $\mathbf{s}$  cell is visited is then proportional to  $Z(\mathbf{s})$ .

We implicitly introduce a prior measure over  $\mathbf{x}$  in order to implement this sampling scheme. For instance, suppose  $\mathbf{x}$  is a vector of non-negative integer-valued quantities, each of which is represented by a binary word, then each possible bit pattern has an equal *a priori* probability. More generally, the prior measure has the form of equal *a priori* probabilities for each cell of the discretised  $\mathbf{x}$  space with which we work in practice. Since the choice of discretisation is a matter of taste, the prior measure is an arbitrary function in our method. This is not a disadvantage because the apparent arbitrariness of the prior measure is a manifestation of its flexibility for incorporating whatever coding scheme (i.e. discretisation) is used to represent the original data.

As usual with Monte Carlo simulations it is necessary for the sequence of sampled  $\mathbf{x}$  states to be ergodic. This requires that the occupied  $\mathbf{s}$  cells must be connected by some path(s) in  $\mathbf{x}$  space which can be generated by the Monte Carlo update scheme chosen. The sort of situation which must be avoided is using  $\mathbf{s}$  cells which are very small (and so contain very few counts), because the constrained Monte Carlo update scheme might then be unable to find any suitable update paths to reach some  $\mathbf{s}$  cells. Whatever Monte Carlo scheme is used, the resulting estimate for  $Z(\mathbf{s})$  is incorporated into a further Monte Carlo simulation which draws samples from  $Q(\mathbf{x})$  according to an update scheme based on Equation 2.6. The sequence of  $\mathbf{x}$  values so obtained is used to estimate  $q(\mathbf{s}, \mathbf{t})$  as required.

A more careful interpretation of  $\Delta S$  in Equation 3.13 follows from a derivation of relative entropy from first principles. Thus define a discrete random variable  $i$  ( $i = 1, 2, \dots, r$ ) which has a state probability  $q_i$  ( $\sum q_i = 1$ ).

Now suppose that we draw independently  $n$  samples of this random variable (i.e.  $n$  generalised Bernoulli trials [12]), and the outcome is  $n p_i$  samples of state  $i$  ( $\sum p_i = 1$ ) in any order. The probability  $P(\mathbf{p}, \mathbf{q}; n)$  of this outcome is given by

$$P(\mathbf{p}, \mathbf{q}; n) = \frac{n!}{(n p_1)! \cdots (n p_r)!} (q_1)^{n p_1} \cdots (q_r)^{n p_r} \quad (4.1)$$

Using Stirling's approximation ( $\log(x!) \sim x \log x - x$ ) we obtain

$$\log P(\mathbf{p}, \mathbf{q}; n) \simeq -n \sum_{i=1}^r p_i \log \left( \frac{p_i}{q_i} \right) \quad (4.2)$$

which is  $n$  times a relative entropy: this is, in fact, the origin of relative entropy. The familiar expression for relative entropy is accurate only in the limit of a large number of generalised Bernoulli trials, and we should revert to using Equation 4.1 directly when the number of trials is small. We may now use Equation 4.1 to rewrite Equation 3.13 to take into account the effect of the limited number  $n$  of  $(\mathbf{s}, \mathbf{t})$  samples which are available for estimating  $p(\mathbf{s}, \mathbf{t})$ . Thus replace the continuous notation  $p(\mathbf{s}, \mathbf{t})$  and  $q(\mathbf{s}, \mathbf{t})$  by the discrete notation  $p_{s,t}$  and  $q_{s,t}$ , where the index pair  $(s, t)$  labels an  $(s, t)$  cell, to give

$$\Delta S \simeq -\frac{1}{n} \log(n!) + \frac{1}{n} \sum_{s,t} (\log(n p_{s,t})! - n p_{s,t} \log(q_{s,t})) \quad (4.3)$$

Equation 4.3 becomes Equation 3.13 as  $n \rightarrow \infty$ . It is important to note that we have not merely replaced integrations in Equation 3.13 with their corresponding summations to obtain Equation 4.3.

Combining the Monte Carlo scheme for estimating  $q(\mathbf{s}, \mathbf{t})$  (and hence  $q_{s,t}$ ) with Equation 4.3 provides us with a practical scheme for assessing the information content (for the purposes of reconstructing  $P(\mathbf{x})$ ) of a novel set of statistics  $\mathbf{t}(\mathbf{x})$ , and hence with a means of building a set of statistics which is matched to  $P(\mathbf{x})$ .

## V. THE APPLICATION TO TEXTURE ANALYSIS

The problem of evaluating  $\Delta S$  in Equation 3.13 is simplified if the variance of  $(\mathbf{s}, \mathbf{t})$  is small compared with its mean (i.e. small relative variance), so that  $p(\mathbf{s}, \mathbf{t})$  may be approximated by a product of delta functions

$$p(\mathbf{s}, \mathbf{t}) \simeq \delta(\mathbf{s} - \mathbf{s}_0) \delta(\mathbf{t} - \mathbf{t}_0) \quad (5.1)$$

where  $\mathbf{s}_0$  and  $\mathbf{t}_0$  are the mean values of  $\mathbf{s}$  and  $\mathbf{t}$  respectively. In the discrete language of Section IV all the observations then fall into a single  $(\mathbf{s}, \mathbf{t})$  cell. For instance, in image processing applications  $\mathbf{s}(\mathbf{x})$  might be the spatial average (over an image) of some function of limited spatial support, in which case the relative variance of  $\mathbf{s}$  can be very small for large images. We do not then

need to estimate  $Z(\mathbf{s})$  as a precursor to using a Monte Carlo simulation for generating samples from  $Q(\mathbf{x})$ , because  $p(\mathbf{s})$  forces  $\mathbf{s}$  to lie in a single  $\mathbf{s}$  cell. The problem of estimating  $q(\mathbf{s}, \mathbf{t})$  thus reduces to performing a Monte Carlo sampling from  $Q(\mathbf{x})$  (constrained so that  $\mathbf{s} = \mathbf{s}_0$ ), and mapping each  $\mathbf{x}$  sample into an  $(\mathbf{s}_0, \mathbf{t})$  cell in order to build up a histogram representation  $q_{s,t}$  of  $q(\mathbf{s}, \mathbf{t})$ .

Now suppose that  $q(\mathbf{s}, \mathbf{t})$  also is well approximated by a product of delta functions

$$q(\mathbf{s}, \mathbf{t}) \simeq \delta(\mathbf{s} - \mathbf{s}_0) \delta(\mathbf{t} - \mathbf{t}'_0) \quad (5.2)$$

where in general  $\mathbf{t}'_0 \neq \mathbf{t}_0$ . When  $\mathbf{t}_0 = \mathbf{t}'_0$   $\Delta S = 0$ , so the statistic  $\mathbf{t}(\mathbf{x})$  measures no information about  $P(\mathbf{x})$  which  $\mathbf{s}(\mathbf{x})$  has not already measured. When  $\mathbf{t}_0 \neq \mathbf{t}'_0$   $\Delta S$  diverges to infinity because  $p(\mathbf{s}, \mathbf{t})$  and  $q(\mathbf{s}, \mathbf{t})$  do not overlap at all. This divergence is an artefact of our approximation of the PDFs as delta functions (or Kronecker deltas).  $\Delta S$  is a measure of the separation of the high probability regions of  $p(\mathbf{s}, \mathbf{t})$  and  $q(\mathbf{s}, \mathbf{t})$ , so we can use the difference  $|\mathbf{t}_0 - \mathbf{t}'_0|$  itself as a crude measure of  $\Delta S$ . The crudeness of this measure does not matter if we wish to obtain only an approximate idea of the useful information content of  $\mathbf{t}$  (given that  $\mathbf{s}$  is already known).

The above scheme for designing sampling functions (or statistics) is highly effective when applied to the problem of texture analysis in image processing applications [6–8]. Textured image data are well approximated by a statistically homogeneous PDF

$$P(L\mathbf{x}) = P(\mathbf{x}) \quad (5.3)$$

where  $L$  is any translation operator. We may impose this symmetry on the maximum entropy reconstruction  $Q(\mathbf{x})$  by choosing  $\mathbf{s}(\mathbf{x})$  to have the form

$$\mathbf{s}(\mathbf{x}) = \frac{1}{N} \sum_L \boldsymbol{\sigma}(L\mathbf{x}) \quad (5.4)$$

where  $N$  is the number of terms in the sum over  $L$ , and  $\boldsymbol{\sigma}(\mathbf{x})$  is usually chosen to be a function of limited support on  $\mathbf{x}$ . It is important to note that the form of  $\mathbf{s}(\mathbf{x})$  used in Equation 5.4 is more restrictive than is necessary to guarantee translation invariance of  $P(\mathbf{x})$ . However, the problem of designing more general translation invariant sampling functions is very subtle, so we shall not attempt to study it here. As required, the translation invariance  $\mathbf{s}(L\mathbf{x}) = \mathbf{s}(\mathbf{x})$  in Equation 5.4 together with the expression for  $Q(\mathbf{x})$  in Equation 2.6 guarantees that

$$Q(L\mathbf{x}) = Q(\mathbf{x}) \quad (5.5)$$

We may estimate the relative variance of  $s_i(\mathbf{x})$  (and analogously  $t_i(\mathbf{x})$ ) as

$$\frac{\text{Var}(s_i)}{\langle s_i \rangle^2} \propto \frac{1}{N_{\text{eff},i}} \quad (5.6)$$

where  $N_{\text{eff},i}$  is the effective number of independent measurements of  $\sigma_i(\mathbf{x})$  in the sum in Equation 5.4, which we

shall assume to be a large number for all  $i$ . We may then use Equation 5.1 to approximate  $p(\mathbf{s}, \mathbf{t})$ , and the estimation of  $\Delta S$  proceeds as above. By repeatedly applying this approach of estimating  $\Delta S$  in order to determine whether to accept or reject  $\mathbf{t}(\mathbf{x})$  we may build up a set of statistics for  $P(\mathbf{x})$ .

Implicit in this scheme is the need to select candidate statistics  $\mathbf{t}(\mathbf{x})$  and also there is an implicit threshold value of  $\Delta S$  above which  $\mathbf{t}(\mathbf{x})$  is accepted; both of these are highly application dependent. A good rule of thumb for selecting a tentative  $\mathbf{t}(\mathbf{x})$  is to compare visually the data sets  $\mathbf{x}$  sampled from  $Q(\mathbf{x})$  with the original data sets sampled from  $P(\mathbf{x})$ . Any obvious differences should suggest what types of  $\mathbf{t}(\mathbf{x})$  might best discriminate (i.e. have a large  $\Delta S$ ) between these data sets. For instance, if the original images (which define  $P(\mathbf{x})$ ) contain bright lines and the simulated images (samples from  $Q(\mathbf{x})$ ) do not, then an excellent candidate  $\mathbf{t}(\mathbf{x})$  is any function which has a strong dependence on the presence or absence of bright lines in the data.

Once we have constructed a set of statistics  $\mathbf{s}(\mathbf{x})$  to characterise the texture in terms of its PDF  $P(\mathbf{x})$  we may attempt to construct an MRF texture model [9]. We make the connection between these two approaches to texture analysis by noting that we approximate  $p(\mathbf{s})$  by a delta function  $\delta(\mathbf{s} - \mathbf{s}_0)$ , so we implicitly retain information about only the mean value  $\mathbf{s}_0$  of  $\mathbf{s}$ . Within this approximation we may rederive the maximum entropy form of  $Q(\mathbf{x})$  from first principles by defining the cost function  $C$  as [3]

$$C \equiv - \int d\mathbf{x} Q(\mathbf{x}) \log(Q(\mathbf{x})) - \boldsymbol{\lambda} \cdot \left( \int d\mathbf{x} Q(\mathbf{x}) \mathbf{s}(\mathbf{x}) - \int d\mathbf{x} P(\mathbf{x}) \mathbf{s}(\mathbf{x}) \right) \quad (5.7)$$

Functionally differentiating with respect to  $Q(\mathbf{x})$  yields the familiar  $Q(\mathbf{x})$  of exponential type

$$Q(\mathbf{x}) = \frac{1}{Z} \exp(-\boldsymbol{\mu} \cdot \mathbf{s}(\mathbf{x})) \quad (5.8)$$

where  $Z$  is a normalisation constant (partition function) and  $\boldsymbol{\mu}$  must be adjusted to satisfy the constraint on the mean of  $\mathbf{s}$ . This form of  $Q(\mathbf{x})$  is a Gibbs distribution, which is equivalent to  $\mathbf{x}$  being an MRF with potentials  $\mathbf{s}(\mathbf{x})$  (or local potentials  $\boldsymbol{\sigma}(\mathbf{x})$ ).

This scheme for extracting an MRF description fails when an implicit assumption in the Lagrange multiplier method is violated. The method assumes that there exists at least one  $Q(\mathbf{x})$  for which the functional derivative of the entropy lies in the function subspace spanned by the functional derivatives of the constraint functions; this is expressed directly in the form of  $C$  in Equation 5.7. When this is not the case there is no solution for  $\boldsymbol{\mu}$ . This point is not usually emphasised, although it is fundamentally important.

Our method of building statistics (which leads ultimately to a sufficient set of statistics) improves the

chance that a solution for  $\mu$  can indeed be found, because we design  $s(\mathbf{x})$  so that it is matched to  $P(\mathbf{x})$ . We shall therefore proceed as if an MRF solution can always be found. A convenient way of adjusting  $\mu$  is provided by the relative entropy which is defined in Equation 3.1. Substituting  $Q(\mathbf{x})$  in Equation 5.8 into Equation 3.1 and differentiating with respect to  $\mu_k$  yields

$$\begin{aligned}\frac{\partial S}{\partial \mu_k} &= \frac{\partial}{\partial \mu_k} \int d\mathbf{x} P(\mathbf{x}) \log(Q(\mathbf{x})) \\ &= - \int d\mathbf{x} P(\mathbf{x}) \left( \frac{\partial \log(Z)}{\partial \mu_k} + \frac{\partial \sum \mu_m s_m(\mathbf{x})}{\partial \mu_k} \right) \\ &= \int d\mathbf{x} Q(\mathbf{x}) s_k(\mathbf{x}) - \int d\mathbf{x} P(\mathbf{x}) s_k(\mathbf{x}) \quad (5.9)\end{aligned}$$

The correctly constrained solution to the maximum entropy problem in Equation 5.7 corresponds to  $\frac{\partial S}{\partial \mu_k} = 0$  in Equation 5.9. Thus we can recover  $Q(\mathbf{x})$  (from knowledge of only the mean of  $s$ ) directly by maximising  $S$  in the subspace of functions  $Q(\mathbf{x})$  of exponential type

given in Equation 5.8. This is essentially the Boltzmann machine learning algorithm of neural network theory [1].

## VI. CONCLUSIONS

We have used information theoretic principles to derive a scheme for building sampling functions for data compression when only the PDF  $P(\mathbf{x})$  over data sets needs to be reconstructed. An advantage of our adaptive approach is that the resulting set of sampling functions has a low information redundancy. The application of this method to texture analysis results in the design of an MRF texture model which is specifically tuned to the observed statistical texture structure. This method could therefore find applications in areas of image processing which already use *ad hoc* MRF models.

The theory which we have presented extends and unifies earlier work by us [6–9], and it prepares the ground for the analysis of more general sampling schemes.

- 
- [1] D H Ackley, G E Hinton, and T J Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Science **9** (1985), no. 2, 147–169.
  - [2] M Bertero and E R Pike, *Resolution in diffraction limited imaging, a singular value analysis I. The case of coherent illumination*, Optica Acta **29** (1982), no. 6, 727–746.
  - [3] E T Jaynes, *Prior probabilities*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 3, 227–241.
  - [4] S P Luttrell, *Prior knowledge and object reconstruction using the best linear estimate technique*, Optica Acta **32** (1985), no. 6, 703–716.
  - [5] ———, *The use of transinformation in the design of data sampling schemes for inverse problems*, Inverse Problems **1** (1985), no. 3, 199–218.
  - [6] ———, *Markov random fields: a strategy for clutter modelling*, Proceedings of AGARD conference on scattering and propagation in random media (Rome), AGARD, 1987, pp. 7.1–7.8.
  - [7] ———, *Radar 87*, ch. Designing Markov random field structures for clutter modelling, pp. 222–226, IEE, London, 1987.
  - [8] ———, *The use of Markov random field models in sampling scheme design*, Proceedings of SPIE international symposium on inverse problems in optics (New York), SPIE, 1987.
  - [9] ———, *The use of Markov random field models to derive sampling schemes for inverse texture problems*, Inverse Problems **3** (1987), no. 2, 289–300.
  - [10] S P Luttrell and C J Oliver, *Prior knowledge in synthetic aperture radar processing*, Journal of Physics D: Applied Physics **19** (1986), no. 3, 333–356.
  - [11] R J McEliece, *The theory of information and coding*, Addison-Wesley, Reading, 1977.
  - [12] A Papoulis, *Probability, random variables and stochastic processes*, McGraw-Hill, New York, 1984.
  - [13] S Watts and K D Ward, *Spatial correlation in  $K$ -distributed sea clutter*, IEE Proceedings F: Radar and Signal Processing **134** (1987), no. 6, 526–532.

# The Inverse Cross Section Problem for Complex Data \*

S P Luttrell

Royal Signals and Radar Establishment, Great Malvern, Worcs WR14 3PS, UK

Using a Gaussian scattering model we present a Bayesian analysis of the recovery of a scattering cross section from complex data samples (the inverse cross section problem). We derive the likelihood function of the cross section given complex data, and its functional derivatives with respect to cross section. The entire analysis is expressed using the same weighted singular value decomposition (SVD) of the imaging system that is used when solving the related inverse object field problem in a weighted space.

## I. INTRODUCTION

There has been much interest in the inverse problem of reconstructing a complex object field from complex data samples [5], specifically in the field of image processing. This involves deconvolving the effect of an instrument function subject to regularisation constraints which control the effects of low rank imaging operators and data noise. In information theoretic terms the most rigorous solution to this problem is the weighted SVD analysis of complex data [8, 9], and related work on the recovery of exponential relaxation rates [1]. However, existing information theoretic analyses deal only with one- and two-point statistics (i.e. the probability density function (PDF) models are Gaussian); more sophisticated analyses must await further research.

In some cases the inverse problem is not to recover the complex object field but rather to recover the source of this field, which is usually expressed as a scattering cross section. The scattering cross section is directly related to the properties of the object itself, whereas the object field is only indirectly related because of the possibly stochastic nature of the scattering process. A practical example of this inverse problem is the recovery of a cross section from complex image data, such as a coherent radar image or a coherent acoustic image. This has received some attention in the literature [2, 3, 7] where the related problem of smoothing a speckled image has been considered. The purpose of our approach is to derive reconstruction algorithms from a fundamental starting point (in this case the axioms of Bayesian inference), and thus to reduce to a minimum the use of *ad hoc* procedures.

We shall use a Gaussian stochastic model of the scattering process and a linear model of the imaging system together with a Bayesian analysis of the inverse problem to perform a detailed analysis of the recovery of a scattering cross section (rather than an object field as studied in [8, 9]). We find that the inverse cross section and inverse object field problems are intimately related insofar as all the data dependence of the former problem

may be derived from the solution to the latter problem (as provided by a suitable weighted SVD analysis).

In Section II we present the Gaussian scattering model and the linear imaging model, and in Section III we briefly review the use of Bayesian techniques for posing inverse problems in terms of a likelihood function and an *a priori* PDF. In Section IV we present the weighted SVD method of decomposing the inverse problem into manageable components, which we then use to derive the first derivative of the likelihood function in Section V, and higher derivatives in Section VI. The principal new results are contained in Section V and Section VI.

## II. THE MODEL

The process of forming a coherent image has three basic stages: illumination of an object by coherent radiation, scattering of the radiation by the object, and coherent image formation at a receiver. The properties of the first and third stages are usually well understood, but the second stage involves complicated scattering mechanisms. We shall not attempt to model the scattering process physically; rather we shall adopt a point-process model. This is justified empirically because it leads to a good fit to the data statistics. The purpose of the point processes is merely to emulate the effect of whatever physical processes are actually occurring, but one can identify the local density of scatterers (point processes) with the local cross section. The object field  $f$  is then given by

$$f_i = \sum_{|j-i| < D} \sqrt{\delta\sigma_j} e^{i\phi_j} \quad (2.1)$$

where  $\delta\sigma_j$  has the dimensions of a scattering cross section and is associated with component  $j$  of the object,  $\phi_j$  is a phase which is random because of the unpredictable scattering phaseshift and unpredictable path length to the field point  $i$ , and  $D$  restricts the sum to that part of the object which contributes to the field at  $i$ . We shall assume that the effective number of terms in Equation 2.1 is large enough that the central limit theorem leads to Gaussian statistics for  $f_i$ . Physically this places a lower bound on the permitted size of  $D$ , because there must be sufficient scatterers present to permit use of the central limit theorem. Consequently our Gaussian model must

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This paper appeared in *Inverse Problems*, 1989, vol. 5, no. 1, pp. 35-50. Received 20 June 1988, in final form 4 October 1988. © Controller, Her Majesty's Stationery Office, London, 1988.

break down at some small length scale commensurate with the typical scatterer separation. Introducing the PDF  $P(f_i)$  [8, 9] allows us to replace Equation 2.1 by a statistical description of  $f_i$

$$P(f_i) \equiv \frac{1}{\pi \sigma_i} \exp\left(-\frac{|f_i|^2}{\sigma_i}\right) \quad (2.2)$$

where  $\sigma_i$  is related to the  $\delta\sigma_j$  by

$$\sigma_i = \sum_{|j-i| < D} \delta\sigma_j \quad (2.3)$$

which may be interpreted as the local cross section (per unit area of the object). In general there are non-trivial higher-order correlations (i.e. two-point and higher) amongst the  $f_i$  because of overlap of the source of contributions to neighbouring field points. Their associated correlation length is  $\mathcal{O}(D)$ , so we may ignore them provided that we do not enquire about the joint statistical properties of the  $f_i$  on a length scale less than  $\mathcal{O}(D)$ .

Subject to these restrictions the joint PDF of the  $f_i$  may be written as

$$\begin{aligned} P[f] &= \prod_i \frac{1}{\pi \sigma_i} \exp\left(-\frac{|f_i|^2}{\sigma_i}\right) \\ &= \frac{1}{\det(\pi \sigma)} \exp(-f^\dagger \sigma^{-1} f) \end{aligned} \quad (2.4)$$

where we have introduced the vector  $f \equiv (f_1, f_2, \dots)^T$  and its Hermitian conjugate  $f^\dagger \equiv f^{T*} = (f_1^*, f_2^*, \dots)$ , and we have defined the matrix  $\sigma$  as

$$\sigma \equiv \text{diag}(\sigma_1, \sigma_2, \dots) \quad (2.5)$$

We have introduced this rather compact notation because it makes our calculations simple to follow. We may also interchange freely between discrete and continuous notation because the matrix-vector formalism which we use treats these two cases on an equal footing. Furthermore, we use an explicit PDF representation rather than just the underlying covariance matrices to emphasise the Bayesian nature of our calculations, and to indicate the full structure of the derivations should one wish to use non-Gaussian PDF in the future.

We shall assume that the imaging process itself may be modelled as a complex linear transformation acting on  $f$  [8, 9]. Ideally we should also introduce a model describing the various noise processes which accompany image formation such as receiver noise, quantisation noise (in the analogue-to-digital conversion step, if any), etc. However, we shall not attempt to model such noise physically; rather we shall introduce a representative additive noise term into the complex linear transformation. Thus we shall describe the imaging process by

$$g_a = \sum_i T_{a,i} f_i + n_a \quad (2.6)$$

## The Inverse Cross Section Problem for Complex Data

or more compactly by

$$g = T f + n \quad (2.7)$$

The index  $i$  labels position in object space ( $i$  is usually a continuous index), and the index  $a$  labels position in image space ( $a$  is usually a discrete index ranging over image samples).  $g_a$  is the  $a$ th sample value of the complex image  $g$ ,  $T_{a,i}$  is the component of the linear imaging operator which couples  $f_i$  to  $g_a$ , and  $n_a$  is the additive noise associated with  $g_a$ . We shall model  $n$  as a complex Gaussian process described by the PDF [8, 9]

$$P[n] \equiv \frac{1}{\det(\pi \mathbf{N})} \exp(-n^\dagger \mathbf{N}^{-1} n) \quad (2.8)$$

where the covariance matrix  $\mathbf{N}$  is chosen to approximate the two-point statistics of the true noise. We shall assume that the noise is uncorrelated, so we may set

$$\mathbf{N} \equiv \nu \mathbf{I} \quad (2.9)$$

where  $\mathbf{I}$  is the unit matrix and  $\nu$  is the noise energy per image sample: this describes uncorrelated Gaussian noise with  $\langle |n_a|^2 \rangle = \nu$ . Note that this white noise assumption is made only for convenience: the calculations may easily be extended to the case of non-diagonal  $\mathbf{N}$ .

The principal limitation of this model, insofar as the inverse cross section problem is concerned, is the assumption of Gaussian scattering statistics. The central limit theorem conditions are not satisfied at field points in the vicinity of highly localised (i.e. much less than a resolution cell in extent) strongly scattering objects, because there would then be a dominant contribution to the overall object field which would reduce the effective number of contributing scatterers. However, if *a priori* we know only that the ensemble averaged  $|f_i|^2$  is proportional to a quantity that we denote ‘cross section’, then information theory informs us that the PDF  $P[f]$  which correctly describes our state of knowledge is obtained by maximising the entropy of  $P[f]$  subject to the constraints  $\langle |f_i|^2 \rangle = \sigma_i$  [6]: this leads directly to Equation 2.2. This is true even if we suspect the presence of additional complications in the  $\sigma \rightarrow f$  mapping, but are unable (through ignorance) to model it by introducing prior knowledge of other statistics of  $|f_i|^2$  such as  $\langle |f_i|^{2k} \rangle$  or  $\langle |f_i|^{2k} |f_j|^{2l} \rangle$ , etc. We do not attempt to use such additional constraints here, so we restrict ourselves to using  $\langle |f_i|^2 \rangle = \sigma_i$  alone.

## III. BAYESIAN INFERENCE

We wish to solve the inverse problem of recovering a scattering cross section  $\sigma$  from noisy complex image data  $g$ . This problem is ill posed because the full imaging process has two stages which destroy information about  $\sigma$ : the scattering process itself is stochastic, and usually the imaging operator  $T$  is not invertible. The problem of illposedness in the related problem of recovering an object field has been studied in [10].

We shall formalise the solution to the inverse cross section problem by first stating in Bayesian terms what information we possess about the true  $\sigma$ . Thus we shall derive a PDF (or measure) over the space of possible  $\sigma$  which expresses the probability that each  $\sigma$  is the true  $\sigma$ . Such a PDF depends on the information present in the data  $g$ , and any prior source of information  $X$  that we might have. The causal relationship between  $X$ ,  $\sigma$  and  $g$  may be represented as a Markov chain  $X \rightarrow \sigma \rightarrow g$ , so we may derive (using Bayes' theorem)

$$\begin{aligned} P[\sigma|g, X] &= \frac{P[g, X|\sigma] P[\sigma]}{P[g, X]} \\ &= \frac{P[g|X, \sigma] P[X|\sigma] P[\sigma]}{P[g|X] P[X]} \\ &= \frac{P[g|\sigma] P[\sigma|X]}{P[g|X]} \end{aligned} \quad (3.1)$$

where we have used  $P[g|X, \sigma] = P[g|\sigma]$  which follows from the Markov chain property. In the final expression for  $P[\sigma|g, X]$  the PDF  $P[\sigma|X]$  plays the role of prior knowledge about  $\sigma$  and the  $\frac{P[g|\sigma]}{P[g|X]}$  factor converts this into an *a posteriori* PDF  $P[\sigma|g, X]$  which combines the effect of both information sources  $g$  and  $X$ .

Note that the  $\sigma$  dependence of  $P[\sigma|g, X]$  in Equation 3.1 is contained entirely in the numerator, so we may ignore the denominator factor  $P[g|X]$  when solving the inverse cross section problem.  $P[g|\sigma]$  is called the likelihood function because it specifies the relative likelihood that each  $\sigma$  is the true  $\sigma$  independently of the prior knowledge  $P[\sigma|X]$ .

There are two terms,  $P[g|\sigma]$  and  $P[\sigma|X]$ , to consider. First, let us construct  $P[g|\sigma]$  using the model of image formation that we presented in Section II. We have used Gaussian PDF throughout, so we may extract  $P[g|\sigma]$  without performing a detailed calculation. The covariance matrix of  $f$  is a which transforms under  $T$  to yield the noiseless image covariance thus:

$$\sigma \rightarrow T \sigma T^\dagger \quad (3.2)$$

Gaussian noise with covariance  $\mathbf{N}$  is then added to produce the full image covariance  $\mathbf{M}$  defined by

$$\mathbf{M} \equiv T \sigma T^\dagger + \mathbf{N} \quad (3.3)$$

where we have used the summation property of covariances of convolutions of independent Gaussian processes. So the Gaussian PDF  $P[g|\sigma]$  is given by

$$P[g|\sigma] \equiv \frac{1}{\det(\pi \mathbf{M})} \exp(-g^\dagger \mathbf{M}^{-1} g) \quad (3.4)$$

The positive definite noise covariance  $\mathbf{N}$  combines with the positive semidefinite covariance  $T \sigma T^\dagger$  to produce a positive definite (i.e. non-singular) covariance  $\mathbf{M}$ : in effect, the admission that the data is corrupted by noise ensures that  $P[g|\sigma]$  is nonsingular, which then regularises the Bayesian reconstruction procedure in Equation 3.1.

The other term which is required to construct  $P[\sigma|g, X]$  is  $P[\sigma|X]$ . However, our goal in this paper is to analyse the data dependence of  $P[\sigma|g, X]$ , for which a study of  $P[g|\sigma]$  alone suffices. In Section V and Section VI we shall derive expressions for the derivatives of  $P[g|\sigma]$  with respect to  $\sigma$  which may be used together with  $P[\sigma|X]$  in a maximum *a posteriori* probability (MAP) reconstruction scheme. For instance,  $P[g|\sigma]$  might be specified indirectly as a set of constraints on  $\sigma$  such as:  $\sigma$  must be of the form  $\sigma = (\sigma_1, \sigma_2, \sigma_3, \dots)$  where each group of components  $\sigma_i$  specifies a contiguous spatial region of object ‘pixels’ having the same cross section and having a boundary which obeys some smoothness criterion. This is precisely the type of prior knowledge which is used in simple segmentation algorithms. Alternatively,  $P[\sigma|X]$  might be specified indirectly as a support constraint which permits  $\sigma_i \neq 0$  within some restricted spatial domain. This is used in some super-resolution algorithms. The Bayesian approach is flexible, because all constraints on the permitted reconstruction may be recast in the language of a prior PDF  $P[\sigma|X]$ . In this paper we shall restrict our attention to the  $P[g|\sigma]$  component of the posterior distribution, because the effect of  $P[\sigma|X]$  may be incorporated afterwards into whatever reconstruction algorithm is used.

The Bayesian approach to solving inverse problems (i.e. Bayesian inference) has both its advantages and its disadvantages. On one hand it provides an information theoretically consistent way of fusing several sources of information (see Equation 3.1), but on the other hand its strict application can lead to intractable PDF calculations. Fortunately within our Gaussian scattering model and linear imaging model the analysis of the inverse cross section problem is completely solvable within the Bayesian framework. More generally, when the complexity of the calculations demands it, one is forced to use the Bayesian approach merely as a guide in selecting a suitable, but sub-optimal, reconstruction algorithm.

#### IV. DECOMPOSITION OF $P[g|\sigma]$

We now analyse the non-diagonal  $\mathbf{M}$  case of  $P[g|\sigma]$  in Equation 3.4 by using a weighted SVD [9] of the imaging operator  $T$  in order to display explicitly the structure of  $P[g|\sigma]$ . Thus

$$\begin{aligned} \sum_i T_{a,i} \sqrt{\sigma_i} u_i^m &= \sqrt{\lambda_m} v_a^m \\ \sum_a T_{a,i}^* \sqrt{\sigma_i} v_a^m &= \sqrt{\lambda_m} u_i^m \end{aligned} \quad (4.1)$$

together with the orthonormality conditions

$$\begin{aligned} \sum_i u_i^{m*} u_i^n &= \delta_{m,n} \\ \sum_a v_a^{m*} v_a^n &= \delta_{m,n} \end{aligned} \quad (4.2)$$

Indices  $m$  and  $n$  label the order of the singular function (or vector), index  $i$  labels position in object space and index  $a$  labels position in image space. The  $u_i^m$  singular functions are used to decompose the visible subspace of the *continuous* object field. These components are mapped to the  $v_a^m$  which are the corresponding *discrete* image space singular vectors. The singular value  $\lambda_m$  gives the energy attenuation factor in channel  $m$ , and the weight  $\sqrt{\sigma_i}$  provides an amplitude scale for the object field at position  $i$ .

We shall now present a decomposition of  $P[g|\sigma]$  which will prove to be of great use in expressing the results of Section V and Section VI. For simplicity, we first derive  $P[g|\sigma]$  for a one-dimensional Nyquist-sampled band-limiting imaging operator (i.e. a ‘sinc’ function); the related object field problem was analysed in [8, 9]. Thus

$$T_a(x) = \frac{\sin[c(x - \frac{a\pi}{c})]}{\pi(x - \frac{a\pi}{c})} \quad a = 0, \pm 1, \pm 2, \dots \quad (4.3)$$

where we have used a continuous object space parametrised by  $x$ , and we have chosen to pass the band  $[-c, +c]$  of spatial frequencies. We now pose the (trivial) inverse problem of recovering the value of a supposedly constant cross section  $\sigma_0$  from the image data  $g$ . The image covariance matrix is given by

$$\mathbf{M} = \sigma_0 T T^\dagger + \nu \mathbf{I} \quad (4.4)$$

where the components of  $T T^\dagger$  are evaluated as follows:

$$\begin{aligned} T T_{a,b}^\dagger &\equiv \int_{-\infty}^{+\infty} dx T_a(x) T_b(x) \\ &= \frac{c}{\pi} \delta_{a,b} \end{aligned} \quad (4.5)$$

to give

$$\mathbf{M}_{a,b} = \left( \frac{\sigma_0 c}{\pi} + \nu \right) \delta_{a,b} \quad (4.6)$$

$P[g|\sigma_0]$  is then obtained as

$$P[g|\sigma_0] = \prod_{a=-\infty}^{+\infty} \frac{1}{\pi \left( \frac{\sigma_0 c}{\pi} + \nu \right)} \exp \left[ -\frac{|g_a|^2}{\frac{\sigma_0 c}{\pi} + \nu} \right] \quad (4.7)$$

which is a product of factors each of which derives from the modulus  $|g_a|$  of a single image sample. This type of data dependence (i.e. on  $|g_a|$  and not  $\arg(g_a)$ ) is implicitly assumed in segmentation algorithms, which use image modulus information alone to recover an estimate of the underlying cross section. The simple form of Equation 4.7 follows from the fortuitous property of Nyquist-sampled sinc functions shown in Equation 4.5, which leads to the diagonal (in fact, isotropic) image covariance  $\mathbf{M}$  shown in Equation 4.6.

We may use the SVD to write the part of  $T \sqrt{\sigma}$  which acts on the visible subspace as

$$(T \sqrt{\sigma})_{a,i} = \sum_m \sqrt{\lambda_m} v_a^m u_i^{m*} \quad (4.8)$$

## The Inverse Cross Section Problem for Complex Data

The part which acts on the invisible subspace is, by definition, zero. The noiseless image covariance matrix  $T \sigma T^\dagger$  may then be written as

$$\begin{aligned} (T \sigma T^\dagger)_{a,b} &= \sum_i (T \sqrt{\sigma})_{a,i} (T \sqrt{\sigma})_{b,i}^* \\ &= \sum_m \lambda_m v_a^m v_b^{m*} \end{aligned} \quad (4.9)$$

where we have used the orthonormality of the  $u^m$ . Combining Equation 2.9, Equation 3.3 and Equation 4.9 reduces the image covariance matrix  $\mathbf{M}$  in Equation 3.4 to

$$\begin{aligned} \mathbf{M} &\equiv T \sigma T^\dagger + \mathbf{N} \\ &= \sum_m (\lambda_m + \nu) v^m v^{m\dagger} \end{aligned} \quad (4.10)$$

so the (invariant) trace and determinant of  $\mathbf{M}$  are given by

$$\begin{aligned} \text{tr}(\mathbf{M}) &= \sum_m (\lambda_m + \nu) \\ \det(\mathbf{M}) &= \prod_m (\lambda_m + \nu) \end{aligned} \quad (4.11)$$

Object space is parametrised by a continuous index, so the dimension of the singular system  $\{u^m, v^m, \lambda_m\}$  of  $T$  is equal to the dimension of the discrete image space; hence the set of image singular vectors  $\{v^m\}$  is complete in image space (note that this may include some  $v^m$  with  $\lambda_m = 0$ ). This could also be deduced by inspection of Equation 4.10 as follows:  $\mathbf{M}$  is positive definite (see the argument after Equation 3.4), so its decomposition requires a complete set of image basis vectors, whence  $\{v^m\}$  is complete. We may therefore express  $\mathbf{M}^{-1}$  in Equation 3.4 as

$$\begin{aligned} \mathbf{M}^{-1} &= (T \sigma T^\dagger + \mathbf{N})^{-1} \\ &= \sum_m (\lambda_m + \nu)^{-1} v^m v^{m\dagger} \end{aligned} \quad (4.12)$$

We finally obtain the decomposition of  $P[g|\sigma]$  as [9]

$$P[g|\sigma] = \prod_m \frac{1}{\pi (\lambda_m + \nu)} \exp \left[ -\frac{|g'_m|^2}{\lambda_m + \nu} \right] \quad (4.13)$$

where the transformed components  $g'_m$  of  $g$  are given by

$$g'_m \equiv \sum_a v_a^{m*} g_a \quad (4.14)$$

The expression for  $P[g|\sigma]$  given in Equation 4.13 has the same structure as it had in Equation 4.7, except that it is now expressed as a product of independent factors which depend only on  $|g'_m|$  in a transformed image space. Thus the data  $g$  provide independent pieces of information about  $\sigma$  in the singular system weighted by  $\sqrt{\sigma}$  in object space. Strictly, we would have to specify the prior

PDF  $P[\sigma|X]$  as well, and then derive the full mutual information between  $g$  and  $\sigma$  to identify the independent pieces of information [9], but for simple purposes it suffices to inspect only the data-dependent part  $P[g|\sigma]$  of the posterior PDF  $\Pr[\sigma|g, X]$  in Equation 3.1. An important property of  $P[g|\sigma]$  in Equation 4.13 is that information about  $\sigma$  is contained in  $\arg(g_a)$  (although not in  $\arg(g_m')$ ). Comparing the assumptions which led to Equation 4.7 and Equation 4.13 reveals that any prior knowledge which permits  $\sigma \neq \sigma_0$  leads to  $\arg(g_a)$  containing useful information about  $\sigma$ , although this effect is probably not important unless  $\sigma$  fluctuates wildly on a length scale of a resolution cell: these are some of the pre-conditions for super-resolution to be applicable [4, 8, 9].

## V. GRADIENT OF $P[g|\sigma]$

In any practical Bayesian scheme for extracting a single representative  $\sigma$  as the solution to an inverse problem, we must make use of the gradient with respect to  $\sigma$  of  $P[g|\sigma]$ . This may either be supplied directly as an explicit expression for the gradient, or it may be obtained indirectly by (numerically) differentiating  $P[g|\sigma]$ : we shall adopt the former approach.

It is convenient to work with the (natural) logarithm  $L[g, \sigma]$  of  $P[g|\sigma]$  defined as

$$\begin{aligned} L[g, \sigma] &\equiv \log \{P[g|\sigma]\} \\ &= -\log \{\det(\mathbf{M})\} - g^\dagger \mathbf{M}^{-1} g + \text{constant} \end{aligned} \quad (5.1)$$

In order to differentiate Equation 5.1 with respect to a single component  $\sigma_i$  of  $\sigma$  we require the following results:

$$\begin{aligned} \frac{\partial \log \{\det(\mathbf{M})\}}{\partial \sigma_i} &= \frac{\partial \text{tr} \{\log(\mathbf{M})\}}{\partial \sigma_i} \\ &= \text{tr} \left\{ \frac{\partial \log(\mathbf{M})}{\partial \sigma_i} \right\} \\ &= \text{tr} \left\{ \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \right\} \end{aligned} \quad (5.2)$$

and

$$\frac{\partial \mathbf{M}^{-1}}{\partial \sigma_i} = -\mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \mathbf{M}^{-1} \quad (5.3)$$

where  $\mathbf{M}$  is operator valued (e.g. a matrix) so the ordering of factors is important. Thus

$$\frac{\partial L}{\partial \sigma_i} = -\text{tr} \left\{ \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \right\} + g^\dagger \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \mathbf{M}^{-1} g \quad (5.4)$$

Inspecting the origin of this expression in Equation 5.1 reveals that the first term (i.e. the determinant) gives the increase in  $L$  due to the decrease in size of the set  $\{g\}_\sigma$  of all possible datasets  $g$  which might derive from  $\sigma$ , and the second term (i.e. the scalar product) gives the increase in  $L$  due to an improved likelihood that the particular observed  $g$  indeed derives from  $\sigma$ . Strictly these two terms are inseparable, so they should always be considered together.

We shall now use the SVD of  $T$  (Equation 4.1 and Equation 4.2) to simplify Equation 5.4. Thus

$$\begin{aligned} \frac{\partial \mathbf{M}}{\partial \sigma_i} &\equiv \frac{\partial (T \sigma T^\dagger + \mathbf{N})}{\partial \sigma_i} \\ &= T \frac{\partial \sigma}{\partial \sigma_i} T^\dagger \end{aligned} \quad (5.5)$$

so

$$\begin{aligned} \frac{\partial \mathbf{M}_{a,b}}{\partial \sigma_i} &= T_{a,i} T_{b,i}^* \\ &= \frac{1}{\sigma_i} \sum_{m,n} \sqrt{\lambda_m \lambda_n} u_i^{m*} u_i^n v_a^m v_b^{n*} \end{aligned} \quad (5.6)$$

Combining Equation 4.12 with Equation 5.6 yields

$$\left( \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \right)_{a,b} = \frac{1}{\sigma_i} \sum_{m,n} \frac{\sqrt{\lambda_m \lambda_n}}{\lambda_m + \nu} u_i^{m*} u_i^n v_a^m v_b^{n*} \quad (5.7)$$

where we have used the orthonormality of the  $v^m$ . We may obtain the trace which is required in Equation 5.4 by contracting the  $a$  and  $b$  indices in Equation 5.7 using the orthonormality of the  $v^m$  to obtain

$$\text{tr} \left\{ \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \right\} = \frac{1}{\sigma_i} \sum_m \frac{\lambda_m}{\lambda_m + \nu} |u_i^m|^2 \quad (5.8)$$

If  $|u_i^m|^2$  is large (for any  $m$ ) it indicates that position  $i$  projects onto the visible subspace of the SVD of  $T$ , so provided  $\lambda_m \gg \nu$  a change in  $\sigma_i$  will lead to a significant change in the size of  $\{g\}_\sigma$ : this is in accord with our interpretation of the first term of Equation 5.4. From Equation 4.12 and Equation 5.7 we may obtain the second term of Equation 5.4 as

$$\left( \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \mathbf{M}^{-1} \right)_{a,b} = \frac{1}{\sigma_i} \sum_{m,n} \frac{\sqrt{\lambda_m \lambda_n}}{(\lambda_m + \nu)(\lambda_n + \nu)} u_i^{m*} u_i^n v_a^m v_b^{n*} \quad (5.9)$$

When contracted with  $g_a^* \cdots g_b$  to yield the second term of Equation 5.4, and recalling that  $g_m'$  (see Equation 4.14) is  $\mathcal{O}(\sqrt{\lambda_m})$ , we see that Equation 5.9 yields a version of Equation 5.8 which is coupled to the particular  $g$  which is observed: this is in accord with our interpretation of the second term of Equation 5.4 (see the discussion following Equation 5.8). Note the suppression of the contribution from those singular vectors for which  $\nu \gg \lambda_m$  in both Equation 5.8 and Equation 5.9.

We may relate these SVD expressions to simple physical quantities by introducing the SVD expression for the reconstruction  $f_r$  of the object field  $f$  [8, 9]

$$f_r \equiv \sigma T^\dagger [T \sigma T^\dagger + \mathbf{N}]^{-1} g \quad (5.10)$$

whence each component  $f_{r,i}$  is given by

$$\begin{aligned} f_{r,i} &= \sqrt{\sigma_i} \sum_{m,a} \frac{\sqrt{\lambda_m}}{\lambda_m + \nu} u_i^m v_a^{m*} g_a \\ &= \sqrt{\sigma_i} \sum_m \frac{\sqrt{\lambda_m}}{\lambda_m + \nu} u_i^m g'_m \end{aligned} \quad (5.11)$$

$f_r$  is the MAP reconstruction of  $f$  in our Gaussian PDF model (it does not depend on prior knowledge of the cross section  $P[\sigma|X]$ ). From Equation 5.11 we may derive the mean of  $|f_{r,i}|^2$  averaged over all images  $g$  in  $\{g\}_\sigma$

$$\begin{aligned} \langle |f_{r,i}|^2 \rangle &= \int dg P(g|\sigma) |f_{r,i}|^2 \\ &= \sigma_i \sum_m \frac{\lambda_m}{\lambda_m + \nu} |u_i^m|^2 \end{aligned} \quad (5.12)$$

where we have used the covariance of the  $g'_m$  (derived from those  $g$  in  $\{g\}_\sigma$ )

$$\langle g'_m g_n^{*\prime} \rangle = (\lambda_m + \nu) \delta_{m,n} \quad (5.13)$$

to eliminate a summation. The angle brackets here denote an average over all stochastic variables (field and image noise) whilst holding  $\sigma$  constant.

Using Equation 5.11 and Equation 5.12 we may reduce the gradient in Equation 5.4 to the simple form

$$\frac{\partial L}{\partial \log(\sigma_i)} = \frac{|f_{r,i}|^2 - \langle |f_{r,i}|^2 \rangle}{\sigma_i} \quad (5.14)$$

It is intuitively reasonable, but non-trivial, that the data dependence of this gradient, and ultimately the solution to the inverse cross section problem, is determined solely by the MAP solution  $f_r$ , to the corresponding inverse object field problem. Note that the  $\langle |f_{r,i}|^2 \rangle$  term depends only on the singular system which is independent of the data - it corresponds to the first term of Equation 5.4 - so it can be ignored when assessing the data dependence of  $\frac{\partial L}{\partial \log(\sigma_i)}$ .

The expression for  $\frac{\partial L}{\partial \log(\sigma_i)}$  in Equation 5.14 has an intuitive appeal because  $\langle |f_{r,i}|^2 \rangle$  tends to mimic  $\sigma_i$  (see Equation 5.12), so an increased likelihood is usually obtained by varying  $\sigma_i$  in such a way as to reduce the difference between  $|f_{r,i}|^2$  and  $\sigma_i$ . This lends support to the iterative super-resolution scheme [4] where  $\sigma$  is updated iteratively so as to enforce the ‘self-consistency’ condition  $\sigma_i = |f_{r,i}|^2$ . This algorithm may now be interpreted as an approximate scheme for maximising the likelihood function  $P[g|\sigma]$ .

As a simple demonstration we shall apply these results to the Nyquist-sampled ‘sinc’ imaging system for  $\sigma = \sigma_0$ . The SVD can be derived explicitly in the form

$$\begin{aligned} u^m(x) &= \sqrt{\frac{\pi}{c}} \frac{\sin[c(x - \frac{m\pi}{c})]}{\pi(x - \frac{m\pi}{c})} \\ v_a^m &= \delta_{m,a} \\ \lambda_m &= \frac{c\sigma_0}{\pi} \end{aligned} \quad (5.15)$$

and using the result

$$\sum_{m=-\infty}^{+\infty} \frac{\sin[c(x - \frac{m\pi}{c})]}{\pi(x - \frac{m\pi}{c})} \frac{\sin[c(y - \frac{m\pi}{c})]}{\pi(y - \frac{m\pi}{c})} = \frac{c}{\pi} \frac{\sin[c(x-y)]}{\pi(x-y)} \quad (5.16)$$

together with the orthonormality relations yields

$$\frac{\partial L}{\partial \log(\sigma_i)} = \frac{\frac{c}{\pi} \frac{\sigma_0 c}{\pi}}{\frac{\sigma_0 c}{\pi} + \nu} \left\{ \left| \sum_m \frac{\sin[c(x - \frac{m\pi}{c})]}{\pi(x - \frac{m\pi}{c})} g'_m \right|^2 - \left( \frac{\sigma_0 c}{\pi} + \nu \right) \right\} \quad (5.17)$$

where the transformed image components  $g'_m$  are identical to the original image components  $g_a$  because  $v_a^m =$

$\delta_{m,a}$ . This gradient clearly depends on the phase  $\arg(g_a)$  of the image data for  $x \neq \frac{n\pi}{c}$  ( $n = 0, \pm 1, \pm 2, \dots$ ), al-

though the likelihood itself does not depend on  $\arg(g_a)$  when  $\sigma = \sigma_0$ .

## VI. HIGHER DERIVATIVES OF $P[g|\sigma]$

In Section V we presented a derivation of the first derivative of the logarithm of the likelihood function  $\frac{\partial L}{\partial \log(\sigma_i)}$ . Whilst this alone would suffice for simple algorithmic purposes, we wish to demonstrate more fully the intimate connection between the inverse object field and the inverse cross section problems. Accordingly, we shall now present a derivation of the  $n$ th derivative  $D_n L$  defined as

$$D_n L[g|\sigma] \equiv \left\{ \prod_{s=1}^n \frac{\partial}{\partial \log(\sigma_{i_s})} \right\} L[g|\sigma] \equiv \partial_{i_1} \partial_{i_2} \cdots \partial_{i_n} L[g|\sigma] \quad (6.1)$$

where we have omitted the suffices from  $D_n$  for clarity, and we have introduced the notation

$$\partial_i \equiv \frac{\partial}{\partial \sigma_i} \quad (6.2)$$

From Equation 5.4 the first derivative  $D_1 L$  may be expressed in the form

$$\begin{aligned} D_1 L &= -A_1 + B_1 \\ A_1 &\equiv \text{tr} \{ \mathbf{M}^{-1} \partial_{i_1} \mathbf{M} \} \\ B_1 &\equiv g^\dagger \mathbf{M}^{-1} (\partial_{i_1} \mathbf{M}) \mathbf{M}^{-1} g \end{aligned} \quad (6.3)$$

$\mathbf{M}$  is a linear function of  $\sigma$  so second and higher derivatives of  $\mathbf{M}$  with respect to  $\sigma$  are zero, so we can obtain higher derivatives by repeatedly applying the rules

$$\begin{aligned} \partial_i \mathbf{M}^{-1} &= -\mathbf{M}^{-1} (\partial_i \mathbf{M}) \mathbf{M}^{-1} \\ \partial_i \partial_j \mathbf{M} &= 0 \end{aligned} \quad (6.4)$$

For example, the second derivative  $D_2 L$  (the Hessian matrix) is obtained in the form

$$\begin{aligned} D_2 L &= -A_2 + B_2 \\ A_2 &\equiv \partial_{i_2} A_1 \\ &= -\text{tr} \{ \mathbf{M}^{-1} (\partial_{i_2} \mathbf{M}) \mathbf{M}^{-1} (\partial_{i_1} \mathbf{M}) \} \\ B_2 &\equiv \partial_{i_2} B_1 \\ &= -g^\dagger \mathbf{M}^{-1} (\partial_{i_2} \mathbf{M}) \mathbf{M}^{-1} (\partial_{i_1} \mathbf{M}) \mathbf{M}^{-1} g - (i_1 \longleftrightarrow i_2) \end{aligned} \quad (6.5)$$

where  $(i_1 \longleftrightarrow i_2)$  denotes interchange of  $i_1$  and  $i_2$ .

This differentiation procedure of inserting  $-\mathbf{M}^{-1} (\partial_i \mathbf{M}) \mathbf{M}^{-1}$  for each  $\mathbf{M}^{-1}$  may be continued to obtain the  $n$ th derivative  $D_n L$  in the form

$$D_n L = -A_n + B_n \quad (6.6)$$

where  $A_n$  and  $B_n$  are given below:

---


$$\begin{aligned} A_n &\equiv \partial_{i_2} \partial_{i_3} \cdots \partial_{i_n} A_1 \\ &= (-1)^{n-1} \sum_q \text{tr} \{ \mathbf{M}^{-1} (\partial_{q_1} \mathbf{M}) \mathbf{M}^{-1} (\partial_{q_2} \mathbf{M}) \mathbf{M}^{-1} \cdots \mathbf{M}^{-1} (\partial_{q_n} \mathbf{M}) \} \\ \{q\} &\equiv (P_n/C_n)(i_1, i_2, \dots, i_n) \end{aligned} \quad (6.7)$$

where  $\{q\}$  is the set of orderings of the labels  $(i_1, i_2, \dots, i_n)$  produced by the action of the elements of the group  $P_n/C_n$ ,  $P_n$  is the permutation group of  $n$  letters and  $P_n/C_n$  has all length- $n$  cyclic rotations removed. The result for  $B_n$  is

$$\begin{aligned} B_n &\equiv \partial_{i_2} \partial_{i_3} \cdots \partial_{i_n} B_1 \\ &= (-1)^{n-1} \sum_p g^\dagger \mathbf{M}^{-1} (\partial_{p_1} \mathbf{M}) \mathbf{M}^{-1} (\partial_{p_2} \mathbf{M}) \mathbf{M}^{-1} \cdots \mathbf{M}^{-1} (\partial_{p_n} \mathbf{M}) \mathbf{M}^{-1} g \\ \{p\} &\equiv (P_n)(i_1, i_2, \dots, i_n) \end{aligned} \quad (6.8)$$

where the summation is now over all permutations of the indices. Unfortunately Equation 6.7 and Equation 6.8 cannot be simplified by commuting the operators to the right or to the left because

$$\begin{aligned} [\partial_i \mathbf{M}, \mathbf{M}^{-1}] &\neq 0 \\ [\partial_i \mathbf{M}, \partial_j \mathbf{M}] &\neq 0 \quad i \neq j \end{aligned} \quad (6.9)$$

Together, Equation 6.6, Equation 6.7 and Equation 6.8 describe the required  $n$ th derivative  $D_n L$ .

We may simplify this result by averaging  $B_n$  over all  $g$  in  $\{g\}_\sigma$  (using  $\langle gg^\dagger \rangle = \mathbf{M}$ ) in order to relate  $A_n$  and  $B_n$ . This yields

$$\begin{aligned}\langle B_n \rangle &= (-1)^{n-1} \sum_p \text{tr} \{ \mathbf{M}^{-1} (\partial_{p_1} \mathbf{M}) \mathbf{M}^{-1} (\partial_{p_2} \mathbf{M}) \mathbf{M}^{-1} \cdots \mathbf{M}^{-1} (\partial_{p_n} \mathbf{M}) \} \\ &= n A_n\end{aligned}\quad (6.10)$$


---

so we may write

$$D_n L[g|\sigma] = B_n - \frac{\langle B_n \rangle}{n} \quad (6.11)$$

which reduces to Equation 5.14 when  $n = 1$ . Together, Equation 6.8, Equation 6.10 and Equation 6.11 provide an alternative description of the required  $n$ th derivative  $D_n$ . The  $\langle B_n \rangle$  term of Equation 6.11 corresponds to the  $n$ th derivative of the first term of Equation 5.4, and a similar correspondence holds for the other term in each of these two equations. Furthermore the interpretation of these terms is analogous to those in Equation 5.14.

We may obtain yet another expression for the  $n$ th derivative by expressing  $B_n$  using the results which we obtained for  $\mathbf{M}^{-1}$  and  $\partial_i \mathbf{M}$  in Equation 4.12 and Equation 5.6 respectively. We may use the orthonormality of the  $v^m$  to remove all the internal summations between the  $\mathbf{M}^{-1}$  and  $\partial_i \mathbf{M}$  to obtain finally

$$B_n = (-1)^{n-1} \sum_p \frac{f_{r,p_1}^*}{\sqrt{\sigma_{p_1}}} \frac{f_{r,p_n}}{\sqrt{\sigma_{p_n}}} B'_{n,p} \quad (6.12)$$

where

$$B'_{n,p} \equiv \sum_{m_1, \dots, m_{n-1}} \frac{\lambda_{m_1} \cdots \lambda_{m_{n-1}}}{(\lambda_{m_1} + \nu) \cdots (\lambda_{m_{n-1}} + \nu)} \left\{ u_{p_1}^{m_1} (u_{p_2}^{m_1^*} u_{p_2}^{m_2}) \cdots (u_{p_{n-1}}^{m_{n-2}^*} u_{p_{n-1}}^{m_{n-1}}) u_{p_n}^{m_{n-1}^*} \right\} \quad (6.13)$$

The analogous expression for  $A_n$  may be obtained either from Equation 6.7 or from Equation 6.12 and Equation 6.13 using Equation 6.10. Either approach yields

$$A_n = (-1)^{n-1} \sum_q A'_{n,q} \quad (6.14)$$

where

$$A'_{n,q} \equiv \sum_{m_1, \dots, m_{n-1}} \frac{\lambda_{m_1} \cdots \lambda_{m_n}}{(\lambda_{m_1} + \nu) \cdots (\lambda_{m_n} + \nu)} \left\{ (u_{q_1}^{m_1^*} u_{q_1}^{m_2}) (u_{q_2}^{m_2^*} u_{q_2}^{m_3}) \cdots (u_{q_n}^{m_n^*} u_{q_n}^{m_1}) \right\} \quad (6.15)$$


---

Equation 6.13 and Equation 6.15 may then be used in Equation 6.6 to compute the  $n$ th derivative  $D_n L$ . Note that the dependence of  $B_n$  on the data is determined entirely by the MAP solution  $f_r$  to the inverse object field problem, and that  $A_n$  depends only on the structure of the SVD itself (which does not depend on  $g$ ), so to all orders of functional differentiation of  $\log(P[g|\sigma])$  with respect to  $\sigma$  there is an intimate relationship between the inverse cross section and inverse object field problems. Whilst we do not suggest that all orders are necessary for constructing a successful reconstruction algorithm for  $\sigma$ , we presume that the matrix of second

functional derivatives (the Hessian matrix)  $D_2 L$  given in Equation 6.5 would be of use in determining the shape of the likelihood function in a space, which in turn would assist in updating  $\sigma$ .

In order to verify the correctness of our higher derivative expressions we shall sum all orders of the Taylor expansion of  $\log(P[g|\sigma])$  for perturbations of the Nyquist-sampled ‘sinc’ imaging system about  $\sigma = \sigma_0$ , and then compare the result with a direct calculation of the same result. Using the results given in Equation 5.15 for the SVD we obtain from Equation 6.13 after summing over  $m_1, \dots, m_{n-1}$  using Equation 5.16

$$B'_{n,p} = \left( \frac{\frac{\sigma_0 c}{\pi}}{\frac{\sigma_0 c}{\pi} + \nu} \right)^{n-1} \frac{\sin[c(x_{p_1} - x_{p_2})]}{\pi(x_{p_1} - x_{p_2})} \cdots \frac{\sin[c(x_{p_{n-1}} - x_{p_n})]}{\pi(x_{p_{n-1}} - x_{p_n})} \quad (6.16)$$

and from Equation 6.15

$$A'_{n,q} = \frac{\frac{\sigma_0 c}{\pi}}{\frac{\sigma_0 c}{\pi} + \nu} \frac{\sin[c(x_{q_n} - x_{q_1})]}{\pi(x_{q_n} - x_{q_1})} B'_{n,p} \quad (6.17)$$

When  $x_{i_1} = x_{i_2} = \dots = x_{i_n}$  Equation 6.16 reduces to

$$B'_{n,p} = \left( \frac{\frac{\sigma_0 c^2}{\pi^2}}{\frac{\sigma_0 c}{\pi} + \nu} \right)^{n-1} \quad (6.18)$$

so Equation 6.12 for  $B_n$  reduces to

$$B_n = (-1)^{n-1} n! \left( \frac{\frac{\sigma_0 c^2}{\pi^2}}{\frac{\sigma_0 c}{\pi} + \nu} \right)^{n-1} \frac{|f_r(x)|^2}{\sigma_0} \quad (6.19)$$

Similarly  $A_n$  reduces to

$$A_n = (-1)^{n-1} (n-1)! \left( \frac{\frac{\sigma_0 c^2}{\pi^2}}{\frac{\sigma_0 c}{\pi} + \nu} \right)^n \quad (6.20)$$

We may use these results to develop a Taylor series for variation of  $\sigma(x)$  at position  $x$  (i.e. functional differentiation) by appealing to the discrete analogue. Thus

$$\Delta \log \{P[g|\sigma]\} = \sum_{n=1}^{\infty} \frac{(-A_n + B_n)}{n!} \sum_{i_1, \dots, i_n} \frac{\Delta \sigma_{i_1}}{\sigma_{i_1}} \frac{\Delta \sigma_{i_2}}{\sigma_{i_2}} \dots \frac{\Delta \sigma_{i_n}}{\sigma_{i_n}} \quad (6.21)$$

If, in the continuum analogue, the perturbation  $\Delta\sigma$  is the addition of a total cross section  $S$  uniformly in an interval of length  $\varepsilon$  (where  $\varepsilon \ll c^{-1}$ ), we may recover  $\Delta \log \{P[g|\sigma]\}$  from the discrete expression above by making the following replacements:

$$\begin{aligned} \frac{\Delta \sigma_i}{\sigma_i} &\rightarrow \frac{S}{\varepsilon \sigma_0} \\ \sum_i &\rightarrow \int_{x-\frac{\varepsilon}{2}}^{x+\frac{\varepsilon}{2}} dx \\ &= \varepsilon \end{aligned} \quad (6.22)$$

which leads to

$$\Delta \log \{P[g|\sigma]\} = \sum_{n=1}^{\infty} \frac{1}{n!} (-A_n + B_n) \left( \frac{S}{\sigma_0} \right)^n \quad (6.23)$$

The summation over  $n$  may be performed to yield

$$\begin{aligned} \Delta \log \{P[g|\sigma]\} &= \frac{\pi |f_r(x)|^2}{c \sigma_0} \frac{\delta}{1 + R \delta} - \log(1 + R \delta) \\ R &\equiv \frac{\frac{\sigma_0 c}{\pi}}{\frac{\sigma_0 c}{\pi} + \nu} \\ \delta &\equiv \frac{c S}{\pi \sigma_0} \end{aligned} \quad (6.24)$$

The radius of convergence of the sums is such that this result is guaranteed to be valid only for  $\delta$  which satisfy  $R |\delta| < 1$ . A good approximation to this condition for  $\nu \ll \frac{\sigma_0 c}{\pi}$  is the inequality

$$|S| < \frac{\pi \sigma_0}{c} \quad (6.25)$$

where  $\frac{\sigma_0 c}{\pi}$  is the cross section per resolution cell. Physically, the perturbation must not dominate the prior estimate of the cross section.

For comparison we may derive directly from Equation 5.1 a special case of Equation 6.24. Thus, place  $S$  at a position coincident with one of the image Nyquist sample points (the zeroth, say) to yield

$$\begin{aligned} \mathbf{M}_{a,b} &= \left( \frac{\sigma_0 c}{\pi} + \nu \right) \delta_{a,b} \\ \Delta \mathbf{M}_{a,b} &= S \frac{c^2}{\pi^2} \delta_{a,0} \delta_{b,0} \\ \Delta \mathbf{M}_{a,b}^{-1} &= -\frac{S \frac{c^2}{\pi^2}}{\frac{\sigma_0 c}{\pi} + \nu} \frac{1}{\frac{\sigma_0 c}{\pi} + \nu + S \frac{c^2}{\pi^2}} \delta_{a,0} \delta_{b,0} \end{aligned} \quad (6.26)$$

We may thus obtain the exact results

$$\Delta \log \{\det(\mathbf{M})\} = \log(1 + R \delta) \quad (6.27)$$

and

$$\Delta (g^\dagger \mathbf{M}^{-1} g) = -|g_0|^2 \frac{\frac{\sigma_0 c}{\pi}}{\left( \frac{\sigma_0 c}{\pi} + \nu \right)^2} \frac{\delta}{1 + R \delta} \quad (6.28)$$

However, using Equation 5.10 the SVD reconstruction at  $x = 0$  is given by

$$f_r(0) = \frac{\frac{\sigma_0 c}{\pi}}{\frac{\sigma_0 c}{\pi} + \nu} g_0 \quad (6.29)$$

Combining the results in Equation 6.27, Equation 6.28 and Equation 6.29 (using Equation 5.1) to obtain  $\Delta \log P[g|\sigma]$  yields equation (6.24) precisely. This checks the correctness of our functional differentiation results in this case. We also note that the radius of convergence criterion for Equation 6.24 is an unnecessary restriction in this case.

## VII. CONCLUSIONS

We have presented a detailed derivation of the properties of the likelihood function  $P[g|\sigma]$  which arises in the Bayesian analysis of the inverse cross section problem for complex imagery. Within the restrictions imposed by the Gaussian scattering model we find that the data dependence of all derivatives of  $P[g|\sigma]$  with respect to  $\sigma$  is contained in the MAP solution  $f_r$  to the inverse object field problem, so the inverse cross section and inverse object field problems are intimately related. The iterative

super-resolution algorithm [4] corresponds closely to gradient ascent of the likelihood function  $P[g|\sigma]$ , although this comparison is admittedly qualitative.

Throughout this paper we have emphasised that the inverse problem which we should be solving is the inverse *cross section* problem, and not the inverse *object field* problem. It remains an open question whether other inverse cross section problems can be simply expressed in terms of known solutions to their corresponding inverse scattered field problems.

- [1] M Bertero, P Brianzi, and E R Pike, *On the recovery and resolution of exponential relaxation rates from experimental data: Laplace transform inversions in weighted spaces*, Inverse Problems **1** (1985), no. 1, 1–15.
- [2] L M Delves, R T McQuillan, and C J Oliver, *A one-dimensional despeckling algorithm for SAR images*, Inverse Problems **4** (1988), no. 2, 471–484.
- [3] L M Delves, R T McQuillan, R Wilkinson, J B E Sandys-Renton, and C J Oliver, *A two-dimensional segmentation algorithm for SAR images*, Inverse Problems **7** (1991), no. 2, 203–220.
- [4] L M Delves, G Pryde, and S P Luttrell, *A super-resolution algorithm for SAR images*, Inverse Problems **4** (1988), no. 3, 681–703.
- [5] Special issue, Journal of the Optical Society of America A **4** (1987), no. 1.
- [6] E T Jaynes, *Prior probabilities*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 3, 227–241.
- [7] D T Kuan, A A Sawchuk, T C Strand, and P Chavel, *Adaptive restoration of images with speckle*, IEEE Transactions on Acoustics, Speech, and Signal Processing **35** (1987), no. 3, 373–383.
- [8] S P Luttrell, *Prior knowledge and object reconstruction using the best linear estimate technique*, Optica Acta **32** (1985), no. 6, 703–716.
- [9] S P Luttrell and C J Oliver, *Prior knowledge in synthetic aperture radar processing*, Journal of Physics D: Applied Physics **19** (1986), no. 3, 333–356.
- [10] W L Root, *Ill-posedness and precision in object-field reconstruction problems*, Journal of the Optical Society of America A **4** (1987), no. 1, 171–179.

# Image Compression Using a Multilayer Neural Network \*

S. P. LUTTRELL

Royal Signals and Radar Establishment, St Andrews Road, Malvern, WORCS, WR14 4NL, UK

We demonstrate that a topographic neural network model (Kohonen, 1984) may be used to data compress synthetic aperture radar (SAR) images by up to a factor of 8.

## I. INTRODUCTION

A property of neural networks is their ability to construct feature detectors as a result of supervised or unsupervised training. We demonstrate that a class of neural networks which produces topographic mappings [1] may be used to data compress SAR images. In Section II we summarise Kohonen's network learning algorithm, and we present an improved version of the algorithm in Section III. In Section IV we generalise the method to multilayer mappings and indicate how such networks might be implemented as table look-up operations. In Section V we apply such a multilayer network to the problem of data compressing SAR images.

## II. KOHONEN'S NETWORK LEARNING ALGORITHM

Define a mapping  $T(\mathbf{x}_{\text{in}})$  from a  $d_{\text{in}}$ -dimensional input vector  $\mathbf{x}_{\text{in}}$  to a  $d_{\text{out}}$ -dimensional ( $d_{\text{out}} \leq d_{\text{in}}$ ) output vector  $\mathbf{x}_{\text{out}}$  as

$$\mathbf{x}_{\text{out}} \equiv T(\mathbf{x}_{\text{in}}) \quad (2.1)$$

A vector quantiser representation of  $T(\mathbf{x}_{\text{in}})$  may be constructed by defining a set of code vectors  $\{\boldsymbol{\nu}_{\text{in},j} : j = 1, 2, \dots, m\}$  together with the  $m L_2$  norms

$$D_j(\mathbf{x}_{\text{in}}) \equiv \|\mathbf{x}_{\text{in}} - \boldsymbol{\nu}_{\text{in},j}\|^2 \quad j = 1, 2, \dots, m \quad (2.2)$$

and forming a mapping  $\mathbf{x}_{\text{in}} \rightarrow j_0$  such that  $j = j_0$  minimises  $D_j(\mathbf{x}_{\text{in}})$  with respect to  $j$ . The index  $j_0$  then plays the role of the output vector  $\mathbf{x}_{\text{out}}$ , and  $T^{-1}(\mathbf{x}_{\text{out}})$  becomes the pseudo-inverse  $j_0 \rightarrow \mathbf{x}_{\text{in},j_0}$ .

Kohonen has presented an algorithm for unsupervised training of the set of  $\boldsymbol{\nu}_{\text{in},j}$  from examples  $\mathbf{x}_{\text{in}}$  drawn from the pattern space [1]. The algorithm is very similar to the  $K$ -means clustering algorithm, and can also be shown to minimise the mean square reconstruction error in certain cases. In its simplest form the algorithm may be implemented as two update rules for each presentation

of a pattern  $\mathbf{x}_{\text{in}}$ :

$$\boldsymbol{\nu}_{\text{in},j_0} \longrightarrow \boldsymbol{\nu}_{\text{in},j_0} + \varepsilon_0 (\mathbf{x}_{\text{in}} - \boldsymbol{\nu}_{\text{in},j_0}) \quad (2.3)$$

$$\boldsymbol{\nu}_{\text{in},N(j_0)} \longrightarrow \boldsymbol{\nu}_{\text{in},N(j_0)} + \varepsilon_1 (\mathbf{x}_{\text{in}} - \boldsymbol{\nu}_{\text{in},N(j_0)}) \quad (2.4)$$

where  $1 > \varepsilon_0 > \varepsilon_1 > 0$ , and  $N(j_0)$  ranges over some neighbourhood set of  $j_0$ . The set of neighbourhoods defines the topology of the discrete output space which is usually chosen to be some regular lattice in one or two dimensions. Equation 2.3 by itself will lead to a vector quantiser, but if, in addition, Equation 2.4 is imposed then the vector quantiser approximates a topographic mapping [1]. This arises because Equation 2.4 tends to force  $\boldsymbol{\nu}_{\text{in},N(j_0)}$  to lie close to  $\boldsymbol{\nu}_{\text{in},j_0}$  in the pattern space, so the pseudo-inverses of  $j_0$  and  $N(j_0)$  are close together. The parameters  $\varepsilon_0$  and  $\varepsilon_1$  are functions of the number of training cycles which have elapsed, and  $\varepsilon_1$  also depends on the neighbourhood distance  $j_0 - N(j_0)$ . However we shall see that all such dependencies can be removed for the purpose of SAR image compression.

## III. AN IMPROVED LEARNING ALGORITHM

For simplicity we shall explain our improved algorithm for the case  $d_{\text{out}} = 1$ . Thus we train initially with a small value of  $m$  ( $m = 2$ , say) with nearest neighbour interactions in Equation 2.4 until the code vectors equilibrate. We then increase  $m$  to  $2m - 1$  by inserting additional code vectors according to the prescription

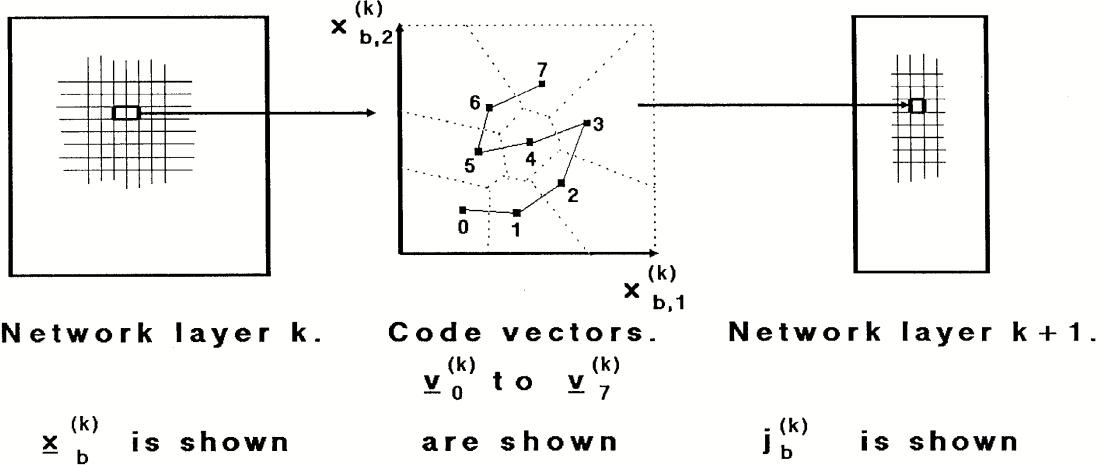
$$\boldsymbol{\nu}'_{\text{in},j} = \frac{1}{2} (\boldsymbol{\nu}_{\text{in},j} + \boldsymbol{\nu}_{\text{in},j+1}) \quad j = 1, 2, \dots, m-1 \quad (3.1)$$

and continue training as before using nearest neighbour interactions only. We repeat the cycle of training followed by insertion of new code vectors until  $m$  is sufficiently large that the pseudo-inverse  $j_0 \rightarrow \boldsymbol{\nu}_{\text{in},j_0}$  produces an acceptable reconstruction. When  $d_{\text{out}} > 1$  an appropriate multidimensional neighbourhood set of  $\boldsymbol{\nu}_{\text{in},j}$  used on the right hand side of Equation 3.1.

This technique of building up the number of code vectors converges more rapidly than Kohonen's original algorithm, because our code vector insertion procedure seeds the code vectors with good positions, and thus reduces the overall amount of computation which is required. We find that the consequential increase in the rate of convergence far outweighs the decrease due to adjustment of old code vectors after new ones have been inserted. This is because our technique ensures that the code vectors tend not to get stuck in 'knotted' configurations from

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 22, 2019.

This paper appeared in *Pattern Recognition Letters*, 1989, vol. 10, pp. 1-7. Received 26 February 1988. Revised 11 October 1988. © British Crown Copyright 1989.

Figure 1: Stages of coding layer  $k$  to produce layer  $k + 1$ .

which they have a low probability of escaping, and also because we do not waste resources attempting to compute the optimum locations of multiply correlated code vector positions during the early (approximate) stages of the training process, as in Kohonen's original scheme.

#### IV. MULTILAYER MAPPINGS

We have described how to map a  $d_{\text{in}}$ -dimensional input space into a  $d_{\text{out}}$ -dimensional output space. For image compression this is not feasible because  $d_{\text{in}}$  is enormous (it is equal to the number of pixels in the image). We must therefore partition the image into a large number

of blocks of pixels each of which is separately processed. Unfortunately this is computationally very intensive. If the number of bits which is used to represent  $\mathbf{x}_{\text{in}}$  is not too large and a particular set of code vectors is to be used for many images, then the processing time may be reduced by explicitly tabulating the mapping  $\mathbf{x}_{\text{in}} \rightarrow j_0$  for all  $\mathbf{x}_{\text{in}}$ . The maximum feasible table size is around 1 Megabyte using current memory technology, which limits the number of input bits to 20 assuming that  $0 < j_0 < 255$  (i.e. 1 byte per table entry). This limitation on  $d_{\text{in}}$  is unacceptable for image compression, so we must resort to multilayer mappings to factorise the overall data compression into calculable pieces.

For convenience we shall use the following notation

---

$\mathbf{x}^{(k)}$	= state of $k$ -th network layer
$\mathbf{x}_b^{(k)}$	= state of $b$ -th block of $k$ -th network layer
$\{\mathbf{x}_j^{(k)} : j = 1, 2, \dots, m_k\}$	= code vectors for $k$ -th network layer
$j_b^{(k)}$	= result of mapping $\mathbf{x}_b^{(k)}$

---

where each layer is partitioned into an array of nonoverlapping blocks of identical shape and orientation such as  $2 \times 1$  or  $2 \times 2$  pixel regions. We define the input image as  $\mathbf{x}^{(0)}$ , which we assume is represented as an array of  $w$ -bit unsigned integers. For simplicity we constrain  $m_k < 2^w$  so that the  $j_b^{(k)}$  may also be represented as a  $w$ -bit word, and we use the same value  $m_k = m$  for each layer  $k$ . The mapping from layer  $k$  to  $k + 1$  is accomplished by computing the  $m$  norms  $\|\mathbf{x}_b^{(k)} - \mathbf{v}_j^{(k)}\|^2$  ( $j = 1, 2, \dots, m$ ) for block  $b$ , and selecting as the com-

pressed version of  $\mathbf{x}_b^{(k)}$  the value  $j = j_b^{(k)}$  which gives the minimum norm. We show part of this process for the case  $m = 7$  in Figure 1, where a 2-tuple of pixels with value  $\mathbf{x}_b^{(k)} = (x_{b,1}^{(k)}, x_{b,2}^{(k)})$  is mapped (via the code vectors) to a 1-tuple  $j_b^{(k)}$  ( $0 \leq j_b^{(k)} \leq 7$  in this case), which becomes the value of the  $b$ -th pixel in layer  $k + 1$ . This process may be iterated by constructing blocks in layer  $k + 1$  to yield the  $\mathbf{x}_b^{(k+1)}$  which are then mapped to layer  $k + 2$  using another set of code vectors, and so on.

The learning algorithm may be generalised to multi-

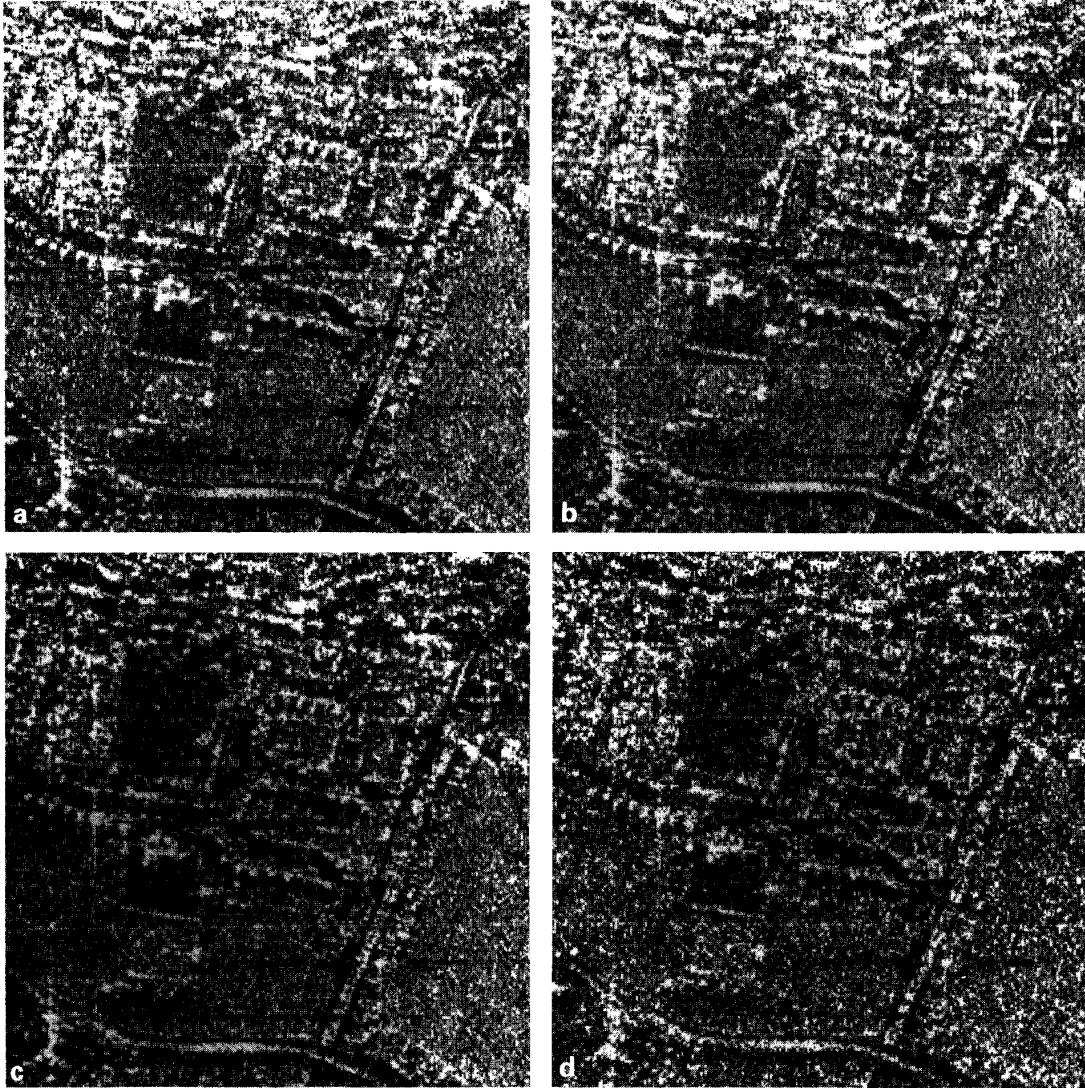


Figure 2: (a) Original modulus SAR image of an urban region. (b) Reconstruction of (a) after data compression by a factor of 2. (c) As (b) but with a compression factor of 4. (d) As (b) but with a compression factor of 8.

layer mappings by training the  $\nu_j^{(0)}$  on a single image  $\mathbf{x}^{(0)}$  followed by mapping  $\mathbf{x}^{(0)}$  to  $\mathbf{x}^{(1)}$  (via the  $j_b^{(1)}$ ). The  $\nu_j^{(1)}$  are then similarly trained on  $\mathbf{x}^{(1)}$ , and so on. The  $\mathbf{x}_b^{(k)} \rightarrow j_b^{(k)}$  mappings may then be tabulated for future use. Various improvements to this scheme are possible, such as simultaneous optimisation of all layers of the mapping, but we do not report them here. In our work we have used  $w = 8$ ,  $m = 2^w = 256$ , and block sizes  $1 \times 2$  and  $2 \times 1$ , so a 65 kilobyte ( $2 \times 8$  bit address space, 8 bit entries) look-up table suffices for each set of code vectors.

Image reconstruction is achieved by backpropagation from the last layer to the zeroth layer of the network. The inverse mapping from a pixel value  $j_b^{(k)}$  in layer  $k+1$  to an image block  $\mathbf{x}_b^{(k)}$  in layer  $k$  is given by the pseudo-inverse  $j_b^{(k)} \rightarrow \nu_j^{(k)}$  where the  $\nu_j^{(k)}$  are real-valued quantities.

It is necessary to round each component of  $\nu_j^{(k)}$  to the nearest integer in order to recover an  $\mathbf{x}_b^{(k)}$  which may be used in the next stage of the backpropagation from layer  $k$  to layer  $k-1$ .

## V. DATA COMPRESSION OF SAR IMAGES

In a recent study [2] some statistical measures of the quality of SAR image reconstructions from the output of predictive compression systems were presented. We shall adopt the same measures, which are:

*Mean:*

$$m \equiv \frac{1}{N} \sum_{n=1}^N x_n \quad (5.1)$$

*Variance:*

$$v \equiv \frac{1}{N} \sum_{n=1}^N (x_n - m)^2 \quad (5.2)$$

*Skewness coefficient:*

$$s \equiv \frac{\frac{1}{N} \sum_{n=1}^N (x_n - m)^3}{v^{3/2}} \quad (5.3)$$

*Kurtosis coefficient:*

$$k \equiv \frac{\frac{1}{N} \sum_{n=1}^N (x_n - m)^4}{v^2} - 3 \quad (5.4)$$

*Autocorrelation at delay one:*

$$a \equiv \frac{1}{N-1} \sum_{n=1}^{N-1} (x_n - m) (x_{n+1} - m) \quad (5.5)$$

where  $x_n$  is the grey level ( $0 \leq x_n \leq 255$ ) of pixel  $n$ , and  $N$  is the number of image pixels ( $1 \leq n \leq N$ ). In our work we use  $256 \times 256$  pixel images so  $N = 65536$ . The autocorrelation at delay one in Equation 5.5 is written in one-dimensional form; the two-dimensional version is analogous.

Our approach to SAR image compression is different from the model based predictive approach of [2]. A priori, the only constraint that we place on the image compression is the number of bits per pixel  $m$  (and hence the number of code vectors per layer in our scheme), and the particular choice of block size and shape to use in each layer of the network. Everything else is deduced from the statistical structure of the training set, which means that, for instance, our approach will attempt to preserve information about image speckle. It requires additional prior knowledge (i.e. a data model) to reject speckle: this is a future area of research.

The network is trained on one or more SAR images which contain the types of features for which the learning algorithm must form code vectors. Typically we use urban images which have many small bright features, and rural images which have characteristic textural correlations. In all cases we globally scale the image pixel values to normalise their mean value to some (approximate) standard.

We use a standard network structure parameterised by  $w = 8$ ,  $m = 2^w = 256$ , and block sizes  $1 \times 2$  and  $2 \times 1$ , where we alternate the  $1 \times 2$  and  $2 \times 1$  blocks from layer to layer of the network. Each network layer thus provides a compression factor of 2. The SAR images which we use are autofocused to remove the blurring effect of anomalous sensor motion; this results in a single-look fully speckled image with a resolution of 1.5 metre. For our purposes we then average the moduli of all contiguous non-overlapping  $2 \times 2$  pixel blocks to produce a smoother image (with  $1/4$  of the number of pixels of the original

Table I: Statistics for the urban images in Figure 2(a)-Figure 2(d).

	$R_0$	$R_1$	$R_2$	$R_3$
$m$	117.4	116.9	116.9	116.2
$v$	3736	3696	3619	3594
$a(a)$	1701	1681	1649	1355
$a(r)$	1823	1811	1736	1515
$s$	0.805	0.820	0.859	0.876
$k$	-0.147	-0.146	-0.032	-0.014

image). This is a trade-off which entails some loss of resolution, but which at the same time reduces image speckle (as in multi-look SAR processing).

In Figure 2(a) we show a typical urban SAR image. We use this image as a training set together with update parameter values  $\varepsilon_0 = 0.1$  and  $\varepsilon_1 = 0.02$ . For each network layer a training cycle consists of selecting  $32m$  image blocks at random in order to train the  $m$  code vectors, starting with  $m = 2$  and performing insertions upon equilibration of the code vectors until  $m = 257$  is reached after 9 training cycles. We finally eliminate one code vector chosen at random and perform a final training cycle to achieve equilibration of  $m = 256 = 2^8$  code vectors. Using these code vectors we forward propagate the image to the next network layer, and we repeat the training process to recover the next set of code vectors, and so on. We do not vary the values of the  $\varepsilon_0$  and  $\varepsilon_1$  parameters at any stage.

In Figure 2(b), Figure 2(c) and Figure 2(d) we show the image reconstructions which we obtain from layers 1, 2, 3 of a network which was trained on Figure 2(a). These correspond to compression factors of 2, 4, and 8 respectively. We also present the values of the statistics for both the original image and its reconstruction in Table I where  $R_k$  denotes the reconstruction from layer  $k$  ( $R_0$  is the original image),  $a(a)$  denotes the azimuth (horizontal) autocorrelation and  $a(r)$  denotes the range (vertical) autocorrelation at delay one.

We also perform a visual comparison of  $R_0$  and  $R_k$  ( $k > 0$ ) by ‘flicker photometry’ in which the images are rapidly interchanged on a display screen.  $R_1$  is virtually indistinguishable from  $R_0$ . The only obvious difference between  $R_2$  and  $R_0$  is that the grey level of some pixels is substantially different, but the resolution of fine detail is preserved.  $R_3$  shows the same effect to a greater extent, and probably represents the limit to which our data compression scheme can be pushed for this type of SAR imagery. The degradation of the  $a(a)$  and  $a(r)$  statistics is a major contributor to the visual differences between the reconstructions and the original.

In order to test the possibility of using a single set of code vectors to compress a variety of SAR images, we apply the same code vectors to a rural SAR image containing a wood and some other cultural features. The original image is produced in the same way as the urban

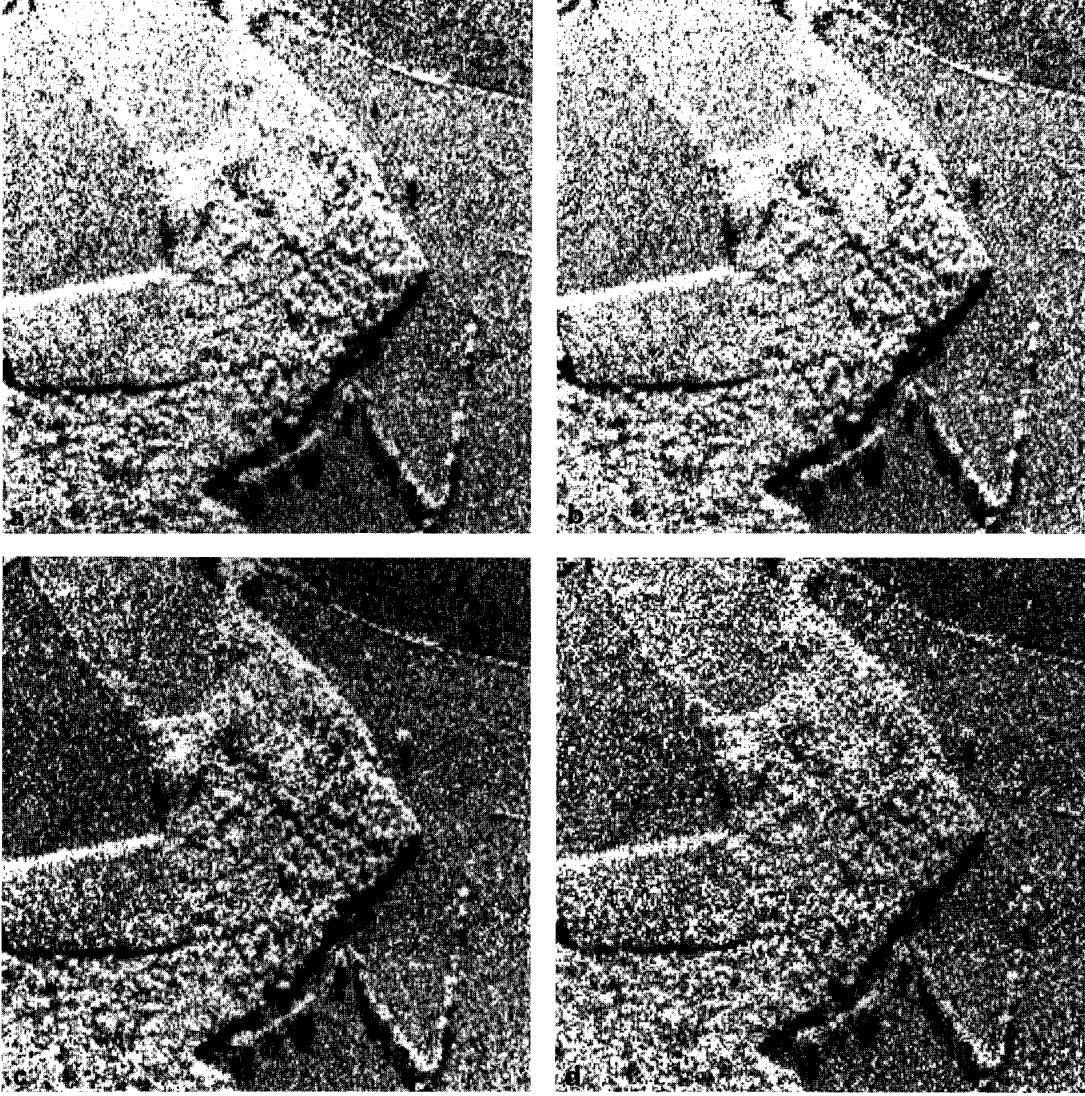


Figure 3: (a) Original modulus SAR image of a rural region. (b)-(d) Analogues of Figure 2(b)-Figure 2(d) for Figure 3(a).

Table II: Statistics for the rural images in Figure 3(a)-Figure 3(d).

	$R_0$	$R_1$	$R_2$	$R_3$
$m$	136.2	135.6	135.2	140.0
$v$	3622	3605	3660	4024
$a(a)$	1457	1443	1461	1396
$a(r)$	1854	1841	1760	1546
$s$	0.436	0.448	0.518	0.567
$k$	-0.643	-0.649	-0.559	-0.668

image and is shown in Figure 3(a). The reconstructions in Figure 3(b), Figure 3(c) and Figure 3(d) correspond to those in Figure 2(b), Figure 2(c) and Figure 2(d) and we present the corresponding statistics in Table II.

Table III: Statistics for the rural images using matched network.

	$R_0$	$R_1$	$R_2$	$R_3$
$m$	136.2	135.7	134.9	134.6
$v$	3622	3583	3591	3594
$a(a)$	1457	1438	1393	1206
$a(r)$	1854	1843	1763	1598
$s$	0.436	0.450	0.403	0.396
$k$	-0.643	-0.652	-0.695	-0.690

For comparison we present in Table III an equivalent set of results from a network which is trained on the rural image.

The similarity of Table II and Table III suggests that

the quality of image reconstruction is not very sensitive to precise matching of the network to the image to be compressed, which therefore suggests that we may tabulate a single set of code vectors for SAR image compression.

## VI. CONCLUSIONS

We have demonstrated the ability of a neural network to learn a feature space suitable for SAR image compres-

sion and reconstruction. The ability of a single network to compress a variety of SAR images suggests that a universal feature set might also be used for other low-level SAR image processing techniques.

- 
- [1] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
  - [2] S A Werness, *Statistical evaluation of predictive data compression systems*, IEEE Transactions on Acoustics, Speech, and Signal Processing **35** (1987), no. 8, 1190–1198.

# Hierarchical Vector Quantisation \*

S. P. Luttrell PhD

Royal Signals and Radar Establishment, St. Andrews Road,  
Malvern, Worcestershire WR14 3PS, United Kingdom

We present a method of vector quantisation which trades off accuracy for speed of encoding. We achieve this by hierarchically structuring a multistage encoder so that each stage encodes low dimensional input vectors. Such hierarchical encoders may easily be realised as a set of fast table look-up operations. We demonstrate how the Euclidean distortion in such a multistage encoder is approximately minimised by using Kohonen's topographic mapping learning algorithm from neural network theory. We also demonstrate the performance of the technique on various stochastic time series. We find that there is little loss in encoding accuracy, when compared with the exact nearest neighbour encoding using an equivalent single stage encoder.

## I. INTRODUCTION

Vector quantisation is the encoding of an input vector as a codeword selected from a codebook. The inverse operation of recovering an approximation to the original input vector is called decoding. The purpose of vector quantisation is to reduce the bandwidth and/or memory required to transmit and/or store the original vector. The LBG algorithm [3] provides a means of adjusting the code vectors to locate a local minimum of the average distortion. The choice of distortion measure depends on the application, but a Euclidean (or  $L_2$ ) distortion is usually used. The performance of the resulting codes is well documented [1].

The most time consuming part of vector quantisation is the encoding process, because the input vector must be compared with each code vector in the codebook, to ascertain which will yield the minimum distortion. For a codebook containing  $N$  vectors this takes  $\mathcal{O}(N)$  time. However, by trading off the accuracy and time to perform this operation one can encode in  $\mathcal{O}(\log(N))$  time [9]. If we use an  $L_2$  distortion measure, then time is measured in units of the time taken to perform a Euclidean distance computation, so even the  $\mathcal{O}(\log(N))$  scheme would require a lot of computation to block encode, for instance, a whole image.

The purpose of this paper is to present a hierarchical vector quantisation scheme, and to demonstrate its performance in some simple situations. Our hierarchical approach attempts to factorise the overall vector quantisation operation into a number of sub-operations each of which requires very little computation - a single table look-up per sub-operation, if possible. We must trade off encoding accuracy against encoding speed. However, we shall demonstrate that the loss in accuracy can be small (as low as a fraction of a dB), whilst the gain in speed (as a result of the use of look-up tables) is large.

## II. THEORY

In this Section we shall review the basic theory of vector quantisation, leading to the so-called LBG algorithm. We shall derive an extension of the LBG algorithm that makes the encoder output robust with respect to various types of code distortion. This extension turns out to be equivalent to a technique that is well known in neural network theory: Kohonen's topographic mapping training algorithm [2]. Finally, we shall apply these robust encoding techniques to the problem of multistage encoder design.

### A. Minimum distortion vector quantisation

We shall restrict our attention to an  $L_2$  distortion measure  $d(\mathbf{x}, \mathbf{x}')$  defined as

$$\begin{aligned} d(\mathbf{x}, \mathbf{x}') &\equiv (\mathbf{x}' - \mathbf{x})^T \cdot (\mathbf{x}' - \mathbf{x}) \\ &\equiv |\mathbf{x}' - \mathbf{x}|^2 \end{aligned} \quad (2.1)$$

where ' $T$ ' denotes transpose, ' $\cdot$ ' denotes vector dot product.  $d(\mathbf{x}, \mathbf{x}')$  measures the Euclidean distance between a vector  $\mathbf{x}$  and its reconstruction  $\mathbf{x}'$  after vector quantisation. It is important to note that we use this distortion measure for its mathematical simplicity, rather than attempt to define a distortion measure that rigorously measures information loss.

We shall assume that the vector quantiser is designed adaptively from a set of input vectors, known as the training set. We can conveniently summarise the information that is contained in the training set by defining the probability density function (PDF)  $P(\mathbf{x})$  over samples  $\mathbf{x}$  chosen from the training set. We may then evaluate any average over the training set by integrating over  $\mathbf{x}$  using probability measure  $P(\mathbf{x})$ .

Denote the encoding operation as  $y(\mathbf{x})$ , and the corresponding decoding operation as  $\mathbf{x}'(y)$ . If the codebook has  $N$  entries, then  $y$  is a discrete variable with  $N$  possible states. However, for convenience, we shall treat  $y$  as if it were a continuous variable. The average  $L_2$  distortion

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 10, 2019.

This paper appeared in *IEE Proceedings Part I*, 1989, vol. 136, no. 6, pp. 405-413. Paper 69361 (E4), first received 8th February and in revised form 21st August 1989.

$D$  is then given by the average of  $d(\mathbf{x}, \mathbf{x}'(y(\mathbf{x})))$  over  $\mathbf{x}$

$$D \equiv \int d\mathbf{x} P(\mathbf{x}) |\mathbf{x}'(y(\mathbf{x})) - \mathbf{x}|^2 \quad (2.2)$$

We may obtain a codebook which locally minimises  $D$  by using the LBG algorithm [3] to vary the functions  $y(\mathbf{x})$  and  $\mathbf{x}'(y)$ . The LBG algorithm must be run several times with different initial code vectors in order to circumvent the local minimum problem.

### B. Minimum distortion vector quantisation with noise stability

We now wish to modify the  $L_2$  distortion of Equation 2.2 to take account of the possibility that the output code  $y$  might subsequently be distorted. For simplicity, we shall model the distortion as an additive noise process  $n$ , which modifies  $y$  thus:  $y \rightarrow y + n$ . We shall describe the properties of this noise process by using a PDF  $\pi(n)$ . The  $L_2$  distortion  $D$  of Equation 2.2 is thus modified to become  $D_1$

$$D_1 \equiv \int d\mathbf{x} P(\mathbf{x}) \int dn \pi(n) |\mathbf{x}'(y(\mathbf{x}) + n) - \mathbf{x}|^2 \quad (2.3)$$

where we have now introduced an integration over realisations of the noise process.

The LBG algorithm can be suitably modified to take account of the effect of  $\pi(n)$  in Equation 2.2 [6, 8]. We present this modification in the form of a ‘continuous’ algorithm, in which the codebook is updated after every presentation of an input vector selected at random from the training set, as follows

$$\mathbf{x}'(y') \rightarrow \mathbf{x}'(y') + \varepsilon \pi(y' - y) [\mathbf{x} - \mathbf{x}'(y')] \quad \varepsilon > 0 \quad (2.4)$$

In order to reduce the size of the updates to  $\mathbf{x}'(y')$ , the size of  $\varepsilon$  should be gradually reduced as the training algorithm proceeds. This algorithm is essentially the same as one that is well known in neural network theory: Kohonen’s topographic mapping training algorithm [2].

The effect of the  $\pi(y' - y)$  is to cause many code vectors to be updated when each training vector is presented. The code vectors are updated in such a way that the distortion caused by the noise process is reduced. The noise process thus induces a set of neighbourhoods (or a topology) in the codebook, such that code vectors in the same neighbourhood tend to have similar code vectors. In our simulations we have restricted ourselves to a one dimensional topology in which the code vectors order themselves into a curve that attempts to fill the input space, as we shall see later on.

### C. Multistage vector quantisation

There are many useful applications of vector quantisers with noise stability. For our purposes we wish to

concentrate on multistage vector quantisation, where the output of one vector quantiser is fed into the input of another vector quantiser, and so on.

Consider two vector quantisers  $VQ_1$  and  $VQ_2$ .  $VQ_1$  feeds its output into  $VQ_2$ , and the output from  $VQ_2$  is the final encoded version of the original input to  $VQ_1$ . From the point of view of  $VQ_1$  the sole effect of  $VQ_2$  is to distort the information that is output by  $VQ_1$ , because in performing its quantisation  $VQ_2$  invariably discards some information. Thus an appropriate training algorithm for  $VQ_1$  is the Kohonen method of Equation 2.4, which takes account of the distortion that is induced by  $VQ_2$ . If we presume that the output of  $VQ_2$  is not distorted in any way, then we can use the LBG algorithm to train  $VQ_2$ .

The generalisation of this heuristic argument to a multistage vector quantiser is straightforward [6, 8]. Each stage must take account of the distortion that will be induced by all subsequent stages, and model it accordingly as a noise process. Kohonen’s topographic mapping training algorithm must therefore be used for all quantiser stages, except the last for which the LBG algorithm will suffice.

Hierarchical vector quantisation is a special case of multistage vector quantisation. Consider the case of a 4-dimensional input vector ( $\mathbf{x} \equiv (x_1, x_2, x_3, x_4)$ ). Figure 1a depicts a single stage vector quantiser  $y(\mathbf{x})$ , and Figure 1b depicts the corresponding two stage hierarchical vector quantiser  $y(h_1(x_1, x_2), h_2(x_3, x_4))$ . The most important difference between Figure 1a and Figure 1b is the input dimensionality of the various encoding operations. In Figure 1a it is 4, whereas in Figure 1b it is 2. If we were to consider a look-up table implementation of the whole encoding operation, then the hierarchical scheme in Figure 1b is very much cheaper (in terms of table size) to implement. This is the crucial advantage of hierarchical vector quantisation which we shall make use of in this paper.

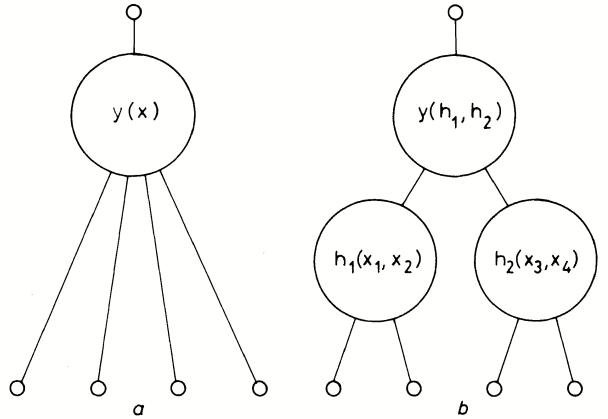


Figure 1: (a) Comparison of a single stage and a two-stage encoder. (b) Two-stage encoder chosen to be a binary-tree.

### III. APPLICATIONS

We shall now compare the performance of our hierarchical vector quantiser with an equivalent single stage vector quantiser. In order to make contact with previously published results, we shall restrict our attention to several stochastic time series selected from [1]. We shall summarise our version of Kohonen's topographic mapping training algorithm, and demonstrate some of its properties, including the effect of the choice of noise model  $\pi(n)$ . We shall then apply a two-stage hierarchical encoder to various times series: uncorrelated Gaussian noise, correlated Gaussian noise (Gauss-Markov, in fact) and Gamma noise. Finally we shall apply a three-stage hierarchical encoder to correlated Gaussian noise.

#### A. Some additional training algorithm details

Although it is not an essential part of our hierarchical vector quantisation scheme, we shall make use of an enhanced version of Kohonen's topographic mapping training algorithm which we have presented before [5]. Our algorithm gives much faster convergence than Kohonen's original algorithm, which saves much time in numerical simulations. Our algorithm has an outer loop which controls the number of code vectors, so that each time the outer loop increments its counter the current code vectors are used to 'seed' a larger set of code vectors. Thus initially we use  $N = 2$  to obtain a very crude vector quantisation, and each time we go round the outer loop we increase  $N$  to obtain a more refined vector quantisation.

Because we now intend to perform numerical simulations, we shall use a discrete notation for  $y$  and  $n$  (i.e.  $y = 0, 1, 2, \dots, N - 1$ , and similarly for  $n$ ). We shall permit only  $\pi(0), \pi(\pm 1), \pi(\pm 2)$  to be nonzero, which constrains the type of noise models that we can use. We shall also use the same number of code vectors for each of the vector quantisers in the hierarchy. More sophisticated algorithms would relax one or both of these simplifying assumptions.

The training algorithm that we use for a single layer of the hierarchical vector quantiser is then:

1. Initialise  $N = 2$ .
2. Initialise the code vectors randomly from the training set.
3. Define  $\pi(0), \pi(\pm 1), \pi(\pm 2)$  ( $\varepsilon$  is absorbed into this definition).
4. Initialise the number of updates per code vector  $U = 32$ .
5. Repeat the following until the distortion is small enough:
  - (a) Sample  $NU$  times from the training set, performing Kohonen topographic updates, as given by Equation 2.4.
  - (b) Seed a larger set of code vectors from the current set.

- (a) Sample  $NU$  times from the training set, performing Kohonen topographic updates, as given by Equation 2.4.
- (b) Seed a larger set of code vectors from the current set.

In step 5(a) the samples should strictly be selected at random from the training set. However, in our applications we shall select the samples sequentially from a time series. Furthermore, to ensure that the algorithm behaves well, the following inequalities must hold (with  $\varepsilon$  absorbed into the definition of  $\pi(n)$ )

$$\frac{1}{NU} \ll \sum_{i=-2}^{+2} \pi(i) \ll 1 \quad (3.1)$$

The lower bound ensures that the algorithm has a sufficient number of updates to converge. The upper bound ensures that the algorithm has a memory of many past input vectors: this enables it to build up a good representation of the distribution of input vectors.

For hierarchical vector quantisers there are some additional considerations. For each stage, we shall adjust  $\pi(\pm 1)$  and  $\pi(\pm 2)$  to appropriately model the distorting effect of subsequent stages. Because our hierarchical scheme is feed-forward, we can train stages in the order in which they appear in the hierarchy. Thus we train the first stage by estimating the distortion induced by second and subsequent stages, and then we run our training algorithm using vectors from the training set. When the first stage has converged, we train the second stage by estimating the distortion induced by third and subsequent stages, and then we run our training algorithm using the output of the first stage as the training set. We continue this process stage by stage until the last stage has converged.

#### B. Preliminary experiments

We now summarise pictorially in Figure 2a the behaviour of our modified version of Kohonen's training algorithm, using  $\pi(0) = 0.1$ ,  $\pi(\pm 1) = 0.01$ ,  $\pi(\pm 2) = 0$ ,  $U = 32$  and  $N = 2, 3, 5, 9, 17, 33$ . The training data consists of 500 2-tuples  $(x_1, x_2)$ , where each  $x_i$  ( $i = 1, 2$ ) is the sum of three i.i.d. random numbers uniformly distributed on  $[0, 1]$  (thus crudely approximating a Gaussian distribution). Note that the code vectors adjust themselves into a continuous line under the influence of the  $\pi(\pm 1)$  updates (i.e. the noise model forces the codebook to adopt a one dimensional topology). The most important property of the results in Figure 2a is the progressive refinement of the line of code vectors as  $N$  is increased. We observe that the general structure of the line that is obtained at  $N = N_0$  is preserved for all  $N > N_0$ , although additional fine structure appears. This observation must break down at some point as the number of updates  $U$  is increased, because the stochastic nature of

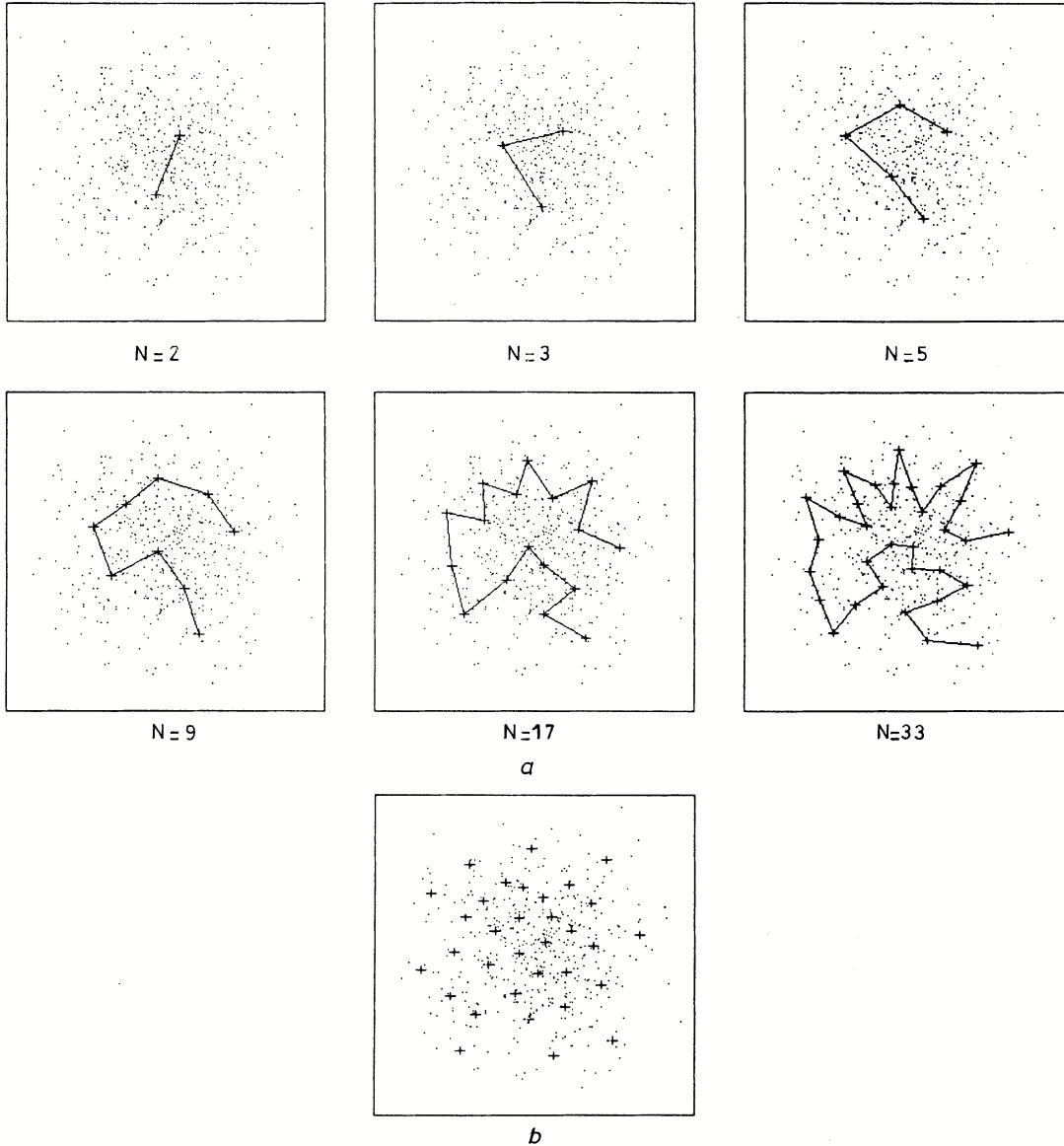


Figure 2: Stages in implementation of our modication to Kohonen’s learning algorithm. (b) shows result of the LBG algorithm applied to the  $N = 33$  case of (a).

the learning algorithm gradually destroys the memory of the earlier form of the line of code vectors. The total number of updates that is required to obtain the final result is 2208 ( $= 32.(2 + 3 + 5 + 9 + 17 + 33)$ ).

We obtain the result in Figure 2b by priming the LBG algorithm with the  $N = 33$  result in Figure 2a, and then run it to convergence. The code vectors are now disconnected (i.e.  $\pi(i) = 0$  for  $i \neq 0$ , effectively), so we do not connect them with a line as we did in Figure 2a. We estimate the  $L_2$  distortion in the  $N = 33$  case in Figure 2a (by averaging over the training set) to be 0.053. After we run the LBG algorithm in Figure 2b it reduces to 0.048. Both here and elsewhere we scale the  $L_2$  distortion by defining  $d \equiv \frac{D}{\text{var}(x)}$ , which is not to be confused

with the  $d(\mathbf{x}, \mathbf{x}')$  of Equation 2.1. The numerator is the distortion  $D$ , as defined in Equation 2.2, which ignores the effect of noise, and  $\text{var}(x)$  is the estimated variance of the training set. Overall,  $d$  is a scale independent way of measuring the distortion which is induced by a particular encoding/decoding scheme.

The relative size of the distortions in Figure 2a and Figure 2b arises from a number of causes. First, we use  $\pi(\pm 1) \neq 0$  in Figure 2a, which models a nonzero noise distortion at the output of the encoder, and thus produces a larger reconstruction distortion than in Figure 2b. Secondly, the stochastic nature of the training algorithm that we use in Figure 2a, together with the possibility that the number of updates  $U$  is too small, conspire

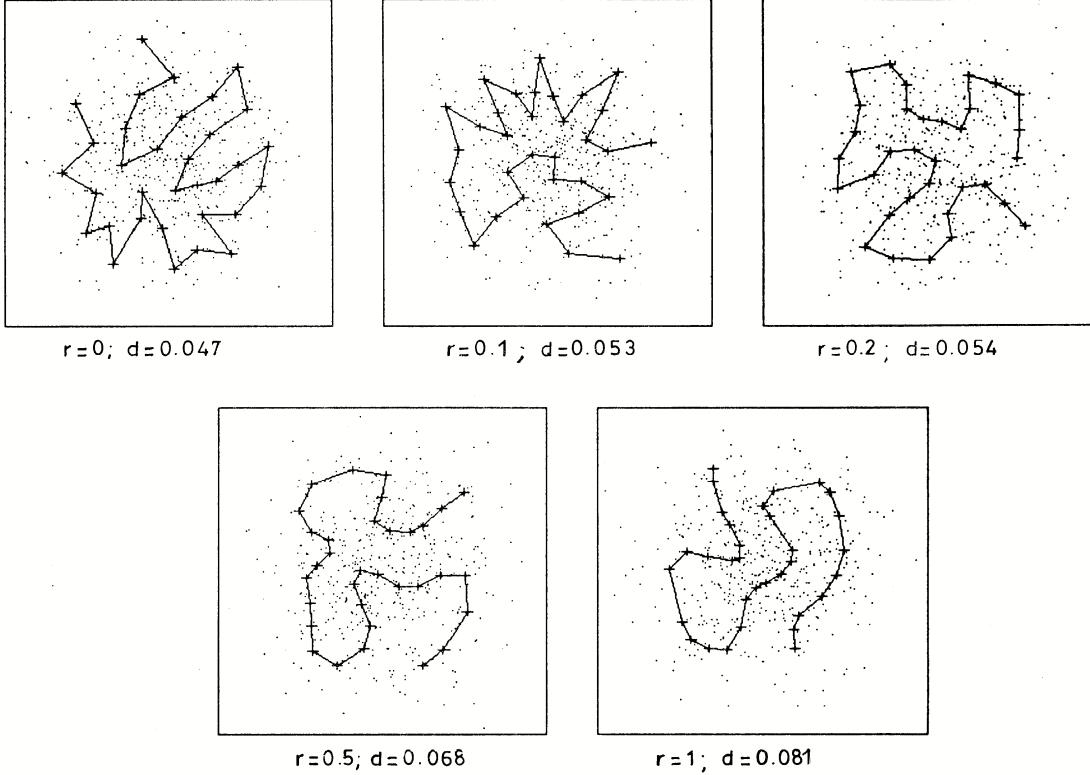


Figure 3: Comparison of structure of line of code vectors which results from different values of  $\pi(\pm 1)$ .

make the reconstruction distortion greater than it should be.

In Figure 3 we present the results of an experiment into the effect of varying  $\pi(\pm 1)$  whilst holding all the other parameters at the values used in Figure 2a. We define a noise level  $r \equiv \frac{\pi(1)}{\pi(0)}$ , and present results for the cases  $r \in \{0, 0.1, 0.2, 0.5, 1\}$  and  $N = 33$ . We run each case independently of the others by using the training data in a different random order each time. The most obvious feature to note is the progressive smoothing and organisation of the line of code vectors as  $\pi(\pm 1)$  is increased. This organisation minimises the additional distortion that is induced by the increasing noise level  $r$ .

In Figure 3 we also show the scaled  $L_2$  distortion  $d$  that is obtained in each case. As expected, increasing the level of noise  $\pi(\pm 1)$  in the model causes  $d$  to increase. The  $\pi(\pm 1) = 0$  (or LBG-like) result is marginally smaller than the LBG result of Figure 2b because the training set uses different random numbers (and also the LBG algorithm is guaranteed to find only a local minimum distortion). Of particular interest in Figure 3 is the structure that we observe in the  $\pi(\pm 1) = 0$  result. In this case the code vectors are not explicitly coupled to each other, so there is no obvious reason why they should form a connected line. The source of this excess structure is the code vector seeding process that we use in our training algorithm, and the fact that we update only a limited number of times

( $U = 32$ ). Taken together, these cause the code vectors to retain a memory of the positions of the code vectors earlier episodes of training, hence the emergence of excess structure.

We anticipate that reducing the strength of the noise  $\pi(\pm 1)$  together with reducing  $U$  at the same time might produce partially cancelling effects, because the loss of structure caused by reducing  $\pi(\pm 1)$  might be compensated for by the excess structure left over after an abbreviated learning algorithm. We shall see an example of this later on in Figure 4.

### C. The effect of the choice of noise model

We now present a preliminary survey of the effect of various choices of  $\pi(0)$ ,  $\pi(\pm 1)$ ,  $\pi(\pm 2)$  and  $U$ . Consider a two-stage encoder as depicted in Figure 1b, and run our learning algorithm with  $N = 2, 3, 5, 9, 17$ . Furthermore, because the statistical properties of the two 2-tuples  $(x_1, x_2)$  and  $(x_3, x_4)$  are the same, we shall set  $h_1(a, b) = h_2(a, b)$  in Figure 1b. Note that this translational symmetry principle generalises to more complicated hierarchical encoders. We may obtain a crude estimate of the correct noise model by estimating the distortion that is introduced by  $y(h_1, h_2)$ . For  $N = 17$ ,  $(h_1, h_2)$  is quantised into seventeen cells, so the distortion in  $h_1$  or  $h_2$  is  $\sqrt{17}$ , thus the range of the appropriate equivalent

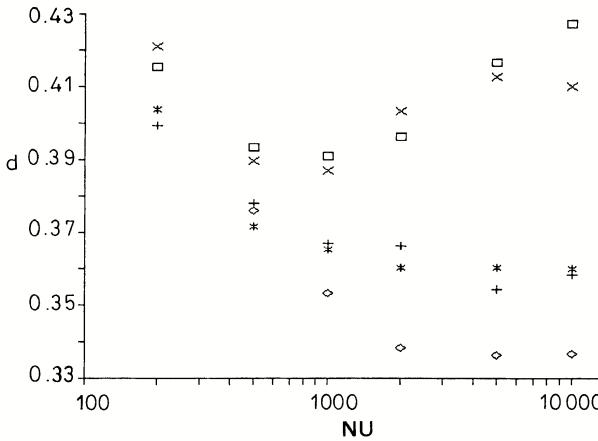


Figure 4: Plots of the scaled reconstruction distortion  $d$  versus the number of updates  $NU$  for various noise models  $\pi(\pm 1)$  and  $\pi(\pm 2)$ .

noise process is  $\frac{\sqrt{17}}{2} \simeq 2$ . Thus we shall use  $\pi(\pm 1) > 0$  and  $\pi(\pm 2) > 0$ , but set  $\pi(\pm k) = 0$  for  $k > 2$ . We shall not attempt to adjust the range or strength of the noise process as we vary  $N$  ( $N = 2, 3, 5, 9, 17$ ), although strictly we should do so. Furthermore, where the noise process attempts to move a code vector outside the range of the codebook (i.e. at the ends of the line of code vectors), we simply ignore the noise.

	$\pi(0)$	$\pi(\pm 1)$	$\pi(\pm 2)$	
Stage 1	0.03	0.015	0	$\times$
	0.03	0.03	0	$\square$
	0.03	0.015	0.015	*
	0.03	0.03	0.03	+
Stage 2	0.03	0	0	$\diamond$

In Figure 4 we show the scaled  $L_2$  distortion  $d$  for an uncorrelated noise input under various training conditions. Each input sample is the sum of five independent and identically distributed (i.i.d.) random variables distributed uniformly on  $[0, 1]$  (i.e. it is approximately Gaussian). We show the progress of the learning algorithm for five different choices for  $\pi(0)$ ,  $\pi(\pm 1)$ ,  $\pi(\pm 2)$  (where  $\pi(\pm k) = 0, k > 2$ ). The value of  $NU$  quoted in Figure 4 is calculated when  $N$  attains its final value ( $N = 17$  in this case). Each point plotted is the average of the distortions obtained from 16 independent runs of the learning algorithm.

The lowest set of points is the result of a single stage encoder trained with  $\pi(\pm 1) = \pi(\pm 2) = 0$  (i.e. LBG-like). This converges to a scaled distortion of about 0.34, which provides a lower bound for the two stage encoder to aim for. The two stage encoder cannot have a smaller distortion than the single stage encoder, because the freedom of choice for placing code vectors in the input space is reduced by using a hierarchically structured encoder. This lower bound corresponds to a (reciprocal) scaled  $L_2$

distortion of 4.7 dB, which is consistent with results reported elsewhere for four-sample block encoding using 1 bit per sample of an uncorrelated Gaussian time series [1] (to be compared with our use of  $\frac{\log_2 17}{4} \simeq 1$  bit per sample).

The two sets of points which converge to a scaled distortion of about 0.36 result from a two-stage encoder that we train with two different noise models using  $\pi(\pm 1) > 0$  and  $\pi(\pm 2) > 0$ . The difference between the results that we obtain from the two noise models appears to be insignificant, and they perform almost as well as the ideal single stage encoder. This demonstrates that, at least in this case, the output of the first encoding stage retains almost all pertinent information relating to vector quantisation of the four-dimensional input space.

Finally, the two sets of points which have the largest distortion derive from a two-stage encoder using a restricted noise model in which  $\pi(\pm 1) > 0$  and  $\pi(\pm 2) = 0$ . For  $NU \simeq 1000$  this restricted model has a significantly smaller distortion than for  $NU \simeq 10000$ . This effect is the result of excess structure in the positions of the first stage code vectors, as discussed in Section III B in connection with Figure 3. This mimics the type of correlations which would have been induced by  $\pi(\pm 2) > 0$ , and so reduces the distortion. We could take advantage of this happy coincidence to speed up the learning algorithm by avoiding the updates which are associated with  $\pi(\pm 2)$ . However, this approach offers little advantage because the speed of the algorithm is dominated by the time it takes to search for the nearest neighbour code vector, which is the same whatever the noise model. We shall therefore not use this short cut.

#### D. Vector quantisation of time series

We shall now present results that we obtain from two-stage vector quantisation of three different time series: uncorrelated Gaussian noise, correlated Gaussian noise (correlation coefficient  $\rho = 0.85$ ), and uncorrelated gamma noise (order parameter  $\nu = 0.5$ ). We shall also present a three-stage vector quantisation of the correlated Gaussian noise case. In all cases we use a hierarchical encoder like that shown in Figure 1b (i.e. a binary tree). We choose the learning parameters to be  $\pi(0) = 0.03$ ,  $\pi(\pm 1) = \pi(\pm 2) = 0.015$ ,  $\pi(\pm k) = 0, k > 2$ ,  $N = 2, 3, 5, 9, 17$ ,  $NU = 2000$  for  $N = 17$ , (recall that  $\pi(\pm 1) = \pi(\pm 2) = 0$  for the last encoder stage, which is LBG-like).

We present our results in Figure 5 to Figure 8. In each of these Figs. our results have the same format, and in this paragraph we use the notation Figure  $Fz$  to refer to part  $z$  ( $z = a, b, c, d$ ) of Figure  $F$  ( $F = 5, 6, 7, 8$ ). Figure  $Fa$  presents the code vectors for each encoder stage as a curve embedded in the two-dimensional input space, together with an estimate of the corresponding co-occurrence matrix measured using  $16 \times 16$  bins. The line of code vectors conveys the same information

that we showed in the  $N = 17$  case of Figure 2a. Figure Fb presents, as fragments of time series, the (centroid of the set of)  $\mathbf{x}$  which selects each code vector in the line of code vectors. In the case of the first encoder stage, this is the code vector itself, and in the case of each subsequent stage it is the reconstruction vector that minimises the  $L_2$  distortion (whilst holding all other variables fixed). Figure Fe shows 512 samples of both the original time series and its reconstruction from the output of the last encoder stage. The horizontal scale in Figure Fc is half that in Figure Fb. We draw a line horizontally through the mean value of the time series, and we place tick marks every 50 samples along the time axis. In all cases the time series are 1024 samples long with 12 bits per sample, and we use 12 bits per sample to represent each of the code vectors whilst the training algorithm is running. In order to match the output of an encoder stage ( $\log(N)$  bits) to the input (12 bits) of the next encoder stage, we perform a simple linear scaling: this does not alter the information content of the encoded data. We use a large number of bits during training to ensure that the code vectors have a fine enough quantisation to adjust smoothly to updates demanded by Equation 2.4. Note that we do not need to use a large number of bits after learning has converged, so we do not then scale the output of each encoder stage from  $\log(N)$  bits to 12 bits. We encode  $2^L$  successive samples of a time series by using an  $L$ -stage hierarchical encoder, and then reconstruct by using the appropriate reconstruction vector as shown in Figure Fb. We then move the encoder along by  $2^L$  samples and repeat the operation, and so on. In Figure Fd we present a co-occurrence matrix created from a pair of samples: an original time series sample, and its corresponding reconstruction. We call this a cross-cooccurrence matrix. The width of the band indicates the extent of the distortion.

We can compare the cost of determining the nearest neighbour code vector by the exact  $\mathcal{O}(N)$  and the approximate  $\mathcal{O}(\log(N))$  Euclidean distances scheme, and our approximate hierarchical scheme. For a four dimensional input space to be encoded into one of  $N$  ( $= 17$ ) code vectors requires seventeen Euclidean distances to be computed, each of which involves four subtract-multiply-accumulate operations. The approximate  $\mathcal{O}(\log(N))$  scheme would have  $N$  replaced by  $\log(N)$  ( $\simeq 4$ ). Because both of these methods need to compute Euclidean distances they are rather slow. On the other hand, in our approximate hierarchical scheme we would record offline all required results in the form of look-up tables, and then process the four dimensional input space as shown in Figure 1b. Translational symmetry implies that we need only two distinct look-up tables in this case, and we need to perform a total of three look-up table operations. The size of each table depends exponentially on the number of input bits, and linearly on the number of output bits. As we have already observed, a large number of bits is required to represent numbers during training in order to enable the update equation to operate correctly, so we do not use look-up tables during

training. However, once training is complete we are free to reduce the number of bits to their normal level, and to fill in the look-up table entries. For the example shown in Figure 1b we may approximate the input samples by using four bits per sample (this is probably an overestimate given the crudity of the quantisation that we are about to perform), so the address space size of the first stage look-up table is 256 ( $= 2^{4+4}$ ). Furthermore, for all encoder stages we use  $N = 17$  (i.e.  $\log(N) \simeq 4$ ), so the number of output bits from each look-up table is approximately four. The approximation  $\log(N) \simeq 4$  also implies that the address space size of the second stage look-up table is 256. Thus all the look-up tables fortuitously have the same size, which is 256 by four bits. We note that using currently available memory technology we could easily upgrade these four bit quantities to eight bit quantities, which would then require 65 kbyte look-up tables. In summary, the overall memory requirements for representing our look-up tables are modest.

### 1. Uncorrelated Gaussian noise

We generate this noise process as a sum of five i.i.d. random variables uniformly distributed on  $[0, 1]$ . The stage one co-occurrence matrix in Figure 5a clearly shows the joint distribution of a pair of uncorrelated variables. The stage one line of code vectors has gravitated towards the centre of this distribution to reduce the  $L_2$  distortion. The stage two co-occurrence matrix is approximately uniform, which indicates that the intermediate codes are used with approximately equal frequency, and that they are independent, as expected. In Figure 5b the stage one fragments of time series corresponding to adjacent code vectors are similar, as expected. We observe a similar effect for the stage 2 fragments, which also begin to reveal typical pieces of the time series such as peaks and troughs. Figure 5c and Figure 5d reveal that the peaks (positive and negative) of the original time series have been clipped in the reconstruction. This is because such peaks are rare events, so they are poorly quantised. The scaled  $L_2$  distortion, in this particular example, is 0.36, which is almost as good (0.3 dB loss) as would be obtained from a single stage four-sample block encoder using one bit per sample [1].

### 2. Correlated Gaussian noise ( $\rho = 0.85$ )

We generate this noise process by using a first order Gauss-Markov model

$$x_{n+1} = \rho x_n + r_n \quad (3.2)$$

where the  $r_n$  are i.i.d. zero mean random Gaussian variables with unit variance ( $\langle r_n^2 \rangle = 1$ ). Thus  $\langle x_n^2 \rangle = \frac{1}{1-\rho^2}$  and  $\frac{\langle x_n x_{n+1} \rangle}{\langle x_n^2 \rangle} = \rho$ . We prime the time series with a zero

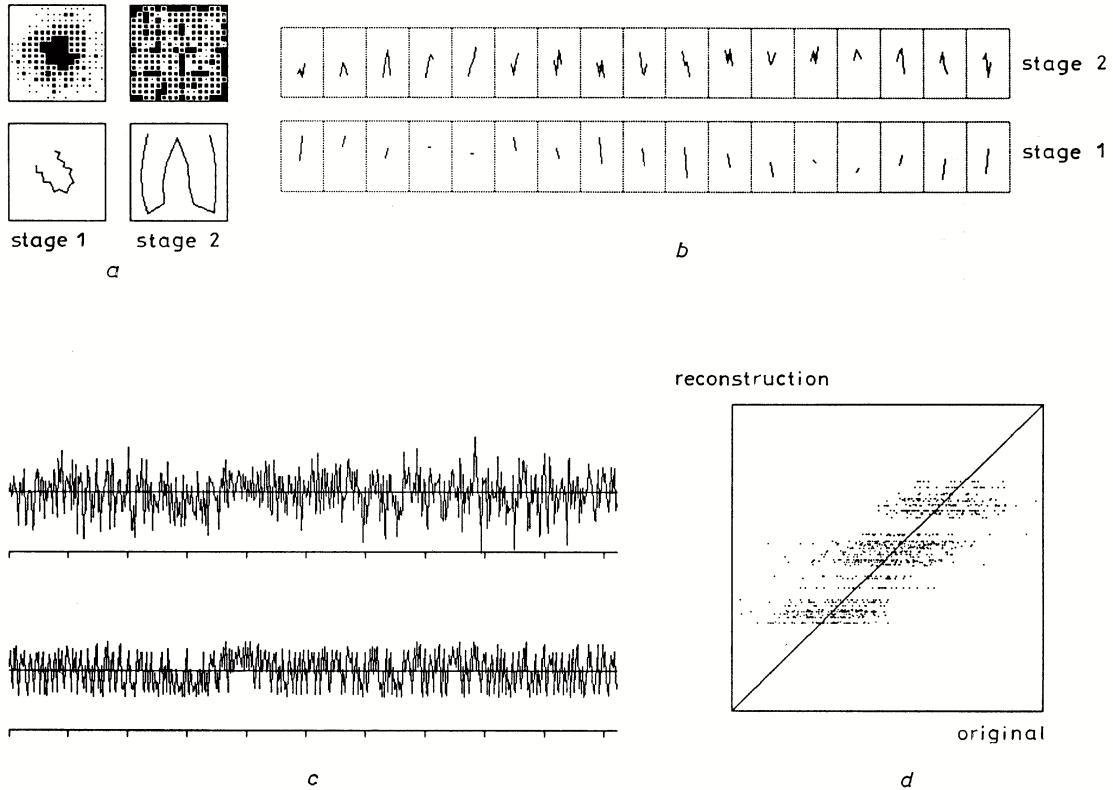


Figure 5: Two-stage encoding/decoding results for uncorrelated Gaussian noise input.

mean Gaussian variable with variance  $\frac{1}{1-\rho^2}$ , before iterating Equation 3.2 to generate the rest of the time series. Figure 6a is the correlated analogue of Figure 5a, and we observe that both the stage one and stage two co-occurrence matrices reveal correlation structure. The lines of code vectors accordingly adjust themselves to the correlations, as revealed by comparing Figure 5a and Figure 6a. This correlation structure also appears in Figure 6b. Figure 6c shows how the reconstruction is a very good representation of the original time series except for the positive and negative peaks which have been clipped, as in Figure 5c. The cross-co-occurrence matrix in Figure 6d is considerably narrower than that in Figure 5d, and the scaled  $L_2$  distortion is now 0.15, which is considerably less than in the uncorrelated case, as expected. This distortion is almost as good (0.5 dB loss) as the 8.8 dB which would be obtained from a single stage four-sample block encoder using 1 bit per sample [1]

### 3. Gamma noise ( $\nu = 0.5$ )

We generate this noise process as the square root of (the modulus of) an uncorrelated Gaussian time series. Figure 7a shows a stage one co-occurrence matrix that is peaked at the origin, because this type of noise is very spiky (mostly small values with the occasional large ex-

cursion). The stage two co-occurrence matrix is not flat, which implies that the stage one code vectors are not used with equal frequency. Figure 7b shows the same general type of result that we obtained in the uncorrelated Gaussian case of Figure 5b, and we observe that the reconstruction in Figure 7c is clipped. The cross-co-occurrence matrix in Figure 7d is quite broad, and we measure the distortion to be 0.36, which is significantly worse (1 dB loss) than the 5.4 dB that one would obtain with a single stage four-sample block encoder using one bit per sample [1]. In this example we can improve the performance by using a larger number of intermediate codes, although we do not present these results here.

### 4. Correlated Gaussian noise ( $\rho = 0.85$ ) using three-stage encoding

We shall now present results which are analogous to those that we discussed in Section III D 2, except that there are now three coding stages in the hierarchy. Thus eight input samples are encoded into four intermediate codes, which are further encoded into two intermediate codes, which are finally encoded into an output code. This is a direct extension of Figure 1b to three stages. Both stage one and stage two of the three-stage encoder have the same learning parameters as stage one of the

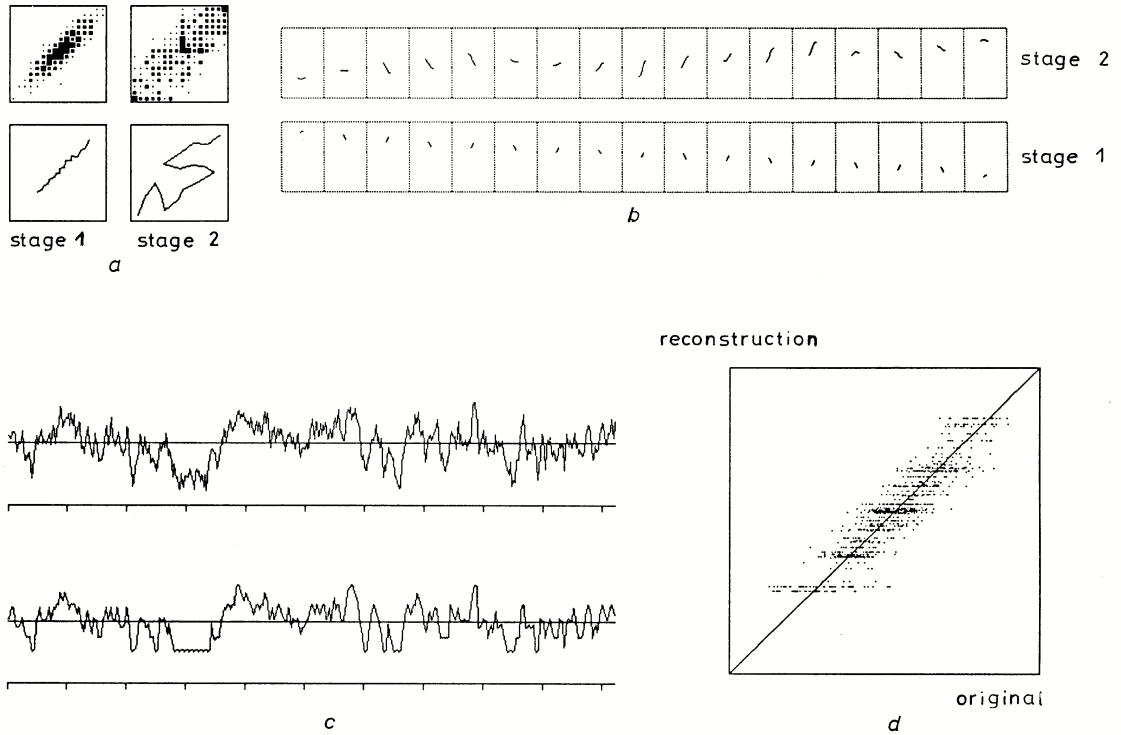


Figure 6: Two-stage encoding/decoding results for correlated Gaussian noise input. Correlation coefficient  $\rho = 0.85$ .

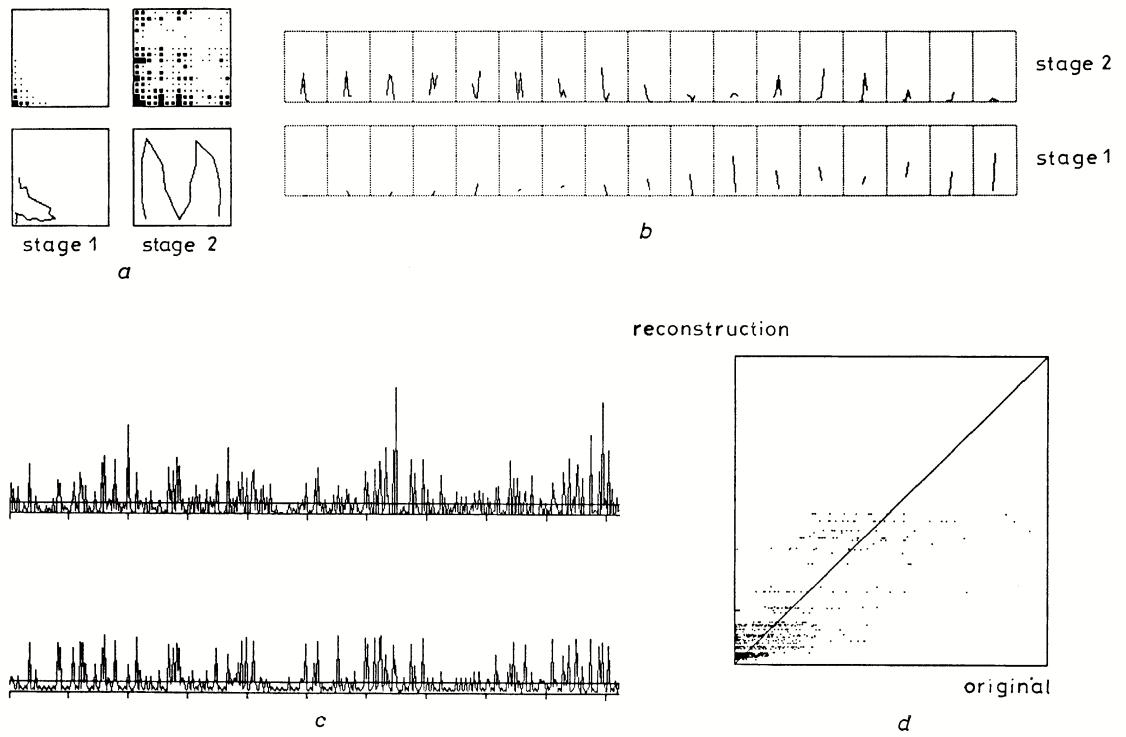


Figure 7: Two-stage encoding/decoding results for uncorrelated Gamma noise input order parameter  $\nu = 0.5$ .

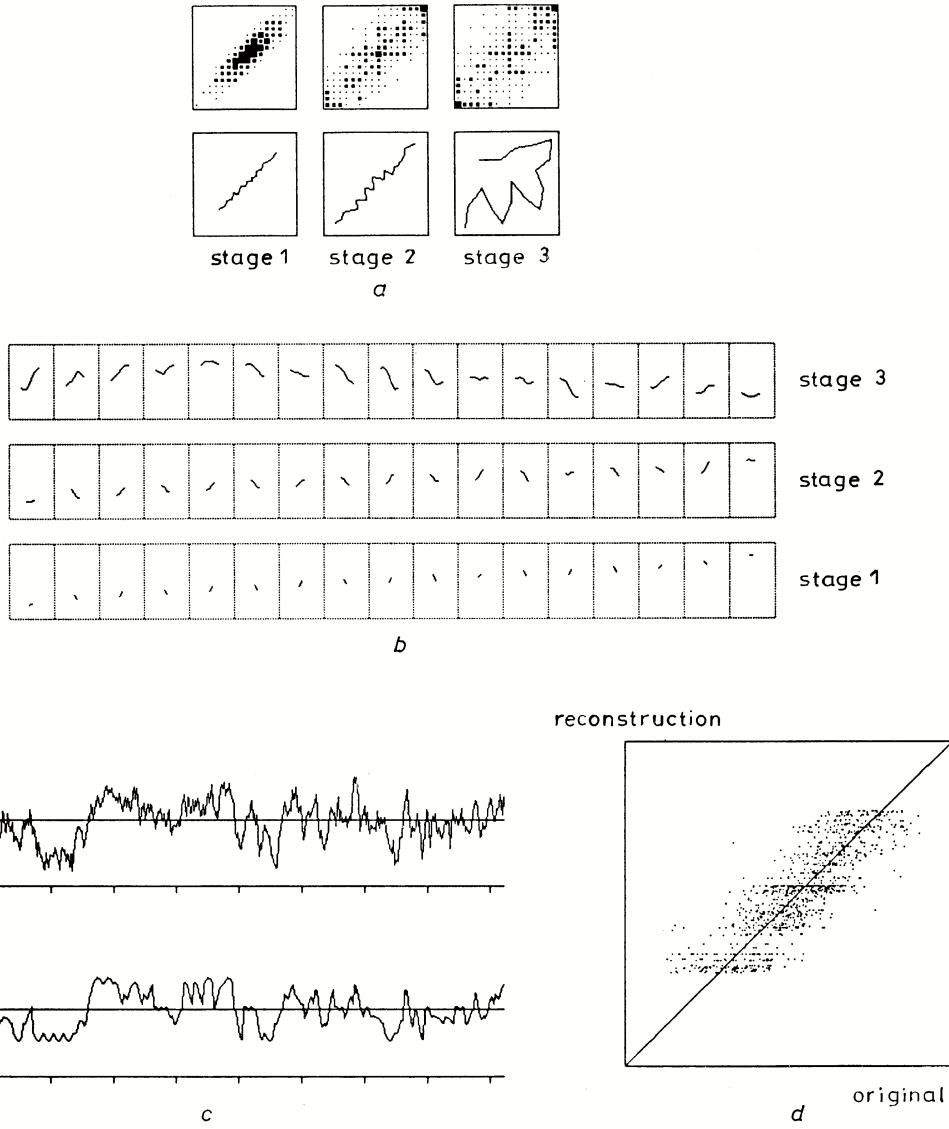


Figure 8: Three-stage encoding/decoding results for the same process as in Figure 6.

two-stage encoder used in Section III D 2, and the output stage remains the same. We continue to use translational symmetry to ensure that all of the encoders in a given stage of the hierarchy are the same. Figure 8a is analogous to Figure 6a. We now use  $\pi(i) \neq 0$  for  $i \neq 0$  when training the stage two code vectors, as required by the presence of a further encoding stage (i.e. stage three). The effect of  $\pi(i) \neq 0$  for  $i \neq 0$  on the stage two code vectors is the same as we observed in Figure 3 as the strength of  $\pi(\pm 1)$  was increased from zero. We still observe correlations in the stage three co-occurrence matrix of Figure 8a, although the correlations are somewhat weaker than in earlier encoder stages. We expect this because the typical time separation of the cause of the correlations (in the original time series) increases as one progresses upwards through the hierarchical coder. In Figure 8b the stage three results resemble eight sam-

ple pieces of time series, as expected. In Figure 8c and Figure 8d the results are somewhat worse than in Figure 6c and Figure 6d because we now encode using only about  $\frac{1}{2}$  bit per sample. We measure the distortion to be 0.31 in this particular case.

#### IV. SUMMARY AND CONCLUSIONS

We construct a multistage hierarchical vector quantiser by paying careful attention to how we train each stage. Thus we use a noise model to characterise the distorting effect of later encoding stages as seen from the point of view of earlier encoding stages. This requires us to use Kohonen's topographic mapping training algorithm to design all stages of the encoder except the last, which can be designed using the LBG algorithm. In Figure 4 we

see that each stage of a two-stage encoder can be trained in  $\mathcal{O}(10^3)$  updates: this is fast for a neural network training algorithm.

The most important practical result is the emergence of a fast hierarchical table look-up means of performing vector quantisation, with little loss in encoding accuracy in the examples that we have examined. In Figure 5 to Figure 8 the quality of the reconstructions leaves little to be desired, bearing in mind that we use one bit per sample coding (and  $\frac{1}{2}$  bit per sample in Figure 8).

Overall, our approach to vector quantisation shows great promise in improving the speed of the encoding operation without increasing the  $L_2$  reconstruction distortion too much in consequence, although there is much

scope for further computer experiments to be conducted. The hierarchical encoder of Figure 1b (which we use to produce the results in Figure 5 to Figure 7) can be re-alised as three table look-up operations, which is very fast.

As a simple generalisation we can apply our hierarchical approach to (2-dimensional) image data without modification. The image compression results which we have presented elsewhere [4, 7] were obtained with precisely the binary-tree hierarchical structure of Figure 1b, and its generalisation to more stages. The gain in speed that look-up tables provides is especially noticeable when compressing images.

- [1] N S Jayant and P Noll, *Digital coding of waveforms*, Prentice-Hall, Englewood Cliffs, 1984.
- [2] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [3] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [4] S P Luttrell, *Image compression using a neural network*, Proceedings of IGARSS conference on remote sensing (Edinburgh), ESA, 1988, pp. 1231–1238.
- [5] ———, *Self organising multilayer topographic mappings*, Proceedings of IEEE conference on neural networks (San Diego), IEEE, 1988, pp. 93–100.
- [6] ———, *Hierarchical self-organising networks*, Proceedings of IEE Conference on Artificial Neural Networks (London), IEE, 1989, pp. 2–6.
- [7] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
- [8] ———, *Self-organisation: a derivation from first principles of a class of learning algorithms*, Proceedings of IJCNN international joint conference on neural networks (Washington DC), IEEE, 1989, pp. 495–498.
- [9] L Miclet and M Dabouz, *Approximate fast nearest neighbour recognition*, Pattern Recognition Letters **1** (1983), no. 5-6, 277–285.



# Rapid Acquisition of Low Signal-to-Noise Carriers \*

S. P. Luttrell and J. A. S. Pritchard

*Royal Signals and Radar Establishment, St. Andrews Road, Malvern WR14 3PS, United Kingdom*

A parallel bank of 1-bit digital filters is proposed as a solution to the rapid carrier acquisition problem in satellite communications. The hardware is compact and cheap, and using a crude threshold detection criterion, it can localise a 30 dBHz carrier to within 100 Hz in a total bandwidth of 12 kHz in about 60 mS. A theoretical analysis of the system performance is also presented, together with predictions of its statistical behaviour, which will assist in the design of more sophisticated signal detection algorithms.

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 11, 2019.

This paper appeared in IEE Proceedings, Part I, 1989, vol. 136, no. 1, pp. 100-108. Paper 6463I (E9, E16), first received 18th January and in revised form 20th October 1988.

S P Luttrell's contribution to this paper was the theoretical analysis.

---

## I. LIST OF PRINCIPAL SYMBOLS

$t$	= time
$n, n', k, k'$	= sampling time indices
$t_n$	= sampling time
$[t_n, t_{n+1}]$	= sampling time interval
$V(t)$	= input voltage
$V_n$	= sampled input voltage
$l, l_1, l_2$	= Fourier component indices
$j$	= integer resonance condition
$B$	= bandwidth
$A$	= filter decay constant
$T$	= filter memory time
$\omega, \omega', \omega_1, \omega_2$	= angular frequencies
$\theta, \theta_1, \theta_2$	= phase shifts
$C(\omega_1, \omega_2)$	= cross correlation between square waves
$E(\omega_1, \omega_2)$	= modulus square of $C(\omega_1, \omega_2)$
$c_+, c_-$	= terms in $C(\omega_1, \omega_2)$
$x_1, x_2$	= scaled frequency in $C(\omega_1, \omega_2)$
$\Theta_n(\omega)$	= phase table for frequency $\omega$ at time $n$
$F_n(\omega)$	= filter output for frequency $\omega$ at time $n$
$R_{n,n'}(\omega)$	= filter impulse response
$\langle V_n V_{n'} \rangle$	= covariance of $V_n$
$\langle F_n(\omega) F_{n'}^*(\omega) \rangle$	= covariance of $F_n(\omega)$
$a^2$	= input noise variance
$b^2$	= input signal variance
$a'^2$	= output noise variance
$b'^2$	= output signal variance
$S_{in}$	= input signal-to-noise ratio
$S_{out}$	= output signal-to-noise ratio
$S'_{in}$	= $S_{in}$ in dBHz
$E$	= $ F_n(\omega) ^2$
$D$	= number of output samples averaged over
$E_D$	= average of $D$ samples of $E$
$\langle E_D \rangle$	= first moment of $P(E_D)$
$\langle (E_D)^2 \rangle$	= second moment of $P(E_D)$
$\text{Var}(E_D)$	= variance of $E_D$

$P(E)$	= PDF of $E$ (Rician)
$P_D(E_D)$	= PDF of $E_D$
$M_r$	= $r$ th moment of $P_D(E_D)$
$k$	= Fourier conjugate of $E$ (or $E_D$ )
$Q(k)$	= characteristic function of $P(E)$
$\xi$	= Lapace conjugate of $DE_D$
$\xi_+, \xi_-$	= limits of Bromwich integral
$L\{\cdot\}$	= Laplace transform operator
$J_\nu(z)$	= Bessel function
$I_0(z), I_\nu(z), I_{D-1}(z)$	= modified Bessel functions
$s$	= number of standard deviations in $E_D - \langle E_D \rangle$
$Q(b'^2, a'^2, s)$	= threshold $s$ standard deviations above $\langle E_D \rangle$
$P_{MD}(b'^2, a'^2, s), P_{MD}$	= probability of missed detection
$P_{FA}(a'^2, s), P_{FA}$	= probability of false alarm
$d$	= register word length

## II. INTRODUCTION

Geostationary satellites are now widely used for reliable long distance communications. In both business and military fields, the need for long range, flexible and portable communications is growing. For many users, the amount of information which needs to be communicated is relatively small, but often it is the portability of the remote terminal and the ease of use which is most important. The soldier may need to relay strategic information from the battlefield, or the mobile business user might wish to send back commercial information to his own country. The frequencies which are usually used for this kind of traffic are the military and civilian bands which lie between 7 and 13 GHz. These bands are high enough in frequency to support a large number of users, yet are not so high as to suffer high atmospheric attenuation or to demand prohibitively expensive microwave components.

To realise self-contained man-portable tactical terminals working at X-band using contemporary technology, a decrease in power with an attendant decrease in data rate is necessary and desirable [3, 5].

In the case of low data rate telegraph signals at the frequencies described, there will be a frequency ambiguity introduced over a microwave satellite link which can be orders of magnitude greater than the bandwidth of the signal. In a channel where the link is implemented most efficiently, no more power than that necessary to support error-free communications is used. In the examples which concern us, the signal-to-noise ratio in the band of ambiguity can be worse than  $-13$  dB. Consequently, for both the base station and the remote terminal, the rapid acquisition of the correct signal is a nontrivial task.

An X-band satellite transponder in geostationary orbit will typically introduce a maximum Doppler shift of about  $\pm 2$  kHz to the signal. The reference frequency uncertainty in the remote terminal, using a temperature

compensated crystal oscillator for a reference, is about  $\pm 3$  kHz. The total frequency ambiguity of the signal is therefore of the order of 10 kHz; the search bandwidth used (with a guard margin) is 12 kHz. Where the data rate is such that the modulated signal bandwidth is much less than the carrier centre frequency ambiguity, carrier acquisition using a conventional swept filter and integrator is no longer satisfactory. For example, the data modulation scheme which is used in the man-portable remote terminal prototype known as the ‘Manpack’ [3, 5] is 50 bits per second coherent differential phase shift keying (DPSK). The time taken to acquire a signal that has a frequency error of  $\pm 6$  kHz using this swept filter technique is between 1 and 40 seconds, depending on the position of the signal in the band. Clearly this is not satisfactory for tactical communications when a brief transmission is desirable, or in environments where signal fading occurs.

Recent studies have shown [2] that rapid acquisition of such signals, using the most advanced known techniques, requires methods which are in general complicated, expensive, or bulky, and in any case are not satisfactory in terms of acquisition time. State of the art methods implement successive approximations using a chirp correlator [2]. Phase lock to a 50 baud DPSK signal in a 12 kHz band may be achieved in 7 seconds, regardless of the carrier centre frequency within the band. This delay in signal acquisition is still not satisfactory for our exacting applications of low data rate links.

In this paper, we take advantage of the fact that the full apparatus of the Fourier transform is not required if we wish only to find a signal carrier frequency. We need only an approximation, and so we emulate a parallel bank of filters using simple serial hardware which is constructed from cheap and readily available digital components. The scheme which we describe will enable lock to be achieved at least an order of magnitude more quickly than before.

In Section III we present a theoretical analysis of a 1-bit quantised single pole recursive filter bank. We de-

rive the response and aliasing properties of such filters. We also derive some statistical properties of the filter outputs; these may be used to predict performance in a noisy environment. In Section IV we describe how fast serial hardware can be built to emulate the parallel bank of filters, and we multiplex many filters into the wide digital bandwidth which is available in a single serial circuit. The filter outputs (which are a crude representation of the Fourier transform of the input signal) are then sampled using a microprocessor. In Section V we present the results of testing the circuit.

### III. THEORY

The primary goal of this research is to design hardware which can rapidly localise a carrier frequency, but which at the same time is compact and inexpensive. We shall assume that the ‘noise environment’ is benign (i.e. no large amplitude spurious coherent signals are present), and that the signal-to-noise ratio (SNR) is small ( $\leq 30$  dBHz). Under these assumptions, 1-bit quantisation of the input signal retains nearly all the signal information, and single (complex) pole recursive filtering of the 1-bit signal is sufficient to extract the required carrier frequency.

We will therefore develop a theory which describes the operation of a bank of single pole recursive filters with 1-bit quantised inputs. The analogue input signal  $V(t)$  comprises a set of modulated carrier signals plus noise in general, which is bandpass filtered to a band  $[0, B]$  and sampled at a sufficient rate to avoid aliasing. We assume throughout that Nyquist sampling for the band  $[-B, B]$  is used, so that the sample times  $t_n$  are given by

$$t_n = \frac{n\pi}{B} \quad (3.1)$$

where the corresponding analogue input signal value is denoted by  $V_n$ . We also assume that the noise accompanying the signal is uncorrelated from sample to sample.

1-bit quantisation of  $V_n$  is defined as follows:

$$V_n \rightarrow \begin{cases} 1, & V_n \geq 0 \\ -1, & V_n \leq 0 \end{cases} \quad (3.2)$$

The effect of which is to preserve most of the information content of the analogue input signal because the noise acts as a ‘dither signal’. There is, however, an aliasing problem associated with such 1-bit quantisation because it turns sine waves into square waves which have out-of-band harmonics. The Fourier decomposition of a square wave  $V(t)$  of frequency  $\omega$  is

$$V(t) = \sum_l \frac{4}{\pi l} \sin[l\omega t] \quad (3.3)$$

where the summation is over odd integers  $l$ , and the  $l$ th harmonic is suppressed in amplitude by a factor  $l$ . If we use 2-(or more) bit quantisation, the harmonic amplitudes decay faster, but we will see later that the effects of aliasing in the 1-bit case are acceptable.

We now introduce our basic system model by defining a single pole recursive filter with input  $V_n$  as

$$F_{n+1}(\omega) = A F_n(\omega) + \Theta_n(\omega) V_n \quad (3.4)$$

where  $F_n(\omega)$  is the complex amplitude response of the filter tuned to  $\omega$ ,  $A$  is the relaxation factor for the filter, and  $\Theta_n(\omega)$  is a 1-bit quantised (in each of the real and imaginary parts) version of the complex exponential

$$\Theta_n(\omega) \equiv \exp \left[ \frac{i n \pi \omega}{B} \right] \quad (3.5)$$

The characteristic memory time  $T$  of the filter measured in sample intervals is simply related to  $A$  by

$$T = \frac{1}{1 - A} \quad (3.6)$$

The solution to Equation 3.4 is the filter response  $F_n(\omega)$  which is given by

$$F_n(\omega) = \sum_{n'=-\infty}^n R_{n,n'}(\omega) V_{n'} \quad (3.7)$$

where the impulse response  $R_{n,n'}(\omega)$  is given by

$$R_{n,n'}(\omega) = \begin{cases} A^{n-n'} \Theta_{n'}(\omega), & n' \leq n \\ 0, & n' > n \end{cases} \quad (3.8)$$

where both  $V_{n'}$  and  $\Theta_{n'}(\omega)$  are 1-bit quantised, but  $A$  and  $F_n(\omega)$  are represented using many bits. Equation 3.7 for  $F_n(\omega)$  may be interpreted as a weighted cross-correlation between the past input signal  $V_n$  and the  $\Theta_n(\omega)$ , and the weighting factor  $A^{n-n'}$  gives rise to the filter memory time  $T$ .

We will now verify that the aliasing, which we alluded to earlier, is inconsequential in the case of 1-bit quantisation. The response to an input sine wave at frequency  $\omega_1$  of a filter (as defined in Equation 3.4) tuned to  $\omega_2$  is obtained by calculating the weighted cross-correlation  $C(\omega_1, \omega_2)$  between the two corresponding sampled square waves (as defined in Equation 3.3). This is given by

$$C(\omega_1, \omega_2) = \sum_{n=0}^{\infty} A^n \sum_{\substack{l_1 \\ \text{odd}}} \sum_{l_2} \frac{16}{\pi^2 l_1 l_2} \sin \left[ \frac{(n + \theta_1) \pi l_1 \omega_1}{B} \right] \exp \left[ \frac{i(n + \theta_2) \pi l_2 \omega_2}{B} \right] \quad (3.9)$$

where  $\theta_1$  and  $\theta_2$  are arbitrary phase shifts. Summing over  $n$  then yields

$$C(\omega_1, \omega_2) = \sum_{\substack{l_1 \\ \text{odd}}} \sum_{l_2} \frac{8}{i\pi^2 l_1 l_2} [c_+ + c_-] \quad (3.10)$$

where

$$c_{\pm} \equiv \pm \frac{1 - A \exp \left[ \mp \frac{i\pi(l_1\omega_1 \pm l_2\omega_2)}{B} \right]}{1 + A^2 - 2A \cos \left[ \frac{\pi(l_1\omega_1 \pm l_2\omega_2)}{B} \right]} \exp \left[ \pm \frac{i\pi(\theta_1 l_1 \omega_1 \pm \theta_2 l_2 \omega_2)}{B} \right] \quad (3.11)$$


---

This response is large when one or both of the denominator factors in the  $c_{\pm}$  are small, which occurs when

$$\begin{aligned} l_1 x_1 \pm l_2 x_2 &= 2j \\ l_1 l_2 &= 1, 3, 5, \dots \\ j &= 0, \pm 1, \pm 2, \dots \\ 0 \leq x_i &\leq 1 \end{aligned} \quad (3.12)$$

where  $x_i$  ( $0 \leq x_i \leq 1$ ) is the scaled frequency  $\frac{\omega_i}{B}$ . There are many solutions to this equation, some of which we will now tabulate.

$l_1 = l_2 = 1$  solutions:

$$j = 0 \quad x_1 = x_2 \quad 0 \leq x_1, x_2 \leq 1$$

$l_1 = 1, l_2 = 3$  solutions:

$$j = -1 \quad x_1 = 3x_2 - 2 \quad 0 \leq x_1 \leq 1 \quad \frac{2}{3} \leq x_2 \leq 1$$

$$j = 0 \quad x_1 = 3x_2 \quad 0 \leq x_1 \leq 1 \quad 0 \leq x_2 \leq \frac{1}{3}$$

$$j = 1 \quad x_1 = -3x_2 + 2 \quad 0 \leq x_1 \leq 1 \quad \frac{1}{3} \leq x_2 \leq \frac{2}{3}$$

$l_1 = 3, l_2 = 1$  solutions: Obtain by interchanging  $x_1$  and  $x_2$  in the solutions for the  $l_1 = 1, l_2 = 3$  case.

The  $l_1 = 1, l_2 = 1$  case is ideally the only solution, and we will define its amplitude as unity. The contribution of particular  $(l_1, l_2)$  solution to  $C(\omega_1, \omega_2)$  is suppressed by factor  $l_1 l_2$ , and so the dominant cases are those in which the fundamental frequency cross correlates with the third harmonic, and which therefore have a typical amplitude contribution of  $\frac{1}{3}$ . The worst case occurs when  $(x_1, x_2)$ , coincidentally satisfies two of the above equations, for then the interference between the crosscorrelation terms can lead to an amplitude contribution of  $\frac{2}{3}$ . However, there are limited number of such coincidental solutions which are  $(x_1, x_2) = (\frac{1}{5}, \frac{3}{5}), (\frac{1}{4}, \frac{3}{4}), (\frac{2}{5}, \frac{4}{5}), (\frac{3}{5}, \frac{1}{5}), (\frac{3}{4}, \frac{1}{4}), (\frac{4}{5}, \frac{2}{5})$ . We have ignored the solutions with  $x_1 = 0$  or  $x_2 = 0$  because

we are not concerned with DC behaviour; and we have ignored the solution  $x_1 = x_2 = \frac{1}{2}$  because the sum over contributions from all  $(l_1, l_2)$  yields no adverse interference (this is most easily seen in terms of crosscorrelated square waves directly). In general, higher order harmonics will produce further spurious contributions to the amplitude response which are suppressed by larger factors than is the third harmonic.

In summary, for 1-bit quantisation, provided that the existence of spurious responses (i.e. aliasing) is acknowledged, the outputs of a bank of single pole recursive filters may be easily interpreted if the number of signals present is not too large. Usually there is only a single signal present, and so the interpretation problem is trivial.

With these provisos in mind we will now derive approximations to the squared filter response and its half-width by approximating the weighted crosscorrelation  $C(\omega_1, \omega_2)$  in the vicinity of  $\omega_1 = \omega_2$  using only the first term of the  $(l_1, l_2)$  sum in Equation 3.9. Thus

$$\begin{aligned} C(\omega_1, \omega_2) &\simeq \frac{8}{i\pi^2} c_- (l_1 = l_2 = 1) \\ &= \frac{8i}{\pi^2} \frac{\exp \left[ -\frac{i\pi(\theta_1 \omega_1 - \theta_2 \omega_2)}{B} \right]}{1 - A \exp \left[ -\frac{i\pi(\omega_1 - \omega_2)}{B} \right]} \end{aligned} \quad (3.13)$$

The numerical factor  $\frac{8}{\pi^2}$  is unimportant, and so (up to a constant factor) this result is what would be obtained if we were to ignore the 1-bit quantisation altogether; we shall henceforth use this approximate equivalence to simplify our calculations. We may thus approximate the squared response  $E(\omega_1, \omega_2)$  by

$$\begin{aligned} E(\omega_1, \omega_2) &\equiv |C(\omega_1, \omega_2)|^2 \\ &\simeq \frac{1}{(1 - A)^2 + A \left[ \frac{\pi(\omega_1 - \omega_2)}{B} \right]^2} \end{aligned} \quad (3.14)$$

which is a Lorentzian with a scaled half-width given by

$$\begin{aligned} \frac{|\omega_1 - \omega_2|}{B} &\simeq \frac{1-A}{\pi} \\ &= \frac{1}{\pi T} \end{aligned} \quad (3.15)$$

In this paper, we will deploy 128 filters across the band  $[0, B]$ , and so the scaled half-width of each filter must be  $\frac{1}{256}$ , thus  $T = \frac{256}{\pi} \simeq 82$  samples.

To analyse the performance of our system in more detail we now derive some of its higher order statistical properties. From Equation 3.7 we obtain

$$\langle F_n(\omega) F_{n'}^*(\omega) \rangle = \sum_{k=-\infty}^n \sum_{k'=-\infty}^{n'} R_{n,k}(\omega) R_{n',k'}^*(\omega) \langle V_k V_{k'} \rangle \quad (3.16)$$

where  $R_{n,k}(\omega)$  is given in Equation 3.8 and the angle brackets  $\langle \dots \rangle$  denote time averaging. When  $V(t)$  is zero mean white Gaussian noise with covariance  $\langle V_n V_{n'} \rangle = a^2 \delta_{n,n'}$  we obtain

$$\begin{aligned} \langle F_n(\omega) F_{n'}^*(\omega) \rangle &= a^2 \sum_{k=-\infty}^{\min(n,n')} R_{n,k}(\omega) R_{n',k}^*(\omega) \\ &= a^2 \frac{A^{|n-n'|}}{1-A^2} \end{aligned} \quad (3.17)$$

$F_n(\omega)$  is zero mean, and so  $\langle F_n(\omega) F_{n'}^*(\omega) \rangle$  measures its covariance, thus we see that  $F_n(\omega)$  decorrelates on a time scale  $|n - n'|$  such that  $A^{|n-n'|} \leq 1$ . Using Equation 3.6, this reduces to  $|n - n'| \geq T$ . We henceforth assume that the filter output is sampled at time intervals  $T$  ( $= 82$  samples), and call this scheme ‘sparse sampling’.

We now compare the squared response to both signal and noise. First, for zero mean white Gaussian noise with input variance  $a^2$ , we obtain the output variance  $a'^2$  from Equation 3.17 (the  $n = n'$  case) as

$$a'^2 = \frac{a^2}{1-A^2} \quad (3.18)$$

Secondly, for a unit amplitude input sine wave with frequency  $\omega'$ , we obtain

$$\begin{aligned} F_n(\omega) &= \sum_{n'=-\infty}^n A^{n-n'} \Theta_{n'}(\omega) \sin\left(\frac{n'\pi\omega'}{B} - \theta\right) \\ &\simeq \frac{ie^{i\theta}}{2} \frac{e^{\frac{i\pi(\omega-\omega')}{B}}}{1-A e^{\frac{i\pi(\omega-\omega')}{B}}} \end{aligned} \quad (3.19)$$

This result should be compared with Equation 3.13. By modulus squaring these amplitudes we may relate the sine wave results to the Gaussian noise results. Thus we obtain the input and output variances  $b^2$  and  $b'^2$ , respectively

(these are analogous to  $a^2$  and  $a'^2$ ) as

$$\begin{aligned} b^2 &\equiv \left\langle \sin^2\left(\frac{n\pi\omega}{B} - \theta\right) \right\rangle \\ &= \frac{1}{2} \\ b'^2 &\equiv \left\langle |F_n(\omega)|^2 \right\rangle \\ &= \begin{cases} \frac{1}{4(1-A)^2} & (\text{best case}) \\ \frac{1}{8(1-A)^2} & (\text{worst case}) \end{cases} \end{aligned} \quad (3.20)$$

The best case occurs when  $\omega = \omega'$  and the worst case occurs when  $|\omega_1 - \omega_2|$  is a Lorentzian half-width (see Equation 3.15); in the worst case the sine wave causes two adjacent filters to respond each with half the best case result.

From Equation 3.18 and Equation 3.20 we may derive the relationship between the input and output signal-to-noise ratios ( $S_{in}$  and  $S_{out}$  respectively). Thus

$$\begin{aligned} S_{out} &\equiv \frac{b'^2}{a'^2} \\ S_{in} &\equiv \frac{b^2}{a^2} \\ S_{out} &= \frac{1}{2} \frac{1+A}{1-A} S_{in} \end{aligned} \quad (3.21)$$

For the particular hardware which we have constructed, the relevant figures are  $T \simeq 81.5$  samples, and so  $A \simeq 0.98773$  and  $\frac{S_{out}}{S_{in}} \simeq 81$ . Note that  $S_{out} \simeq T S_{in}$  because the memory  $T$  of the filter coherently combines approximately  $T$  sine wave input samples, whereas it only incoherently combines the same number of Gaussian noise input samples.

We may re-express these results in an alternative form where  $S_{in}$  is expressed in dBHz (which we denote as  $S'_{in}$ ).  $S_{in}$  and  $S'_{in}$  are then related by

$$S'_{in} = 10 \log_{10}(BS_{in}) \quad (3.22)$$

whence from Equation 3.21 and Equation 3.22

$$S_{out} = \frac{1}{2} \frac{1+A}{1-A} \frac{10^{\frac{S'_{in}}{10}}}{B} \quad (3.23)$$

For our hardware,  $B = 12\text{kHz}$ ,  $T = 81.5$ ,  $A = 0.98773$ , and so Equation 3.23 reduces to  $S_{out} = 0.0067 \times 10^{\frac{S'_{in}}{10}}$ .

We have presented results for the statistical properties of the filter response to a sine wave and to Gaussian noise, considered separately. We will now extend these results to the (more realistic) case of a sum of a sine wave plus Gaussian noise. In this case, the statistical properties of the squared response are not completely described by means and covariances, and so we resort to a treatment using a full probability density function (PDF).

The probability density function of the squared filter response  $E \equiv |F_n(\omega)|^2$  is given by a Rician distribution

$$P(E) = \frac{1}{a'^2} \exp\left\{-\frac{(E+b'^2)}{a'^2}\right\} I_0\left\{\frac{2E^{\frac{1}{2}}b'}{a'^2}\right\} \quad (3.24)$$

where  $I_0(z)$  is the zeroth order modified Bessel function of the first kind. This result is strictly correct only for complex Gaussian noise, but the approximation is good for real-valued noise when  $T \gg \frac{2\pi}{\omega}$  (i.e. for all filters other than those near DC). Independent observations of  $E$  are obtained by using the sparse sampling scheme which was designed for independent observations of  $F_n(\omega)$ . Strictly,  $E$  decorrelates twice as rapidly as  $F_n(\omega)$ , and so the sparse sampling scheme errs on the

side of caution for the purposes of an average squared response.

We implement an averaging procedure in the hardware, and so we must also derive the PDF of the averaged squared response. Thus, defining  $E_D$  as the average of  $D$  independent observations of  $E$ , we obtain (see Appendix A for a detailed derivation of Equation 3.25 and Equation 3.26)

$$P_D(E_D) = \frac{D}{a'^2} \left( \frac{E_D}{b'^2} \right)^{\frac{D-1}{2}} \exp \left\{ -\frac{D(E + b'^2)}{a'^2} \right\} I_{D-1} \left\{ \frac{2DE_D^{\frac{1}{2}}b'}{a'^2} \right\} \quad (3.25)$$

whence the first two moments of  $P_D(E_D)$  are given by

$$\begin{aligned} \langle E_D \rangle &= a'^2 + b'^2 \\ \langle (E_D)^2 \rangle &= \left(1 + \frac{1}{D}\right) a'^4 + 2 \left(1 + \frac{1}{D}\right) a'^2 b'^2 + b'^4 \end{aligned} \quad (3.26)$$

thus the normalised variance of  $E_D$  is given by

$$\frac{\text{Var}(E_D)}{\langle E_D \rangle^2} = \frac{a'^2(a'^2 + 2b'^2)}{D(a'^2 + b'^2)^2} \quad (3.27)$$

The statistics in Equation 3.26 and Equation 3.27 provide a simpler description of the properties of  $E_D$  than does the full PDF  $P_D(E_D)$ . This simplicity is obtained at the cost of sacrificing knowledge of the higher order statistical properties of the nonGaussian  $P_D(E_D)$ .

For practical applications we must define a suitable threshold for signal detection in the presence of noise. The average squared output  $E_D$  follows a distribution  $P_D(E_D)$  of values which depend on both the signal level  $a^2$  and the noise level  $b^2$ . Loosely speaking, a signal is deemed to be present if very little of the upper tail of  $P_D(E_D)$  (without signal present) lies above the value of  $E_D$  which is actually observed. We will quantify this by defining a threshold function

$$Q(b'^2, a'^2, s) \equiv \langle E_D \rangle \left\{ 1 + \frac{s(\text{Var}(E_D))^{\frac{1}{2}}}{\langle E_D \rangle} \right\} \quad (3.28)$$

which is the value that  $E_D$  would have if it were displaced by  $s$  standard deviations from its mean  $\langle E_D \rangle$ . Assuming that the effect of the signal appears in a single filter response, the probability  $P_{MD}$  of a missed detection is given by

$$P_{MD}(b'^2, a'^2, s) = \int_0^{Q(b'^2, a'^2, s)} dE_D P_D(E_D) \quad (3.29)$$

Assuming that the averaged squared responses of all 128 filters are independent in the presence of pure noise, the

probability  $P_{FA}$  of a false alarm in one or more of the 128 filters is given by

$$P_{FA}(a'^2, s) = 1 - \left[ \int_0^{Q(0, a'^2, s)} dE_D P_D(E_D) \right]^{128} \quad (3.30)$$

These assumptions about the filter responses are not strictly true in practice: the signal usually affects more than one filter response, and the responses of the filters to pure noise are not independent when they are sparse sampled according to the decorrelation time  $T$  of a single filter (i.e. the filter responses overlap).

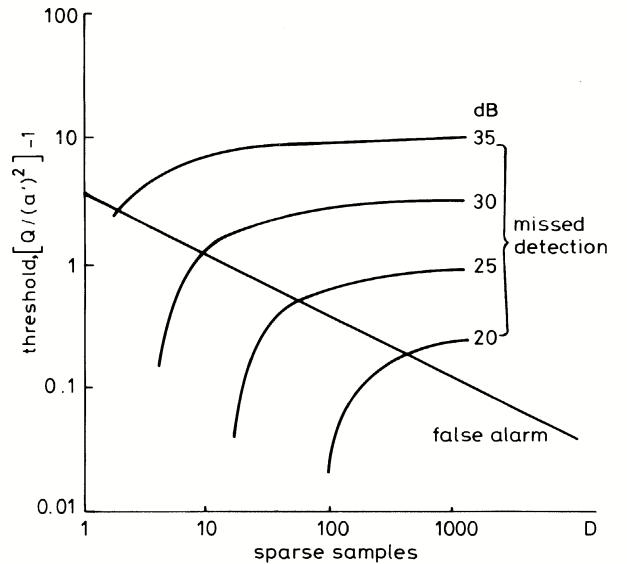


Figure 1: Plot of thresholds against  $D$  for various  $S'_{in}$  for a 1% missed detection and 1% false alarm probability.

We may obtain a useful approximation to the above signal detection results by assuming that  $D$  is large enough to enable us to use a Gaussian approximation for

$P_D(E_D)$ . The integrals in Equation 3.29 and Equation 3.30 may then be looked up in a table of error functions. Thus we may select  $P_{MD}$  and  $P_{FA}$  and deduce suitable values of  $s$  for various  $\frac{b'^2}{a'^2}$  and  $D$ . In Figure 1 we show plots of the threshold function for  $P_{MD} = P_{FA} = 0.01$ . Note that we have subtracted  $a'^2$  from all the threshold values to make the plots clearer. The false alarm line marks the lower bound of permitted thresholds (as a function of  $D$ ) which guarantee  $P_{FA} \leq 0.01$ , and each missed detection curve marks the upper bound of permitted thresholds which guarantee  $P_{MD} \leq 0.01$  for a given  $S'_{in}$ . Combining these inequalities reveals that for each signal-to-noise ratio there is a minimum value of  $D$  for which the missed detection and false alarm criteria are simultaneously satisfied.

#### IV. HARDWARE

We wish to design the hardware in such a way that it most simply reflects the processes required to emulate a bank of single (complex) pole recursive filters, as defined in Equation 3.4. We envisage that this hardware device will be used as a front-end processor for a microprocessor as detailed in [4].

Figure 2 shows the block diagram of a simple realisation of the circuit. Because we are using 1-bit quantisation of the analogue input voltage  $V(t)$ , only the sign of sample  $V_n$  is retained; this may be achieved using a comparator and a latch. The latch is triggered each time a new sample is required for processing.

The  $\Theta_n(\omega)$  have the general form  $\exp[\frac{i n \pi \omega}{B}]$  (see Equation 3.5) and hence may be generated by sine and cosine functions. One bit quantisation suffices, and so look-up tables are chosen. This has the added advantage of retaining complete flexibility of choice in the single bit functions which may be used. Care must be taken to eliminate the DC component by alternating the sign of the ‘zero’ crossing points of each of the functions. The single bit representations are stored in an 8 K by 8 EPROM, where the eight outputs are multiplexed to use it as a 32 K  $\times$  2 memory. The 32 K is configured as 128 sets of 256 tables which can be accessed sequentially. We call this look-up table the ‘phase table’.

Each  $F_n(\omega)$  is represented by  $2d$  bits ( $d$  for each of the real and imaginary parts), where  $d$  is determined by the value of  $T$  which is used. A suitable value of  $d$  is obtained by calculating  $a'^2$  from Equation 3.18 using appropriate values of  $a^2$  and  $A^2$  to estimate the mean of the squared noise response. For a 1-bit quantised input,  $a^2$  is forced to satisfy  $a^2 \simeq 1$ , and so from Equation 3.18 we obtain  $a'^2 \simeq 8$ . A suitable value of  $d$  should accommodate this squared response, and so we choose  $d = 4$ . The simplest practical method of implementing the transformation  $F_n(\omega) \rightarrow AF_n(\omega)$  for  $d < 16$  is by using tables, which we call ‘decrement tables’. These tables are realised as a  $2^d \times d$  EPROM; this would also permit us to use a more complicated memory term than  $AF_n(\omega)$ , if

required. The 4-bit quantisation of  $AF_n(\omega)$  is sufficient to accomodate  $a'^2$ , but is nevertheless rather coarse and leads to difficulties in implementing the required decrement factor  $A$ . We therefore introduce a further 4 bits to accommodate a binary fraction correction to the leading 4 bits. This may be viewed equivalently as an 8-bit representation, where the upper 4 bits are used to record the (1-bit) output of the multiplier, and the lower 4 bits are used to attain the required finesse for the decrement operation.

The decrement tables may be generated by calculating a decaying exponential using real numbers, which is then rounded to produce 8-bit entries for the table. Care must be taken to hard limit the extrema of the tables to avoid overflow or underflow in the single pole recursive filters. It should be noted that the 8-bit approximation used in these tables is a very coarse representation which divides an exponential decay into five regions: there is a band around zero where there is no decay, there are two hard limiting regions at the positive and negative extremes, and there are two bands in between where the decrement is one least significant bit.

As the input is a time series  $V_n$ , consisting only of 1 and 0 (or equivalently 1 and -1), a complex multiplication to generate  $\Theta_n(\omega)V_n$  in Equation 3.4 is not necessary. Two real multiplications are used, each of which reduces to an exclusive NOR with 1-bit quantisation. These outputs are added to  $AF_n(\omega)$  as in Equation 3.4. In practical realisations of the circuit, an exclusive OR gate is used together with two’s complement addition at the adder. This inverts  $F_n(\omega)$  in the complex plane twice (leaving it unchanged) and simplifies the hardware. If the output of the multiplier is 0, then  $AF_n(\omega)$  is incremented, and if the output is 1, then  $AF_n(\omega)$  is decremented; the new value  $F_{n+1}(\omega)$  is stored in the register. The circuit shown in Figure 2 is designed to operate in a ‘lock step’ mode. The value  $F_n(\omega)$  is latched at the output of its register, and the decrement table is then allowed to settle. Simultaneously, the input  $V_n$  is multiplied with the corresponding output of the phase table and this too is allowed to settle.

If the digital bandwidth available exceeds the analogue bandwidth which is being analysed, then it is advantageous to use the same circuit in parallel to calculate  $F_n(\omega)$  for several frequencies. The elements of the block diagram which are represented as three dimensional shapes are blocks of memory where the extra dimension represents  $\omega$ . As a single input sample  $V_n$  is held in the input latch, a counter scans through the registers and the look-up tables. In any particular application we would seek to accommodate the update of all  $F_n(\omega)$  within the input sample time interval  $[t_n, t_{n+1}]$  (see Equation 3.1).

We arrange 128 filters to be evenly spaced in frequency so that they span the range  $[0, B]$ ; the phase table entries then repeat with a period equal to that of the tables in the lowest frequency filter. In the example, the filter widths are 94 Hz ( $= \frac{12\text{kHz}}{128}$ ), and so the lowest frequency filter is

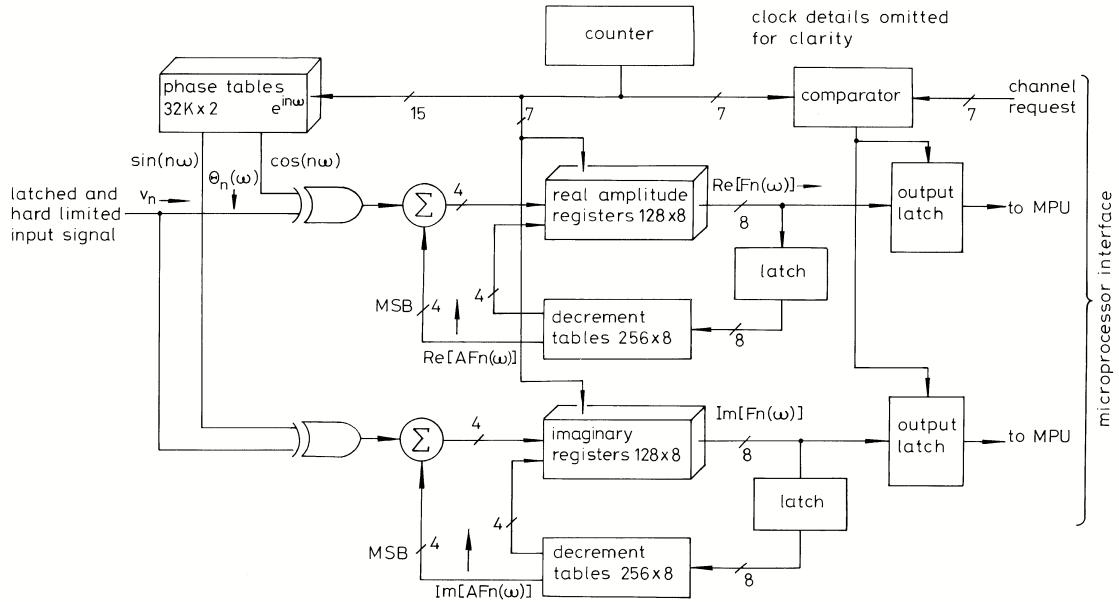


Figure 2: Block diagram of the complex pole recursive filter hardware.

centred on 47 Hz, and the repeat period is 21 ms. This determines the length of the phase table whose elements are accessed cyclically as the input samples arrive.

The real and imaginary parts of each  $F_n(\omega)$  may be multiplexed into the same memory to save space and hardware, but this results in a speed and timing complexity penalty. For the breadboard circuit, the real and imaginary parts were separated for simplicity.

When an external processor requires a sample  $F_n(\omega)$ , the address of the register is placed on the channel request lines shown in Figure 2. When the counter next reaches this address, a pulse is output from the comparator which loads the current value of  $F_n(\omega)$  into a second set of latches. The contents of these may be examined at any time before the next value of  $F_n(\omega)$  is loaded.

In the prototype circuit, we use a switch on the channel request lines and direct the real and imaginary outputs to 8-bit digital-to-analogue convertors. The complex value of any one of the  $F_n(\omega)$  can then be viewed by using it to drive the  $x$  and  $y$  inputs of an oscilloscope to create an Argand diagram (or Lissajous figure). This proves to be a very useful representation for diagnostic purposes.

The prototype circuit is designed to reduce the centre frequency ambiguity of a low data rate signal in a satellite band. The width of the signal in the band is about 100 Hz (which is commensurate with the width 94 Hz of each digital filter), and the width of the band is 12 kHz. The communications band is translated down to baseband (0-12 kHz) and Nyquist sampled, followed by input to each of the 128 filters. This requires a clock frequency of 3.072 MHz ( $128 \times 24$  kHz) which is well within the scope of TTL logic. The maximum propagation delay around the recursive filter loop is therefore 325 ns, as is the propagation delay from the counters through the

phase tables, exclusive OR gates and adders to the registers. This allows 150 ns 8K  $\times$  8 CMOS EPROMS to be used for the phase and decrement tables, although faster (50 ns) random access memories have to be used for the registers.

The memory which we use for the decrement tables is only  $256 \times 8$  in size. We note that the use of a larger memory could absorb the adder and multiplier functions by connecting extra input lines on the EPROM to the sample input  $V_n$  and the phase tables  $\Theta_n(\omega)$ , and then adjusting the decrement table appropriately.

## V. RESULTS

To help understand the behaviour of the filter which we have constructed, it is instructive to view the Argand plane representation of their state spaces during operation. The display produced shows which states of the filter have been accessed. The more frequently accessed states are brighter, although the transition sequence is not exactly enumerable from the display.

Figure 3 shows photographs of the response of a typical filter to the frequency at the centre of its band in varying signal-to-noise conditions. The noise is Gaussian and covers the input band which is 0-12 kHz. At 50 dBHz the registers are fully saturated and there is barely enough noise to create a 'dither' signal. As the signal-to-noise ratio is decreased to 30 dBHz, the profile of the noise response becomes apparent. At 20 dBHz the display structure is similar in appearance to that generated by pure noise; the signal perturbs the probability distribution of the states by much less than the width of the noise response. For the processor to work well, we re-

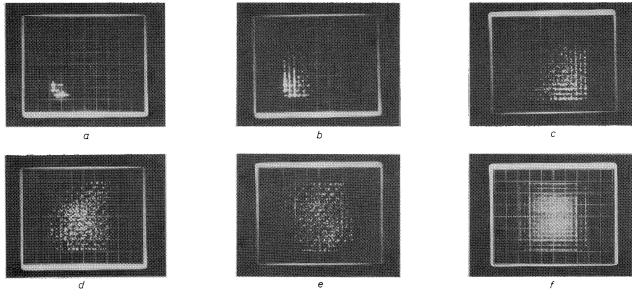


Figure 3: Response of a filter to various noise levels at centre frequency. (a) 50 dBHz. (b) 40 dBHz. (c) 30 dBHz. (d) 20 dBHz. (e) noise. (f) noise averaged for 4s (4s exposure).

quire that the noise response is contained mostly within the allowed states within the Argand plane at low signal-to-noise ratios. This condition is manifestly satisfied, and so the choice  $d = 4$  is adequate. All exposures in Figure 3 are for  $\frac{1}{30}$  second except for Figure 3f, which shows noise averaged (by the film) for 4 seconds. In Figure 3f, the striations apparent near the edges of the photograph are due to the modulo-16 resynchronisation, which occurs when a state transition across the hard limiting boundaries imposed by the registers is attempted.

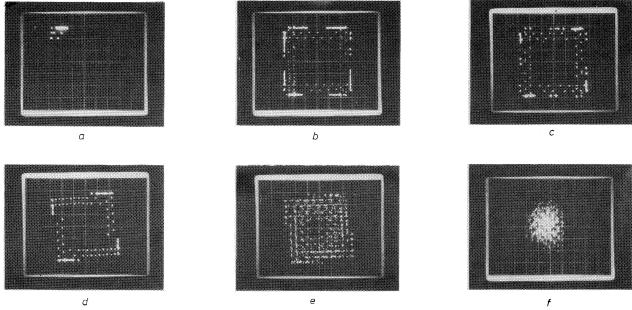


Figure 4: Response of a filter to a noiseless sine wave at various frequencies. (a) Input signal at centre frequency. (b) CF+47Hz ( $\frac{1}{2}$  BIN). (c) +97Hz (1 BIN). (d) +188Hz (2 BINS). (e) +470Hz (5 BINS). (f) +2kHz.

Figure 4 shows the response of a filter to various signals with no noise. In Figure 4a, the signal frequency is at the centre frequency of the filter; there is little difference between this and the 50 dBHz example in Figure 3a. As the frequency is shifted, the small cluster of occupied states begins to move around the periphery of the Argand plane as expected. At first it makes rapid transitions from corner to corner, and then begins to occupy more states on the edges. At a frequency difference of about half the width of the response of the filter, the occupied states begin to collapse towards the origin, as required. An interesting case occurs when the input frequency is far away from the filter centre frequency, illustrated by Figure 4f. Here, the amplitude of the filter must still be incremented or decremented by one during each clock cycle due to the 1-bit quantisation. However

these increments and decrements are not random, as is the case when the input is uncorrelated noise. The result is a state distribution in the Argand plane which is smaller than the noise response depicted in Figure 3e. This is because a greater proportion of the energy of the input signal lies outside the pass band of the filter than in the case where the input is noise alone.

The filter bank is connected to a TMS320-10 microprocessor, so that various processing algorithms, such as averaging, may be tested in conjunction with this system. The nature of these algorithms and the extra hardware are outside the scope of this paper, which seeks only to describe the filter system. We therefore confine ourselves to a few simple, empirical results which will illustrate the behaviour of the circuit and its suitability for our stated application. The power spectra shown in Figure 5 and Figure 6 are generated by the microprocessor and are output to an oscilloscope using digital-to-analogue converters.

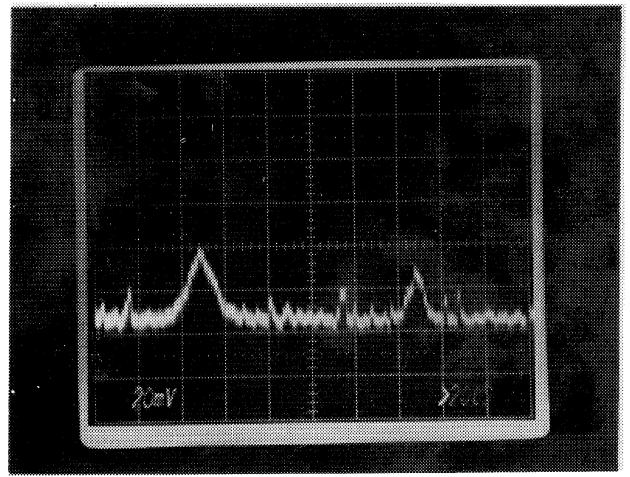


Figure 5: Display of averaged filter output energies showing aliasing.

Figure 5 shows an example of the aliasing behaviour. The horizontal axis on the display represents frequency, and the vertical axis represents the averaged filter response  $E_D$  which is calculated over a large number  $D$  of sparse samples by the microprocessor. The input is a pure sine wave at 3 kHz. The strong response at 3 kHz is accompanied by the large peak at 9 kHz with  $\frac{4}{9}$  of the power. This second large peak is predicted by theory, and is an example of the worst case where there are coincident solutions of Equation 3.12 (i.e.  $i_1 = 1, i_2 = 3, j = -1$  and  $i_1 = 3, i_2 = 1, j = 0$ ). The smaller peaks are also predicted by other solutions; for example, two of the next largest solutions are when  $i_1 = 1, i_2 = 3, j = 0$  and when  $i_1 = 1, i_2 = 3, j = 1$ . These give rise to aliases which are about 0.08 and 0.58 of the way up the band, respectively. Where there are minor coincident solutions, the relative phase of the contributions from each source is not predictable, but this does not concern us here, as it will not

materially affect the performance of the circuit.

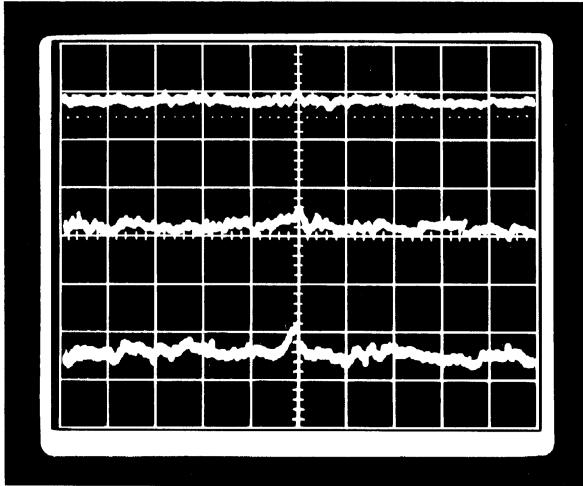


Figure 6: Display of filter output energies with various amounts of averaging. (a) Averaged response after 7 sparse samples. (b) Averaged response after 27 sparse samples. (c) Averaged response after 46 sparse samples.

Figure 6 shows the power spectrum produced by a sine wave added to Gaussian noise where the signal-to-noise ratio is 30 dBHz. This is the lowest signal-to-noise ratio which will normally be present in our application. The averaged response is shown after 7, 27 and 46 sparse samples in Figure 6a, b and c, respectively. The response to a signal which has 50 bit per second phase shift keying (the modulation scheme which we use) is similar in appearance but slightly degraded due to spreading of the peak.

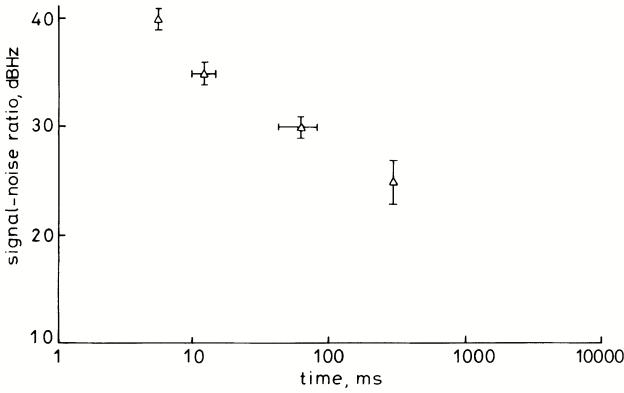


Figure 7: Graph of time to detect a noisy sine wave with 95% confidence against input signal-to-noise ratio (using a crude thresholding algorithm).

Simple tests can be made to determine the time required to find which filter has its centre frequency nearest to the signal frequency. We do not implement the full false alarm and missed detection apparatus of Figure 1.

Instead we choose a crude ‘first past the post’ thresholding technique. In this scheme, we sum the sparse sample values  $E$  and detect which filter sum first passes some preset threshold criterion. We choose this criterion so that (for a given signal-to-noise ratio) the probability of finding the correct filter approximately exceeds 95%. We show these results in Figure 7, where we see that at 30 dBHz we correctly localise the signal in 60 ms. This is two orders of magnitude better than the best state of the art method [2]. However, the results which we obtain are not as good as the theoretical predictions which are plotted in Figure 1 (these predictions are relatively insensitive to missed detection and false alarm levels). This is because of the extremely crude thresholding scheme which we use.

## VI. CONCLUSIONS

The hardware which we have developed primarily to reduce the frequency ambiguity of low data rate communications signals in satellite channels shows considerable improvement over state of the art techniques. Using the averaging and crude thresholding technique which we have described, followed by a short serial search using a swept filter (not described in this paper), we have achieved phase lock to a 30 dBHz signal in a 12 kHz band of ambiguity more than an order of magnitude faster than the seven seconds required before. Using a full false alarm and missed detection analysis, and an improved phase lock loop rather than a final sweep, we expect a further substantial improvement.

There are implications for military and business users of low data rate satellite links. The probability of hostile exploitation of the link will be reduced, especially for short communications where signal acquisition accounts for a large proportion of the time ‘on the air’. Simplex or ‘over, over’ communications become feasible with an attendant saving in transmission time and battery power. In cases where signal fading is encountered, the carrier may be rapidly reacquired.

## VII. ACKNOWLEDGEMENT

We thank A.C. Baynham for supplying the data that are shown in Figure 5, Figure 6 and Figure 7.

In this Appendix we present a detailed derivation of the response PDF of an average of squared values of sparse samples.

The basic PDF is the Rician  $P(E)$

$$P(E) = \frac{1}{a'^2} \exp \left\{ -\frac{(E + b'^2)}{a'^2} \right\} I_0 \left\{ \frac{2E^{\frac{1}{2}}b'}{a'^2} \right\} \quad (1)$$

The characteristic function  $C(k)$  is defined as

$$C(k) \equiv \int_0^\infty dE e^{-ikE} P(E) \quad (2)$$

then

$$C(k) = \frac{\exp(-\frac{b'^2}{a'^2})}{1 + ika'^2} \exp\left[\frac{1}{1 + ika'^2} \frac{b'^2}{a'^2}\right] \quad (3)$$

The characteristic function of the PDF of an average  $E_D$  of  $D$  independently distributed random variables  $E$  is

then  $C(\frac{k}{D})^D$ , and so

$$P_D(E_D) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} dk \exp(ikE_D) C\left(\frac{k}{D}\right)^D \quad (4)$$

Substituting  $\xi = 1 + \frac{ika'^2}{D}$  yields

$$P_D(E_D) = \frac{D}{(a')^{2D}} \exp\left\{-\frac{D(E_D + b'^2)}{a'^2}\right\} \frac{1}{2\pi i} \int_{\xi_-}^{\xi_+} d\xi \exp(sDE_D) \frac{1}{\xi^D} \exp\left[\frac{Db'^2}{\xi a'^4}\right] \quad (5)$$

where  $\xi_{\pm} \equiv \frac{1}{a'^2} \pm i\infty$ . The Bromwich integral in Equation 5 may be evaluated by noting that (eqn. 29.3.80 of [1])

$$L\left\{\left(\frac{t}{a}\right)^{\frac{\nu}{2}} J_{\nu}(2(at)^{\frac{1}{2}})\right\} = \frac{\exp(-\frac{a}{\xi})}{\xi^{\nu+1}} \quad \nu > -1 \quad (6)$$

where  $L\{f(t)\}$  is the Laplace transform of  $f(t)$ . Now using (eqn. 9.6.3 of [1])

$$J_{\nu}(it) = i^{\nu} I_{\nu}(t) \quad (7)$$

which leads to

$$P_D(E_D) = \frac{D}{a'^2} \left(\frac{E_D}{b'^2}\right)^{\frac{D-1}{2}} \exp\left\{-\frac{D(E + b'^2)}{a'^2}\right\} I_{D-1}\left\{\frac{2DE_D^{\frac{1}{2}}b'}{a'^2}\right\} \quad (8)$$

The moments  $M_r$  of  $P_D(E_D)$  may be derived by making use of the characteristic function  $C(\frac{k}{D})^D$ . Thus

$$\begin{aligned} M_r &\equiv \int_0^{\infty} dE_D (E_D)^r P_D(E_D) \\ &= \frac{i^r}{2\pi} \int_0^{\infty} dE_D \int_{-\infty}^{+\infty} dk \exp(ikE_D) \frac{\partial^r [C(\frac{k}{D})^D]}{\partial k^r} \end{aligned} \quad (9)$$

We wish to calculate  $M_1$  and  $M_2$ , and so we need the following results

$$\begin{aligned} \frac{\partial [C(\frac{k}{D})^D]}{\partial k} &= -i \left[ \frac{a'^2}{1 + \frac{ika'^2}{D}} + \frac{b'^2}{(1 + \frac{ika'^2}{D})^2} \right] \left[ C\left(\frac{k}{D}\right)^D \right] \\ \frac{\partial^2 [C(\frac{k}{D})^D]}{\partial k^2} &= - \left[ \frac{(1 + \frac{1}{D}) a'^4}{(1 + \frac{ika'^2}{D})^2} + \frac{2(1 + \frac{1}{D}) a'^2 b'^2}{(1 + \frac{ika'^2}{D})^3} + \frac{b'^4}{(1 + \frac{ika'^2}{D})^4} \right] \left[ C\left(\frac{k}{D}\right)^D \right] \end{aligned} \quad (10)$$

which are easily obtained by differentiating  $C(\frac{k}{D})^D$ . To perform the integration over  $k$  we need to use the result

$$\begin{aligned} K_r &\equiv \frac{1}{2\pi} \int_{-\infty}^{+\infty} dk \exp(ikE_D) \frac{1}{(1 + \frac{ika'^2}{D})^r} \exp\left[\frac{1}{1 + \frac{ika'^2}{D}} \frac{Db'^2}{a'^2}\right] \\ &= D \exp\left(\frac{Db'^2}{a'^2}\right) P_r(D E_D) \end{aligned} \quad (11)$$

Combining results and integrating over  $E_D$  using the normalisation of  $P_r$  we obtain

$$\begin{aligned} M_1 &= a'^2 + b'^2 \\ M_2 &= \left(1 + \frac{1}{D}\right) a'^4 + 2\left(1 + \frac{1}{D}\right) a'^2 b'^2 + b'^4 \end{aligned} \quad (12)$$

[1] M Abramowitz and I A Stegun, *Handbook of mathematical functions*, Dover Publications, New York, 1972.

[2] A R Duddle, R G Powell, M G Warner, K M Gannon,

- and M Aeigus, *Rapid synchronisation of CPSK modems used for tactical satellite communications*, Technical Report BL 4105, Marconi Space and Defence Systems, Stanmore, 1982.
- [3] C H Jones, *A manpack satellite communications earth station*, The radio and electronic engineer **51** (1981), no. 6, 259–271.
- [4] S P Luttrell and J A Pritchard, *Signal transforming device*, Patent GB 2190221 B, 1987.
- [5] P J Skilton and I L Westall, *Military microwaves*, ch. ‘Manpack’ SHF satellite ground terminal, pp. 87–93, Microwave Exhibitions and Publishers, London, 1982.

# Derivation of a Class of Training Algorithms \*

S. P. LUTTRELL

*Royal Signals and Radar Establishment, Malvern, Worcs., WR14 3PS, U.K.*

This paper presents a novel derivation of Kohonen's topographic mapping training algorithm, based upon an extension of the Linde-Buzo-Gray (LBG) algorithm for vector quantiser design. Thus a vector quantiser is designed by minimising an  $L_2$  reconstruction distortion measure, including an additional contribution from the effect of code noise which corrupts the output of the vector quantiser. The neighbourhood updating scheme of Kohonen's topographic mapping training algorithm emerges as a special case of this code noise model. This formulation of Kohonen's algorithm is a specific instance of the "robust hidden layer principle", which stabilises the internal representations chosen by a network against anticipated noise or distortion processes.

## I. INTRODUCTION

Vector quantisation theory is a generalisation of scalar quantisation theory as expressed by the Lloyd-Max equations [4, 9]. The generalisation of the Lloyd-Max equations to vector quantisation is straightforward, and the Linde-Buzo-Gray (LBG), or  $k$  means, algorithm [3] provides a means of adjusting the code vectors to locate a local minimum of a distortion measure.

In this paper, the  $L_2$  distortion measure is extended to include the effect of corruption of the vector quantiser output code. Kohonen's topographic mapping training algorithm then emerges naturally. Some extensions of the technique are indicated.

## II. THE LBG ALGORITHM

Although it is not strictly necessary, we shall restrict our attention to an  $L_2$  distortion measure  $d(\mathbf{x}, \mathbf{x}')$  defined as the following, which measures the Euclidean distance between a vector  $\mathbf{x}$  and its reconstruction after vector quantisation  $\mathbf{x}'$ :

$$d(\mathbf{x}, \mathbf{x}') \equiv \|\mathbf{x}' - \mathbf{x}\|^2 \quad (2.1)$$

We shall be concerned with the average of  $d(\mathbf{x}, \mathbf{x}')$  over a training set of vectors  $\mathbf{x}$ , so we shall introduce the probability density function (PDF)  $P(\mathbf{x})$  over samples  $\mathbf{x}$  selected at random from the training set.

Denote the encoding operation as  $y(\mathbf{x})$ , and the corresponding decoding operation as  $\mathbf{x}'(y)$ . The *average*  $L_2$  distortion  $D$  is the average of  $d(\mathbf{x}, \mathbf{x}'(y(\mathbf{x})))$  over  $\mathbf{x}$  sampled from  $P(\mathbf{x})$

$$D \equiv \int d\mathbf{x} P(\mathbf{x}) \|\mathbf{x}'(y(\mathbf{x})) - \mathbf{x}\|^2 \quad (2.2)$$

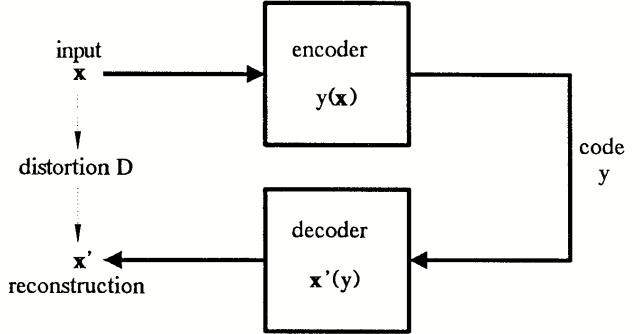


Figure 1: Representation of Equation 2.2 describing the  $L_2$  distortion  $D$  after encoding using  $y(\mathbf{x})$  and then decoding using  $\mathbf{x}'(y)$ . This is a single stage encoder.

Equation 2.2 is depicted schematically in Figure 1, where  $\mathbf{x}'(y)$  acts as a pseudoinverse for  $y(\mathbf{x})$ . The optimum encoding/decoding scheme is found by appropriately varying the functions  $y(\mathbf{x})$  and  $\mathbf{x}'(y)$  so as to minimise  $D$ . Necessary conditions for minimisation of  $D$  are [3]:

A1) Given  $\mathbf{x}$ , the appropriate code  $y = y(\mathbf{x})$  minimises the distortion  $\|\mathbf{x}'(y(\mathbf{x})) - \mathbf{x}\|^2$ .

A2) Given  $y$ , the appropriate reconstruction  $\mathbf{x}' = \mathbf{x}'(y)$  is the centroid of those  $\mathbf{x}$ 's which satisfy  $y = y(\mathbf{x})$ .

Note that A1 is a nearest neighbour encoding rule. A1 and A2 imply that  $D$  is stationary (and locally minimum) with respect to variations of  $y(\mathbf{x})$  and  $\mathbf{x}'(y)$ , respectively. Such conditions are therefore sufficient for a local minimum of  $D$ , but are only necessary for a global minimum. The LBG algorithm runs in batch training mode where the whole training set is presented before performing an update, which consists of alternately adjusting  $y(\mathbf{x})$  according to A1, and then adjusting  $\mathbf{x}'(y)$  according to A2.

## III. TWO STAGE QUANTISATION

Now consider a scheme in which the encoding operation is broken down into two stages,  $\mathbf{h}(\mathbf{x})$  followed by  $y(\mathbf{h})$ , where  $\mathbf{h}$  is an intermediate code. The  $L_2$  distortion

\*Typeset in LATEX on May 21, 2019.

This paper appeared in IEEE Trans. Neural Networks, 1990, vol. 1, no. 2, pp. 229-232. Manuscript received June 2, 1989; revised November 27, 1989. An earlier version of this paper was presented at the 1989 International Conference on Neural Networks, Washington, DC, June 19-22, 1989. © 1990 British Crown Copyright.

is then given by

$$D \equiv \int d\mathbf{x} P(\mathbf{x}) \|\mathbf{x}'(y(\mathbf{h}(\mathbf{x}))) - \mathbf{x}\|^2 \quad (3.1)$$

Equation 3.1 is more clumsy than Equation 2.2 because there are many ways of achieving the same overall encoding operation 7 by delicately balancing the functional forms of  $\mathbf{h}(\mathbf{x})$  and  $y(\mathbf{h})$ . In general, the functional forms of  $\mathbf{h}(\mathbf{x})$  and  $y(\mathbf{h})$  are not simple, although  $y(\mathbf{h}(\mathbf{x}))$  itself is a vector quantiser.

The essential step is to model the average distortion due to the second stage of encoding  $y(\mathbf{h})$  as a noise process acting on the output of the first stage  $\mathbf{h}(\mathbf{x})$ . Thus, consider a modified distortion defined as

$$D_1 \equiv \int d\mathbf{x} P(\mathbf{x}) \int d\mathbf{n} \pi(\mathbf{n}) \|\mathbf{x}'(\mathbf{h}(\mathbf{x}) + \mathbf{n}) - \mathbf{x}\|^2 \quad (3.2)$$

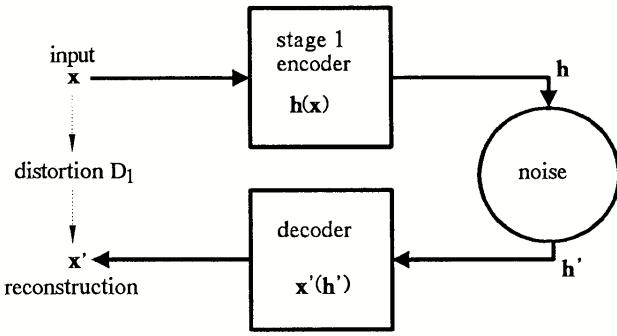


Figure 2: Representation of Equation 3.2 describing the  $L_2$  distortion  $D_1$ , after encoding using  $\mathbf{h}(\mathbf{x})$  and modelling the distorting effect of  $y(\mathbf{h})$  as a noise process that corrupts  $\mathbf{h}$  into  $\mathbf{h}'$ , and then decoding using  $\mathbf{x}'(\mathbf{h}')$ .

Equation 3.2 is depicted in Figure 2. This is the  $L_2$  distortion which would arise from encoding  $\mathbf{x}$  using  $\mathbf{h}(\mathbf{x})$ , followed by addition of noise  $\mathbf{n}$  with PDF  $\pi(\mathbf{n})$  (to produce  $\mathbf{h}'$ ), followed by decoding using  $\mathbf{x}'(\mathbf{h}')$ . By a suitable choice of  $\pi(\mathbf{n})$  we can model the average distortion due to  $y(\mathbf{h})$ . Note that we use a somewhat cavalier notation by using  $\mathbf{x}'(\mathbf{h}')$  here and  $\mathbf{x}'(y)$  elsewhere: this does not imply that the functions are the same.

The use of a modified distortion measure ( $D_1$ , rather than  $D$ ) is a specific instance of the application of the “robust hidden layer principle”. This principle states that a network should be trained in the presence of all anticipated noise and distortion processes acting on its internal representations. The effect of this is to encourage the network to encode information in its internal representations in such a way that it is robust with respect to the damaging effects of noise and distortion. In this case we use  $\pi(\mathbf{n})$  to model stochastically the effects of the distortion process.

The noise model that we have used in Equation 3.2 is additive. We could use more sophisticated noise models to take into account the different distortions introduced

by  $y(\mathbf{h})$  for different  $\mathbf{h}$ , in which case  $\pi(\mathbf{n})$  would become  $\pi(\mathbf{n}|\mathbf{h})$ . We do not study such an extension of the model in this paper.

In Figure 3 we show the set theoretic version of Figure 2. Note how the distortion is intuitively and consistently represented by the sizes of the subsets depicted. Neither  $\mathbf{h}(\mathbf{x})$  nor  $y(\mathbf{h})$  is guaranteed to have a simple form (only  $y(\mathbf{h}(\mathbf{x}))$  is a complete vector quantiser), but we have nevertheless represented them as if they mapped clusters of nearby points to the same output: this is purely for diagrammatic convenience. The equivalent noise which  $y(\mathbf{h})$  induces on a particular  $\mathbf{h}$  ( $\mathbf{h}_0$  say) is then represented by the set of  $\mathbf{h}$  which satisfy  $y(\mathbf{h}) = y(\mathbf{h}_0)$ . In Figure 3 we show how four separate  $\mathbf{h}$  values are mapped to the same  $y$  value, and we show how each of those four  $\mathbf{h}$  values derives from a small cluster of  $\mathbf{x}$  values, and how the four clusters themselves are clustered, thus guaranteeing a small overall  $L_2$  distortion.

Note that we have not yet said anything about the detailed form of the noise model, not even whether it is a continuous function. It is determined entirely by the particular form of the distortion introduced by  $y(\mathbf{h})$ .

#### IV. A MODIFICATION OF THE LBG ALGORITHM

Now let us minimise  $D_1$  with respect to  $\mathbf{h}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{h})$ . The functional derivatives of  $D_1$  with respect to  $\mathbf{h}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{h})$  are given by

$$\frac{\delta D_1}{\delta \mathbf{h}(\mathbf{x})} = P(\mathbf{x}) \int d\mathbf{n} \pi(\mathbf{n}) \left. \frac{\partial \|\mathbf{x}'(\mathbf{h}) - \mathbf{x}\|^2}{\partial \mathbf{h}} \right|_{\mathbf{h}=\mathbf{h}(\mathbf{x})+\mathbf{n}} \quad (4.1)$$

$$\frac{\delta D_1}{\delta \mathbf{x}'(\mathbf{h})} = 2 \int d\mathbf{x} P(\mathbf{x}) \pi(\mathbf{h} - \mathbf{h}(\mathbf{x})) [\mathbf{x}'(\mathbf{h}) - \mathbf{x}] \quad (4.2)$$

These two derivatives imply that the two necessary conditions A1 and A2 for minimising  $D$ , must be modified to become:

- B1) Given  $\mathbf{x}$ , choose  $\mathbf{h}(\mathbf{x})$  to minimise  $\int d\mathbf{n} \pi(\mathbf{n}) \|\mathbf{x}'(\mathbf{h}(\mathbf{x}) + \mathbf{n}) - \mathbf{x}\|^2$ .
- B2) Given  $\mathbf{h}$ , choose  $\mathbf{x}'(\mathbf{h})$  to satisfy

$$\mathbf{x}'(\mathbf{h}) = \frac{\int d\mathbf{x} P(\mathbf{x}) \pi(\mathbf{h} - \mathbf{h}(\mathbf{x})) \mathbf{x}}{\int d\mathbf{x} P(\mathbf{x}) \pi(\mathbf{h} - \mathbf{h}(\mathbf{x}))} \quad (4.3)$$

As a consistency check, if  $\pi(\mathbf{n}) = \delta(\mathbf{n})$ , where  $\delta(\mathbf{n})$  is a Dirac delta function (or the analogous Kronecker delta if we consider the discrete case), then the model reduces to the encoding/decoding depicted in Figure 1 (with  $y$  replaced by  $\mathbf{h}$ ), and the above conditions reduce to the standard LBG vector quantisation conditions, as expected.

It remains to determine a suitable form for the noise model  $\pi(\mathbf{n})$ , which depends on the distorting effect of the second stage of quantisation  $y(\mathbf{h})$ . We need to be more

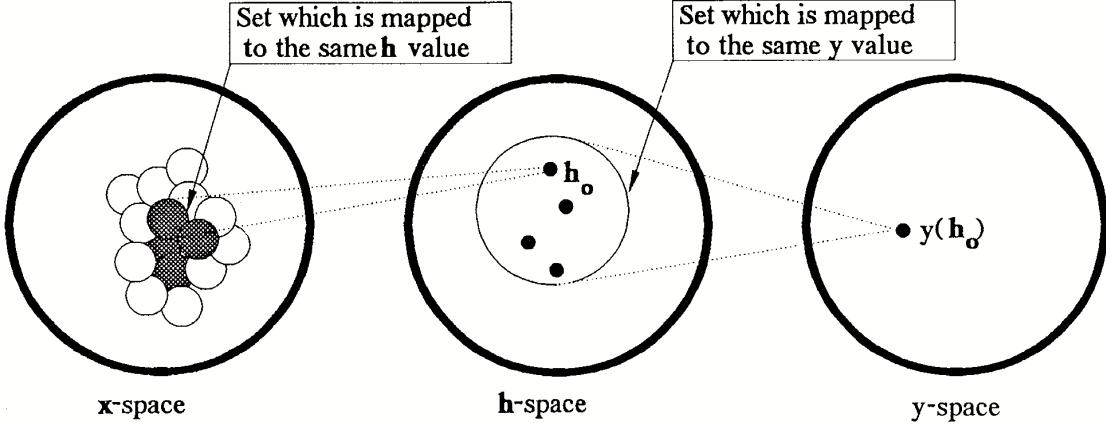


Figure 3: Set theoretic representation of Figure 2.

specific about  $y(\mathbf{h})$ , so we shall now assume that  $y(\mathbf{h})$  is a minimum  $L_2$  distortion vector quantiser for  $\mathbf{h}$ . Thus we shall minimise  $D_2$  with respect to  $y(\mathbf{h})$  and  $\mathbf{h}'(y)$ , where  $D_2$  is defined as

$$D_2 \equiv \int d\mathbf{h} P_h(\mathbf{h}) \|\mathbf{h}'(y(\mathbf{h})) - \mathbf{h}\|^2 \quad (4.4)$$

$P_h(\mathbf{h})$  is the PDF of the output  $\mathbf{h}$  of the first stage of quantisation. Given the value of  $D_2$ , and no further in-

formation about the distorting effect of  $y(\mathbf{h})$ , a zero mean Gaussian noise model with variance  $D_2$  is then the appropriate maximum entropy choice for  $\pi(\mathbf{n})$ .

We may simplify B1 by Taylor expanding  $\mathbf{x}'(\mathbf{h}(\mathbf{x}) + \mathbf{n})$

$$\mathbf{x}'(\mathbf{h}(\mathbf{x}) + \mathbf{n}) = \exp[\mathbf{n} \cdot \nabla_{\mathbf{h}}] \mathbf{x}'(\mathbf{h})|_{\mathbf{h}=\mathbf{h}(\mathbf{x})} \quad (4.5)$$

where  $\nabla_{\mathbf{h}}$  is the vector differentiation operator with respect to  $\mathbf{h}$ . This yields to second order

---


$$\int d\mathbf{n} \pi(\mathbf{n}) \|\mathbf{x}'(\mathbf{h}(\mathbf{x}) + \mathbf{n}) - \mathbf{x}\|^2 \simeq \left[ 1 + \frac{1}{2} D_2 \nabla_{\mathbf{h}}^2 \right] [\mathbf{x}'(\mathbf{h}) - \mathbf{x}] \Big|_{\mathbf{h}=\mathbf{h}(\mathbf{x})} \quad (4.6)$$


---

where the statistics of the maximum entropy zero mean Gaussian noise model for  $\pi(\mathbf{n})$  are

$$\begin{aligned} \int d\mathbf{n} \pi(\mathbf{n}) &= 1 \\ \int d\mathbf{n} \pi(\mathbf{n}) n_i &= 0 \\ \int d\mathbf{n} \pi(\mathbf{n}) n_i n_j &= D_2 \delta_{i,j} \end{aligned} \quad (4.7)$$

The first term on the right-hand side of Equation 4.6 is the conventional distortion, and the second (curvature) term arises from the output noise model  $\pi(\mathbf{n})$ . We shall henceforth assume that the curvature term is small (i.e.,  $\mathcal{O}(D_2 \nabla_{\mathbf{h}}^2) \ll 1$ , effectively), so that B1 may be approximated by A1 (with  $y$  replaced by  $\mathbf{h}(\mathbf{x})$ ). This reduces B1 to a nearest neighbour encoding rule, as before.

B2 cannot be simplified in an analogous fashion to B1. However, from Equation 4.2 we may derive a stochastic gradient descent algorithm for realising B2. Thus we draw  $\mathbf{x}$  samples at random from the training set using

the  $\int d\mathbf{x} P(\mathbf{x})$  factor, and gradient descent of  $D_1$  is then achieved by updates of the following form where  $\epsilon > 0$ ,  $\mathbf{x}$  is drawn from the probability density function  $P(\mathbf{x})$ , and  $\mathbf{h}(\mathbf{x})$  is the nearest neighbour encoding approximation to B1:

$$\mathbf{x}'(\mathbf{h}') \longrightarrow \mathbf{x}'(\mathbf{h}') + \epsilon \pi(\mathbf{h}' - \mathbf{h}(\mathbf{x})) [\mathbf{x} - \mathbf{x}'(\mathbf{h}')] \quad (4.8)$$

This update prescription is applied to all  $\mathbf{h}'$  for which  $\pi(\mathbf{h}' - \mathbf{h}(\mathbf{x})) > 0$ . This update scheme is identical to the one that Kohonen has proposed for obtaining topographic mappings [2], and it leads to a decoding operation  $\mathbf{x}'(\mathbf{h})$  that is a smoothly varying function of  $\mathbf{h}$ , although the encoding operation  $\mathbf{h}(\mathbf{x})$  is not guaranteed to be a smooth function of  $\mathbf{x}$  when  $\dim(\mathbf{h}) < \dim(\mathbf{x})$ . Note that we have had to make an approximation (where we retain only the leading term in Equation 4.6) in order to obtain the correspondence between a minimum  $L_2$  distortion vector quantiser (with added code noise) and Kohonen's topographic mappings.

Our noise model does not make specific predictions about the dependence of  $\pi(\mathbf{n})$  on the number of updates

that have elapsed. However,  $D_1$  has multiple local minima which makes it difficult to locate the global minimum by using the gradient descent algorithm specified in Equation 4.8. The randomness of sampling from the training set makes unpredictable the order in which updates are performed according to Equation 4.8, which alleviates the local minimum problem somewhat. However, a broader  $\pi(\mathbf{n})$  gives rise to a  $D_1$  which has a softer dependence on  $\mathbf{h}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{h})$ , so better convergence to the global minimum of  $D_1$  is obtained by starting with a broad  $\pi(\mathbf{n})$ , and then progressively reducing the width of  $\pi(\mathbf{n})$  as the updates proceed until the required form is obtained [2]. This is equivalent to a renormalised scheme in which  $\pi(\mathbf{n})$  is fixed while the underlying space is progressively dilated as the updates proceed [5]. The renormalised scheme gives much faster convergence than Kohonen's original scheme.

A similar approach to that presented in this paper has been presented elsewhere [10]. They minimise a quadratic potential which is analogous to our distortion measure  $D_1$  in Equation 3.2. However, our approach provides a more detailed interpretation of the neighbourhood updating aspect of the training algorithm.

## V. EXTENSION TO A MULTILAYER VECTOR QUANTISER

We shall now use the robust hidden layer principle to state the form of Equation 3.2 that applies to an adjacent pair of layers in a multilayer vector quantiser. Thus define a distortion  $D(m)$  for encoding/decoding layer  $m$  of a multilayer vector quantiser

$$D(m) \equiv \int d\mathbf{x}_m P_m(\mathbf{x}_m) \int d\mathbf{n} \pi_m(\mathbf{n}) \|\mathbf{x}'_m(\mathbf{x}_{m+1}(\mathbf{x}_m) + \mathbf{n}) - \mathbf{x}_m\|^2 \quad (5.1)$$

where  $\mathbf{x}_m$  is a vector representing the state of layer  $m$  of the quantiser,  $\mathbf{x}_{m+1}(\mathbf{x}_m)$  is the encoding operation used to obtain the state of layer  $m + 1$  from layer  $m$ ,  $\mathbf{x}'_m(\mathbf{x}_{m+1})$  is the decoding operation used to obtain the state of layer  $m$  from layer  $m + 1$ , and  $\pi_m(\mathbf{n})$  is a noise

process that models the distortion due to all subsequent stages of the multilayer vector quantiser. The input probability density function  $P_m(\mathbf{x}_m)$  is obtained as shown where we now write  $\mathbf{x}_1$  instead of  $\mathbf{x}$  for the original input vector:

$$P_m(\mathbf{x}_m) = \int d\mathbf{x}_1 d\mathbf{x}_2 \cdots d\mathbf{x}_{m-1} P_1(\mathbf{x}_1) \delta(\mathbf{x}_2 - \mathbf{x}_2(\mathbf{x}_1)) \cdots \delta(\mathbf{x}_m - \mathbf{x}_m(\mathbf{x}_{m-1})) \quad (5.2)$$

In practice, samples  $\mathbf{x}_m$  from  $P_m(\mathbf{x}_m)$  may easily be generated by propagating samples  $\mathbf{x}_1$  from  $P_1(\mathbf{x}_1)$  through the layers of the network until layer  $m$  is reached. Thus the output of layer  $m - 1$  is used as the training set for determining the optimum encoding/decoding operations on layer  $m$ .

The advantage of modeling the distorting effect of all subsequent layers of the multilayer vector quantiser as a noise process is that the encoding/decoding operations associated with layer  $m$  can be optimised (almost) independently of the other layers' encoding/decoding opera-

tions. Coupling between these optimisations is achieved via the distortion models  $\pi_m(\mathbf{n})$ . Although this will not lead to an optimal overall encoding/decoding scheme, it is computationally much cheaper than simultaneously optimising all of the coupled layers. Finally, when the multilayer vector quantiser has been trained, and we wish to decode  $\mathbf{x}_m$  to obtain a reconstruction  $\mathbf{x}_{1,\text{rec}}(\mathbf{x}_m)$  of the input vector  $\mathbf{x}_1$ , we should use the following formula which is the centroid of the set of input vectors that map to  $\mathbf{x}_m$ :

$$\mathbf{x}_{1,\text{rec}}(\mathbf{x}_m) = \frac{\int d\mathbf{x}_1 d\mathbf{x}_2 \cdots d\mathbf{x}_{m-1} P_1(\mathbf{x}_1) \delta(\mathbf{x}_2 - \mathbf{x}_2(\mathbf{x}_1)) \cdots \delta(\mathbf{x}_m - \mathbf{x}_m(\mathbf{x}_{m-1})) \mathbf{x}_1}{P_m(\mathbf{x}_m)} \quad (5.3)$$

## VI. FURTHER EXTENSIONS

All of the results in this paper can be reexpressed using an arbitrary (non- $L_2$ ) distortion measure. The choice of a

preferred form for  $d(\mathbf{x}, \mathbf{x}')$  depends on whatever structure

in  $\mathbf{x}$  one deems to be important.

The LBG algorithm is the batch training version of Kohonen's vector quantisation algorithm with zero neighbourhood size. It is also possible to use Equation 4.1 and Equation 4.2 to derive a batch training version of Kohonen's topographic mapping algorithm.

We have presented elsewhere some applications of 2-stage and multistage vector quantisation to low-level image processing [5], cluster decomposition of PDF's [8], image compression [7], and time series analysis [6].

A nice example of the use of the robust hidden layer principle has been reported in [1]. The problem studied there was the design of a codebook containing 8 bit codewords that needed to be robust with respect to bit errors.  $\pi(\mathbf{n})$  must therefore describe Hamming noise, which induces an 8-dimensional Hamming cube topology on the codebook. A worthwhile reduction of 3 dB in the reconstruction distortion was obtained by topographically organising the codebook in this fashion.

## VII. SUMMARY AND CONCLUSIONS

We have shown how topographic mappings arise naturally in a 2-stage vector quantiser whose output is made

robust with respect to various types of distortion. The choice of neighbourhood used at the output of the first stage is determined solely by the additional distortion introduced by the second stage of the quantiser. This principle generalises directly to multistage quantisers. The advantage of our approach is that (variants of) Kohonen's self-organising neural network may be derived from first principles.

We have encapsulated this robustness argument in the robust hidden layer principle: train your network in the presence of all anticipated distortions to its internal representations. The network will then do its best to choose internal representations that encode information in such a way that it is robust with respect to being damaged by the distortions.

A hierarchical multistage quantiser constructed along these lines is an adaptive pyramid processor in which successively cruder quantisations (or codings) of the input are extracted. Current research indicates that such neural networks show much promise [6].

- 
- [1] D S Bradburn, *Reducing transmission error effects using a self-organising network*, Proceedings of IJCNN international joint conference on neural networks (Washington DC), IEEE, 1989, pp. 531–537.
  - [2] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
  - [3] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
  - [4] S P Lloyd, *Least squares quantisation in PCM*, IEEE Transactions on Information Theory **28** (1982), no. 2, 129–137.
  - [5] S P Luttrell, *Self organising multilayer topographic mappings*, Proceedings of IEEE conference on neural networks (San Diego), IEEE, 1988, pp. 93–100.
  - [6] ———, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
  - [7] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
  - [8] ———, *Maximum entropy and Bayesian methods*, ch. The use of Bayesian and entropic methods in neural network theory, pp. 363–370, Kluwer, Dordrecht, 1989.
  - [9] J Max, *Quantising for minimum distortion*, IEEE Transactions on Information Theory **6** (1960), no. 1, 7–12.
  - [10] H Ritter and K Schulten, *Kohonen's self-organising maps: Exploring their computational capabilities*, Proceedings of IEEE conference on neural networks (San Diego), IEEE, 1988, pp. 109–116.



# A Bayesian Derivation of an Iterative Autofocus/Super-Resolution Algorithm \*

Stephen P Luttrell  
RSRE, Malvern, WR14 3PS, UK

We derive an estimate-maximise formulation of a Bayesian super-resolution algorithm for reconstructing scattering cross sections from coherent images. We generalise this result to obtain an ‘autofocus/super-resolution’ method, which simultaneously autofocuses an imaging system and super-resolves its image data. We present an explanatory numerical example to illustrate the implementation of our method on images of single and double point targets that are defocused by  $\mathcal{O}$ (depth of focus). These are successfully super-resolved by autofocus/super-resolution, but not by pure super-resolution. We conjecture that autofocus/super-resolution might usefully be applied to the interpretation of airborne synthetic aperture radar images that are subject to defocusing effects.

## I. INTRODUCTION

Super-resolution is the name that we give to the process of increasing the effective bandwidth of an image (or time series) by introducing collateral information to augment the dataset: the classical Rayleigh resolution limit for distinguishing two point targets can thereby be overcome. Super-resolution belongs to the general class of ‘inverse problems’ because it attempts to recover an object from its image by deconvolving the imaging operator.

A limitation of our existing super-resolution technique [3] is its assumption that the parameters of the imaging system are known precisely, which causes problems for some applications. Although this problem of imaging system calibration is quite general, this work was originally prompted by the need to super-resolve images obtained from synthetic aperture radar (SAR) systems which have time varying imaging system parameters (due to phase shifts caused by anomalous motion of the transmitter/receiver). An ideal SAR can be modelled as if it were a simple linear imaging system, and the anomalous motion can (in first order) be modelled as a simple defocusing of this linear imaging system. It is therefore convenient to visualise a SAR as being a microwave version of an optical bench experiment using coherent illumination and with the lens misplaced from its correct focus. We shall consider  $\mathcal{O}$ (depth of focus) errors in the placement of the lens, which can cause severe degradation in super-resolved image quality [10]. This  $\mathcal{O}$ (depth of focus) criterion applies independently of what the actual physical dimensions of a depth of focus happen to be for a particular imaging system, because it can be expressed alternatively as an upper bound on the quadratic phase error (which is dimensionless) that is acceptable at the edge of the aperture.

We need to use a precise autofocus method in order to super-resolve successfully. We therefore develop a hybrid

‘autofocus/super-resolution’ technique, in which autofocusing uses the super-resolved image (rather than the original image) to adjust the focusing parameter(s). The philosophy of this technique is to find the set of imaging system parameters and super-resolved image that simultaneously fit the image data and collateral information. In this paper we develop a theoretical framework that is applicable to any coherent imaging system, so it is not important which specific application (namely SAR image analysis) originally prompted this work, and our results are therefore of interest to a wide audience.

Throughout this paper we use Bayesian calculus, because it is the only fully consistent means of performing inferences from limited information [2, 8]. Bayesian calculus uses probabilities to encode information, and makes inferences by manipulating these probabilities. For clarity, we use physical arguments to justify the form of each probability that we introduce. One could also note the equivalence between *inverse problems* and *inference problems* as a justification for making Bayesian calculus the appropriate language for formulating and solving inverse problems.

In Section II we summarise our Gaussian scattering model and our linear imaging model. We construct our models in terms of probabilities to facilitate the use of Bayes’ theorem to solve the inverse problem of determining what caused a particular dataset.

In Section III we present a complete and rigorous derivation of an iterative Bayesian super-resolution algorithm. This is an improved version of an algorithm that we described in [9, 14][13], and should be regarded our current definitive treatment of super-resolution. Our method is an application of the estimate-maximise (EM) method of solving maximum-likelihood problems [1, 4].

In Section IV we extend our iterative algorithm to account for uncertainties in the imaging system. The technique that we derive is a precise autofocus method, which effectively focuses on structure in the super-resolved image (rather than the original image). We also present a linearised version of the algorithm for use in simple cases.

In Section V we present an explanatory numerical example to demonstrate how our autofocus/super-resolution method might be applied in practice. Note that we do not attempt to construct a robust general-

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 11, 2019.

This paper appeared in Inverse Problems, 1990, vol. 6, pp. 975-996. Received 5 February 1990, in final form 8 June 1990. © Controller, Her Majesty’s Stationery Office, 1990.

purpose algorithm in this paper. Thus we use synthetic data that is generated by a defocused ‘sinc’ function, in which case the linearised version of autofocus/super-resolution is sufficient, provided that the lens is within  $\mathcal{O}(\text{depth of focus})$  of the correct focus. We demonstrate autofocus/super-resolution for both the single and double point target cases.

In Appendix A we gather together various definitions and derivations that would otherwise distract the flow of the argument in the main body of the paper.

## II. THE MODEL

In this section we summarise our coherent scattering and imaging model. We attempt to formulate our Bayesian model by appealing to physical reasoning, wherever possible.

In Figure 1 we represent as a network the various stages of image formation.

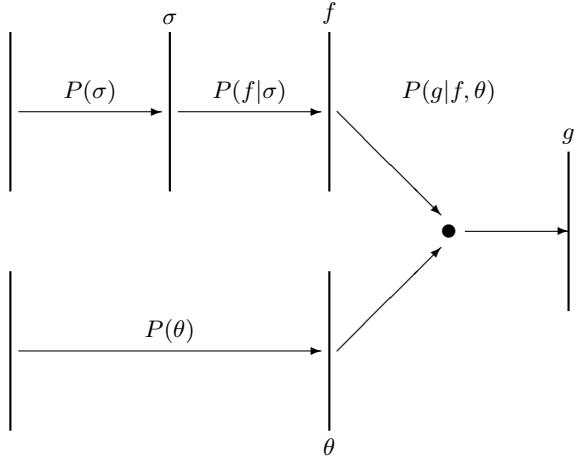


Figure 1: Network decomposition of imaging.

The notation in Figure 1 is defined as

$$\begin{aligned}
 \sigma &\equiv \text{scattering cross section} \\
 f &\equiv \text{scattered field} \\
 g &\equiv \text{image data} \\
 \theta &\equiv \text{imaging parameters} \\
 P(\sigma) &\equiv \text{prior probability over cross sections} \\
 P(f|\sigma) &\equiv \text{scattering model} \\
 P(g|f, \theta) &\equiv \text{imaging model} \\
 P(\theta) &\equiv \text{prior probability over imaging parameters}
 \end{aligned} \tag{2.1}$$

Note that we are somewhat cavalier in our choice of notation, because for instance  $P(\sigma)$  and  $P(\theta)$  are different functions of their respective arguments, yet we use the notation  $P(\cdot)$  for both. Thus the meaning of  $P(\cdot)$  should be deduced from context.

Figure 1 may be used to decompose the joint probability  $P(g, f, \theta, \sigma)$  into a product of factors

$$P(g, f, \theta, \sigma) = P(g|f, \theta) P(f|\sigma) P(\sigma) P(\theta) \tag{2.2}$$

This has the form of a Markov tree, where each argument depends directly on only a limited number of other arguments, and there are no circular dependencies. We frequently use analogous Markovian decompositions of parts of Figure 1 as intermediate steps in our derivations, so it is helpful to keep Figure 1 in mind when reading this paper.

We have deliberately omitted the annotation from the left hand part of Figure 1. This indicates that, in general, we are ignorant of the physical origin of  $P(\sigma)$  and  $P(\theta)$ : they serve only as prior probabilities to encode our state of ignorance about the values that  $\sigma$  and  $\theta$  might have. This is not a failing of the Bayesian approach, rather it is an honest expression of our ignorance about the finer details of the imaging model.

### A. Prior probability over cross sections

The upper left part of Figure 1 generates the prior probability over cross sections  $P(\sigma)$ . The cross section  $\sigma$  is an idealised model of those properties of the illuminated object that affect the scattered field  $f$ . This use of a cross section is phenomenological, because it is incapable of capturing the full range of properties of the process that generates the scattered field.

For completeness, we include the  $P(\sigma)$  term in our Bayesian derivations. However, the main goal of this paper is to demonstrate autofocus/super-resolution, which we manage to simulate in simple cases without introducing  $P(\sigma)$ . In more complicated cases we might need to use explicit prior knowledge, such as  $\Gamma$ -distributed cross section models [7, 16], or Markov random field cross section models [6].

### B. Stochastic scattering model

The upper centre part of Figure 1 generates a scattered field  $f$  according to a scattering model  $P(f|\sigma)$ . We assume that each cross section element  $\sigma_i$  acts as if it produces a large number of scattered wavelets that combine coherently to produce an element of scattered field  $f_i$ , which leads to Gaussian statistics via the central limit theorem. Note that  $f$  is a near field which retains essentially the same spatial structure as the scattering cross section itself, so it could be directly imaged without using a lens (in principle). For  $m$  cross section elements we obtain

$$\begin{aligned}
 P(f|\sigma) &= \prod_{i=1}^m \frac{\exp\left(-\frac{|f_i|^2}{\sigma_i}\right)}{\pi\sigma_i} \\
 &= \frac{1}{\det(\pi\sigma)} \exp\left(-f^\dagger \sigma^{-1} f\right)
 \end{aligned} \tag{2.3}$$

where we define

$$\sigma \equiv \begin{pmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_m \end{pmatrix} \quad (2.4)$$

$$f^\dagger \sigma^{-1} f \equiv \sum_{i,j=1}^m f_i^* (\sigma^{-1})_{ij} f_j \quad (2.5)$$

In the first line of Equation 2.3 we ignore any correlations that might exist between different components of  $f$ , whereas in the second line of Equation 2.3 we use a notation that allows us to include off-diagonal elements in  $\sigma$  to model such correlations, should we wish to do so. For generality, we assume that  $\sigma$  is matrix-valued throughout our derivations, unless we state otherwise.

We use operator/state notation (as in  $f^\dagger \sigma^{-1} f$ ) throughout this paper, because it is both economical and it facilitates calculations that would be tedious if performed using explicit summations over indices. Note also that each component of the state vector  $f$  is a complex number, which explains the unusual normalisation of the Gaussian probability in Equation 2.3.

### C. Stochastic imaging model and its parameters

The right hand part of Figure 1 depends on both the scattered field  $f$  and on the imaging parameters  $\theta$ . Define an imaging model  $P(g|f, \theta)$  as

$$P(g|f, \theta) = \frac{1}{\det(\pi N)} \exp \left[ -(g - T(\theta)f)^\dagger N^{-1} (g - T(\theta)f) \right] \quad (2.6)$$

where  $N$  is a positive semi-definite covariance matrix that is used to model additive Gaussian image data noise. In the limit where all the eigenvalues of  $N$  tend to zero this reduces to  $P(g|f, \theta) = \delta(g - T(\theta)f)$  (where  $\delta(g - T(\theta)f)$  is the Dirac delta function), which describes the noiseless imaging equation  $g = T(\theta)f$ .

We use the parameter vector  $\theta$  to parameterise variability in the imaging system, and we use  $P(\theta)$  to model our prior knowledge of these parameters. In general,  $\theta$  will be a low dimensional vector that describes those components of the imaging system that cannot be calibrated once and for all. We find that we do not need to introduce  $P(\theta)$  in order to demonstrate autofocus/super-resolution in simple cases, but we include it in our derivations, for completeness.

In Section IV B we derive a linearised autofocus scheme which uses a Gaussian model for  $P(\theta)$ . Thus

$$P(\theta) = \frac{1}{\det(\pi \Lambda)} \exp(-\theta^\dagger \Lambda^{-1} \theta) \quad (2.7)$$

where  $\Lambda$  is a positive semi-definite covariance matrix. We choose  $P(\theta)$  to be zero mean, because all Gaussian prior probabilities can easily be transformed into this form.

## III. SUPER-RESOLUTION

In this section we assume that the imaging parameters  $\theta$  are exactly known, and we derive an iterative ‘re-estimation’ scheme for computing the cross section  $\sigma_0$  that maximises the posterior probability  $P(\sigma|g)$ . We call this ‘super-resolution’ because  $\sigma_0$  can have a higher spatial resolution than the image data, although this effect is significant only where there are small bright regions embedded in the image data [11].

### A. Posterior probability over cross sections

Suppose one asks the question: ‘What can I deduce about the cross section  $\sigma$ , given that I know the imaging system model (including all prior probabilities) and that I have available a dataset  $g$ ?’. Bayesian calculus says categorically: ‘The answer to your question is the posterior probability  $P(\sigma|g)$ ’ [2, 8]. Any reply that does not include enough information to construct  $P(\sigma|g)$  has not answered the stated question.  $P(\sigma|g)$  can therefore be used to deduce the answer to any other question that might have been asked. For instance, the  $\sigma$  that maximises  $P(\sigma|g)$  (let us call it  $\sigma_0$ ) is usually requested. If  $P(\sigma|g)$  has a single well-defined peak in  $\sigma$ , then  $\sigma_0$  can be used as a representative reconstruction of the cross section  $\sigma$  given the data  $g$ .

In order to calculate  $P(\sigma|g)$ , we must first of all use Bayes’ theorem to express  $P(\sigma|g)$  in terms of quantities that are defined in the imaging system model. Thus

$$\begin{aligned} P(\sigma|g) &= \frac{P(g|\sigma) P(\sigma)}{\int d\sigma' P(g|\sigma') P(\sigma')} \\ &= \frac{P(g|\sigma) P(\sigma)}{P(g)} \end{aligned} \quad (3.1)$$

which leads to

$$\sigma_0 = \arg \max_{\sigma} \{ \log[P(g|\sigma)] + \log[P(\sigma)] \} \quad (3.2)$$

$\sigma_0$  is a compromise between maximising  $P(g|\sigma)$  and maximising  $P(\sigma)$ . The  $P(g|\sigma)$  term attempts to maximise the probability that the data  $g$  could derive from  $\sigma$ , whereas the  $P(\sigma)$  term ignores the data entirely and attempts to maximise the prior probability that  $\sigma$  could have occurred irrespective of the data.  $\sigma_0$  is the cross section that best satisfies these conditions simultaneously. The information contained in  $\sigma_0$  is less than the information contained in  $P(\sigma|g)$  (except for the special case  $P(\sigma|g) = \delta(\sigma - \sigma_0)$ ).  $P(\sigma|g)$  contains everything that can be inferred from the data and the stated prior knowledge, whereas  $\sigma_0$  is merely the mode of  $P(\sigma|g)$ .

In [12] we presented a calculation of the derivatives of  $P(g|\sigma)$ , which provided a mechanism for iteratively computing  $\sigma_0$  by a ‘gradient ascent’ (i.e. infinitesimal update step sizes) scheme. We now improve upon these

results by deriving a ‘re-estimation’ (i.e. finite update step sizes) scheme. This turns out to be very similar to the empirical scheme that we suggested in [3], and, furthermore, it is very simple to relate this to the theory that we presented in [12].

### B. Lower bound on the posterior probability

We now maximise a quantity that is related to  $P(\sigma|g)$ , but which is constructed in such a way that it is much easier to maximise yet has the same local maxima as  $P(\sigma|g)$ . The method that we use is based on the estimate-maximise (EM) method of maximising likelihood functions [4].

As a preliminary step we shall transform our probabilities into log-probabilities, because this will make our subsequent derivations much easier to follow. Thus define  $L_1(\sigma|g)$  as

$$L_1(\sigma|g) \equiv \log[P(\sigma|g)] \quad (3.3)$$

Now we shall derive an important inequality that provides a lower bound for  $L_1(\sigma|g)$ :

$$\begin{aligned} L_1(\sigma'|g) &\stackrel{1}{=} \log \left[ \frac{P(g|\sigma') P(\sigma')}{P(g)} \right] \\ &\stackrel{2}{=} \log \left[ \frac{\int df P(g|f) P(f|\sigma') P(\sigma')}{P(g)} \right] \quad (3.4) \\ &\stackrel{3}{=} \log \left[ \int df P(f|g, \sigma) \frac{P(g|f) P(f|\sigma') P(\sigma')}{P(f|g, \sigma) P(g)} \right] \\ &\stackrel{4}{=} \log \left[ \int df P(f|g, \sigma) \frac{P(f|\sigma')}{P(f|\sigma)} \frac{P(g|\sigma) P(\sigma')}{P(g)} \right] \\ &\stackrel{5}{\geq} \int df P(f|g, \sigma) \log \left[ \frac{P(f|\sigma')}{P(f|\sigma)} \frac{P(g|\sigma) P(\sigma')}{P(g)} \right] \end{aligned}$$

We have used the following manipulations in the various steps of this derivation

Step 1. Use Bayes’ theorem as formulated in Equation 3.1 to express  $L_1(\sigma'|g)$  in terms of quantities that are specified in the imaging system model.

Step 2. Introduce the scattered field  $f$  as intermediate variables, as in Equation A3 of Appendix A.

Step 3. Introduce a factor of unity in the form  $\frac{P(f|g, \sigma)}{P(f|g, \sigma)}$ . This tautology prepares the integrand for stage 5 of the manipulation.

Step 4. Use the following

$$\begin{aligned} P(f|g, \sigma) &= \frac{P(g, f, \sigma)}{P(g, \sigma)} \\ &= \frac{P(g|f) P(f|\sigma)}{P(g|\sigma)} \quad (3.5) \end{aligned}$$

to rearrange the  $\frac{1}{P(f|g, \sigma)}$  term.

Step 5. Use Jensen’s inequality for convex functions

$$\log \left[ \int dx u(x) v(x) \right] \geq \int dx u(x) \log[v(x)] \quad (3.6)$$

(where  $u(x)$  must satisfy  $\int dx u(x) = 1$ ) to move the integral outside the logarithm. Equality holds if, and only if,  $P(f|\sigma') = P(f|\sigma)$  for all  $f$ . In our model, this requires  $\sigma = \sigma'$ .

It is convenient to rewrite the final inequality in Equation 3.4 in the form

$$\begin{aligned} L_1(\sigma'|g) &\geq L_2(\sigma', \sigma|g) + L_1(\sigma|g) \quad (3.7) \\ L_2(\sigma', \sigma|g) &\equiv \int df P(f|g, \sigma) \log \left[ \frac{P(f|\sigma')}{P(f|\sigma)} \right] + \log \left[ \frac{P(\sigma')}{P(\sigma)} \right] \end{aligned}$$

where the function  $L_2(\sigma', \sigma|g)$  contains all the  $\sigma'$  dependence of the lower bound of  $L_1(\sigma'|g)$ , and is constructed so that  $L_2(\sigma, \sigma|g) = 0$ . We summarise Equation 3.7 in Figure 2.

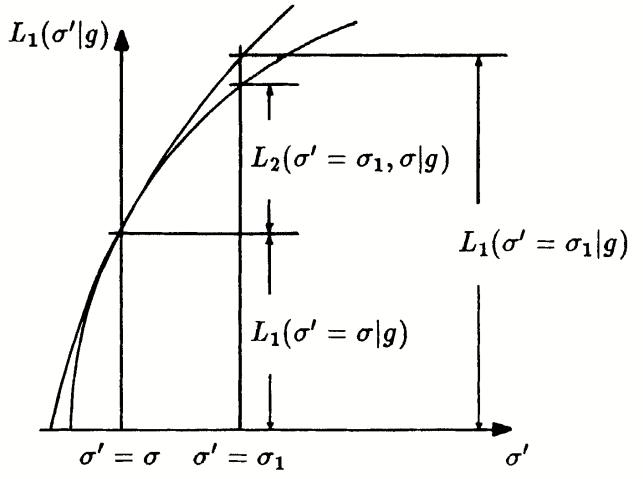


Figure 2: Jensen’s inequality applied to the posterior probability.

Instead of maximising  $L_1(\sigma'|g)$  with respect to  $\sigma'$ , we now maximise  $L_2(\sigma', \sigma|g)$  with respect to  $\sigma'$  (for some fixed  $\sigma$ ): this will recover a greatest lower bound for the true maximum of  $L_1(\sigma'|g)$ .

### C. Iteratively maximising the lower bound

Before proceeding any further, we present an outline of our proposed algorithm for maximising the posterior probability  $P(\sigma|g)$ . We do this to provide a concrete framework and motivation for the rather involved calculations that we perform later on.

Maximising  $P(\sigma|g)$  is equivalent to maximising  $L_1(\sigma|g)$ . In turn, maximising  $L_1(\sigma'|g)$  can be replaced by maximising  $L_2(\sigma', \sigma|g)$ , although this leads only to a greatest lower bound for  $L_1(\sigma'|g)$ , as given in Equation 3.7. If this greatest lower bound process is iterated by replacing  $\sigma$  with the optimum value of  $\sigma'$  that was found during the previous iteration, then the greatest lower bound converges towards a local maximum of  $L_1(\sigma'|g)$ .

1. In broad outline, our proposed algorithm for maximising  $L_1(\sigma'|g)$  is:
2. Initialisation step: Make an initial choice of  $\sigma$ .
3. Re-estimation step: Maximise  $L_2(\sigma', \sigma|g)$  with respect to  $\sigma'$ .
4. Update step:  $\sigma \rightarrow \sigma'$ .
5. Iteration step: If the update in  $\sigma$  does not satisfy some convergence criterion, then go to step 2.

$\sigma$  is now close to a local maximum of  $L_1(\sigma|g)$ . This does not guarantee that  $\sigma \approx \sigma_0$ , which is the required global maximum. Jensen's inequality, together with our model, guarantees only that fixed points of the re-estimation step are local maxima of  $L_1(\sigma|g)$ .

In Figure 3 we show three iterations of this algorithm.

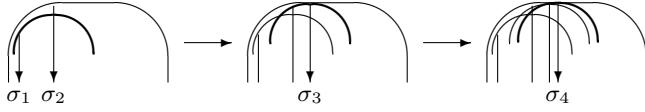


Figure 3: Three iterations of the maximisation algorithm.

We represent  $L_1(\sigma|g)$  as a large inverted cup, and  $L_2(\sigma', \sigma|g) + L_1(\sigma|g)$  as a small inverted cup. The initial choice is  $\sigma = \sigma_1$ , so  $L_2(\sigma', \sigma_1|g) + L_1(\sigma_1|g) \leq L_1(\sigma'|g)$  with equality at  $\sigma' = \sigma_1$ , as shown.  $\sigma \rightarrow \sigma_2$  is then the outcome of the re-estimation step. Accordingly, we show  $\sigma_2$  as the position of the maximum of  $L_2(\sigma', \sigma_1|g)$  in Figure 3. The remainder of Figure 3 depicts two further iterations, producing  $\sigma_3$  then  $\sigma_4$ , and making steady progress towards a maximum of  $L_1(\sigma|g)$ . It is geometrically obvious how this algorithm is guaranteed to find a local maximum of  $L_1(\sigma|g)$ .

#### D. Greatest lower bound on the posterior probability

The algorithm in Section III C relies critically on the re-estimation step. We therefore derive the stationary point(s) of  $L_2(\sigma', \sigma|g)$ , as defined in Equation 3.7.

From Equation 3.7 and Equation 2.3 we obtain

$$L_2(\sigma', \sigma|g) = \int df P(f|g, \sigma) \{ [-\log [\det(\pi\sigma')] - f^\dagger \sigma^{-1} f + \log [P(\sigma')]] - [\sigma' \rightarrow \sigma] \} \quad (3.8)$$

Neither the  $\log[\det(\pi\sigma')]$  term nor the  $\log[P(\sigma')]$  term depend on  $f$ , so their  $f$  integrals can be discarded (using  $\int df P(f|g, \sigma) = 1$ ).

However, the  $f^\dagger \sigma^{-1} f$  term requires more attention.  $P(f|g, \sigma)$  is given in Equation A5 of Appendix A, where we see that it is a Gaussian with mean  $\bar{f}$ . This makes the  $f^\dagger \sigma^{-1} f$  term of Equation 3.8 relatively easy to evaluate. To facilitate this integration we rearrange  $f^\dagger \sigma^{-1} f$  so that  $f$  appears only in the combination  $f - \bar{f}$ :

$$f^\dagger \sigma^{-1} f = (f - \bar{f})^\dagger \sigma'^{-1} (f - \bar{f}) + [\bar{f}^\dagger \sigma'^{-1} (f - \bar{f}) + CC] + \bar{f}^\dagger \sigma'^{-1} \bar{f} \quad (3.9)$$

where 'CC' denotes 'complex conjugate'. This yields

$$\int df P(f|g, \sigma) f^\dagger \sigma'^{-1} f = \text{tr} [\sigma'^{-1} C] + \bar{f}^\dagger \sigma'^{-1} \bar{f} \quad (3.10)$$

To obtain this result we rewrite the  $(f - \bar{f})^\dagger \sigma'^{-1} (f - \bar{f})$  term by using  $(f - \bar{f})^\dagger \sigma'^{-1} (f - \bar{f}) = \text{tr} [\sigma'^{-1} (f - \bar{f})(f - \bar{f})^\dagger]$ , and then using the covariance  $C$  of  $f - \bar{f}$  (see Equation A5 of Appendix A) to average under the trace. The term linear in  $f - \bar{f}$  averages to zero, by symmetry.

Finally, insert Equation 3.10 into Equation 3.8 to obtain

$$L_2(\sigma', \sigma|g) = [-\log [\det(\pi\sigma')] - \text{tr} [\sigma'^{-1} C] - \bar{f}^\dagger \sigma'^{-1} \bar{f} + \log [P(\sigma')]] - [\sigma' \rightarrow \sigma] \quad (3.11)$$

which may also be written in the form

$$\begin{aligned} L_2(\sigma', \sigma|g) &= \log \left[ \frac{P_{\text{eff}}(g|\sigma')}{P_{\text{eff}}(g|\sigma)} \right] + \log \left[ \frac{P_{\text{eff}}(\sigma')}{P_{\text{eff}}(\sigma)} \right] \\ P_{\text{eff}}(g|\sigma') &\propto \exp (-\bar{f}^\dagger \sigma'^{-1} \bar{f}) \\ P_{\text{eff}}(\sigma') &\propto \frac{P(\sigma')}{\det(\pi\sigma')} \exp (-\text{tr} [\sigma'^{-1} C]) \end{aligned} \quad (3.12)$$

where we include only the data dependent part in  $P_{\text{eff}}(g|\sigma')$ . For a diagonal  $\sigma$ -matrix (as in Equation 2.4)

this reduces to

$$\begin{aligned} P_{\text{eff}}(g|\sigma') &\propto \prod_{i=1}^m \exp\left(-\frac{|\bar{f}_i|^2}{\sigma_i'}\right) \\ P_{\text{eff}}(\sigma') &\propto P(\sigma') \prod_{i=1}^m \frac{1}{\pi\sigma_i'} \exp\left(-\frac{C_{ii}}{\sigma_i'}\right) \end{aligned} \quad (3.13)$$

We have now greatly simplified  $L_2(\sigma', \sigma|g)$  (Equation 3.8) because we have succeeded in integrating out the intermediate variable  $f$ .

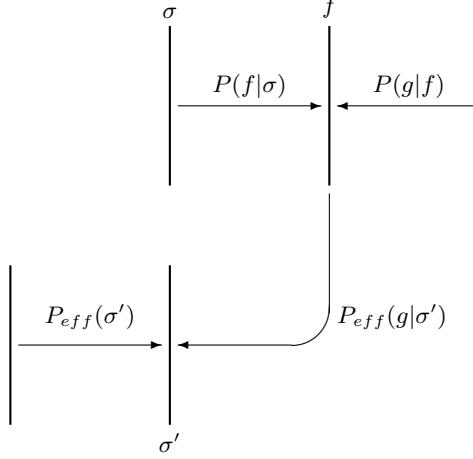


Figure 4: Network decomposition of reconstruction.

In Figure 4 we show the relationship between the various probabilities that we have introduced. Figure 4 contains two coupled inference processes. Firstly, the top half of Figure 4 constructs the posterior probability  $P(f|g, \sigma)$ . This is a Gaussian probability with mean  $\bar{f}$  that is normally used as the ‘maximum posterior probability’ reconstruction. Secondly, we input  $\bar{f}$  to the bottom half of Figure 4, to construct an effective posterior probability (proportional to  $P_{\text{eff}}(\sigma') P_{\text{eff}}(g|\sigma')$ ) whose

logarithm is (up to an additive constant)  $L_2(\sigma', \sigma|g)$  in Equation 3.11.

It is important to note that the second stage of the above inference process does not construct a true Bayesian posterior probability, because  $\sigma'$  is merely a mathematical convenience, not a physical reality. We have written the second stage in the style of a posterior probability merely to aid its interpretation. The true Bayesian posterior probability is  $P(\sigma|g)$ , which is maximised by an algorithm that consists of several iterations as described above.

We now differentiate each term in Equation 3.11 with respect to  $\sigma'$ . It turns out to be convenient to differentiate with respect to the transpose of the inverse matrix  $(\tilde{\sigma}')^{-1}$ . Differentiating the  $\log[\det(\pi\sigma')]$  term requires considerable care, because matrix-valued quantities do not necessarily commute, so we give a compact derivation in Appendix A. The other terms in Equation 3.11 pose no problem, and we obtain finally

$$\begin{aligned} \frac{\partial}{\partial(\tilde{\sigma}')^{-1}} \log[\det(\pi\sigma')] &= -\sigma' \\ \frac{\partial}{\partial(\tilde{\sigma}')^{-1}} \text{tr}[\sigma'^{-1} C] &= C \\ \frac{\partial}{\partial(\tilde{\sigma}')^{-1}} \bar{f}^\dagger \sigma'^{-1} \bar{f} &= \bar{f} \bar{f}^\dagger \end{aligned} \quad (3.14)$$

Combine Equation 3.14 with Equation 3.11 to obtain

$$\frac{\partial}{\partial(\tilde{\sigma}')^{-1}} L_2(\sigma', \sigma|g) = \sigma' - \bar{f} \bar{f}^\dagger - C + \frac{\partial}{\partial(\tilde{\sigma}')^{-1}} \log[P(\sigma')] \quad (3.15)$$

This is the central result from which our super-resolution ‘re-estimation’ algorithm can be deduced.

We may express Equation 3.15 in a form that is appropriate for a diagonal  $\sigma$ -matrix. Thus

---


$$\frac{\partial}{\partial \log[\sigma_i']} L_2(\sigma', \sigma|g) = \frac{|\bar{f}_i|^2 + C_{ii} - \sigma_i'}{\sigma_i'} + \frac{\partial}{\partial \log[\sigma_i']} \log[P(\sigma')] \quad (3.16)$$

$$\frac{\partial}{\partial \log[\sigma_i']} L_2(\sigma', \sigma|g) = \frac{[|\bar{f}_i|^2 - \langle |\bar{f}_i|^2 \rangle] + [\sigma_i - \sigma_i']}{\sigma_i'} + \frac{\partial}{\partial \log[\sigma_i']} \log[P(\sigma')] \quad (3.17)$$


---

where we have used the results in Equation A1 and Equation A2. The appropriate update  $\sigma \rightarrow \sigma'$  (to obtain the required greatest lower bound of  $L_1(\sigma'|g)$ ) is to replace  $\sigma$  by the value of  $\sigma'$  that makes the right hand side of Equation 3.16 (and Equation 3.17) equal to zero.

On the other hand, we may relate Equation 3.15 to the results of [12]. Thus we paraphrase equation (5.14)

of [12] in the notation of this paper as follows

$$\frac{\partial}{\partial \log[\sigma_i]} L_1(\sigma|g) = \frac{|\bar{f}_i|^2 - \langle |\bar{f}_i|^2 \rangle}{\sigma_i} + \frac{\partial}{\partial \log[\sigma_i]} \log[P(\sigma)] \quad (3.18)$$

When we set  $\sigma_i' = \sigma_i$  in Equation 3.17 we recover Equation 3.18, as required. This demonstrates the consistency

of the expressions for the gradient of  $P(\sigma|g)$  that are contained in this paper and in [12]. In a sense, [12] implements the bottom half of Figure 4 in a half-hearted fashion, by suggesting small ‘gradient ascent’ style updates using Equation 3.18, rather than by making the better ‘re-estimation’ style updates using the zero(s) of Equation 3.16 or Equation 3.17.

#### IV. SIMULTANEOUS SUPER-RESOLUTION AND AUTOFOCUSING

In this section we extend the results of Section III to account for uncertain imaging parameters  $\theta$ . We derive an iterative scheme for computing  $(\sigma_0, \theta_0)$  that maximises the posterior probability  $P(\sigma, \theta|g)$ . We call this ‘autofocus/super-resolution’ because we recover the imaging parameters  $\theta$  (i.e. we ‘focus’ the imaging sys-

tem) at the same time as we recover the cross section  $\sigma$  (i.e. ‘super-resolve’ the image).

##### A. Derivation of the re-estimation equation

Equation 3.2 becomes

$$(\sigma_0 \theta_0) = \arg \max_{\sigma, \theta} \{ \log[P(g|\sigma, \theta)] + \log[P(\sigma)] + \log[P(\theta)] \} \quad (4.1)$$

where we assume that the prior probabilities of  $\sigma$  and  $\theta$  are independent. Equation 3.3 becomes

$$L_1(\sigma, \theta|g) \equiv \log[P(\sigma, \theta|g)] \quad (4.2)$$

and the derivation in Equation 3.4 becomes

$$\begin{aligned} L_1(\sigma', \theta'|g) &= \log \left[ \frac{P(g|\sigma', \theta') P(\sigma') P(\theta')}{P(g)} \right] \\ &= \log \left[ \frac{\int df P(g|f, \theta') P(f|\sigma') P(\sigma') P(\theta')}{P(g)} \right] \\ &= \log \left[ \int df P(f|g, \sigma, \theta) \frac{P(g|f, \theta') P(f|\sigma') P(\sigma') P(\theta')}{P(f|g, \sigma, \theta) P(g)} \right] \\ &= \log \left[ \int df P(f|g, \sigma, \theta) \frac{P(f|\sigma')}{P(f|\sigma)} \frac{P(g|f, \theta')}{P(g|f, \theta)} \frac{P(g|\sigma, \theta) P(\sigma') P(\theta')}{P(g)} \right] \\ &\geq \int df P(f|g, \sigma, \theta) \log \left[ \frac{P(f|\sigma')}{P(f|\sigma)} \frac{P(g|f, \theta')}{P(g|f, \theta)} \frac{P(g|\sigma, \theta) P(\sigma') P(\theta')}{P(g)} \right] \end{aligned} \quad (4.3)$$

where we have manipulated the expressions in the same way as in Equation 3.4. Equation 3.7 becomes

$$\begin{aligned} L_1(\sigma', \theta'|g) &\geq L_{2,\sigma}(\sigma', \sigma, \theta|g) + L_{2,\theta}(\theta', \sigma, \theta|g) + L_1(\sigma, \theta|g) \\ L_{2,\sigma}(\sigma', \sigma, \theta|g) &\equiv \int df P(f|g, \sigma, \theta) \{ [-\log[\det(\pi\sigma')] - f^\dagger \sigma'^{-1} f + \log[P(\sigma')]] - [\sigma' \rightarrow \sigma] \} \\ L_{2,\theta}(\theta', \sigma, \theta|g) &\equiv \int df P(f|g, \sigma, \theta) \{ [[f^\dagger T'^\dagger N^{-1} g + CC] - f^\dagger T'^\dagger N^{-1} T' f + \log[P(\theta')]] - [\theta' \rightarrow \theta] \} \end{aligned} \quad (4.4)$$

which should be compared with the result in Equation 3.8.

The most important property of Equation 4.4 is the separation of the  $\sigma'$  and the  $\theta'$  dependences. This implies that we can optimise  $\sigma'$  and the  $\theta'$  independently. We take advantage of this by interleaving separate  $\sigma'$  and  $\theta'$  update iterations in our implementation of autofocus/super-resolution in Section V. The  $\sigma'$  re-estimation process is the same as in Section III, except

that the imaging operator  $T(\theta)$  is used with the current value of  $\theta$  inserted. On the other hand, the  $\theta'$  re-estimation process requires some further calculation in order to obtain the corresponding re-estimation equation.

There are two terms in  $L_{2,\theta}(\theta', \sigma, \theta|g)$  that require attention. The  $f^\dagger T'^\dagger N^{-1} T' f$  term can be obtained by a derivation analogous to Equation 3.9 and Equation 3.10 to yield

$$\int df P(f|g, \sigma, \theta) f^\dagger T'^\dagger N^{-1} T' f = \text{tr}[T'^\dagger N^{-1} T' C] + \bar{f}^\dagger T'^\dagger N^{-1} T' \bar{f} \quad (4.5)$$

The  $f^\dagger T'^\dagger N^{-1} g$  term simply yields

$$\int df P(f|g, \sigma, \theta) f^\dagger T'^\dagger N^{-1} g = \bar{f}^\dagger T'^\dagger N^{-1} g \quad (4.6)$$

Finally,  $L_{2,\theta}(\theta', \sigma, \theta|g)$  may be expressed (together with  $L_{2,\sigma}(\sigma', \sigma, \theta|g)$ , for completeness) as

$$\begin{aligned} L_{2,\sigma}(\sigma', \sigma, \theta|g) &\equiv [-\log[\det(\pi\sigma')] - \text{tr} [\sigma'^{-1} C] - f^\dagger \sigma'^{-1} f + \log[P(\sigma')]] - [\sigma' \rightarrow \sigma] \\ L_{2,\theta}(\theta', \sigma, \theta|g) &\equiv \left[ -\text{tr} [T'^\dagger N^{-1} T' C] - \bar{f}^\dagger T'^\dagger N^{-1} T' \bar{f} + [\bar{f}^\dagger T'^\dagger N^{-1} g + \text{CC}] + \log[P(\theta')] \right] - [\theta' \rightarrow \theta] \end{aligned} \quad (4.7)$$

We use the result for  $L_{2,\theta}(\theta', \sigma, \theta|g)$  to derive a re-estimation equation for  $\theta$ .

### B. Greatest lower bound on the posterior probability

We now introduce a linearised model of  $T(\theta)$

$$T(\theta) = T_0 + \sum_{i=1}^r \theta'_i T_i \quad (4.8)$$

where  $T_0$  is the ideal imaging operator, and  $T_i, i = 1, 2, \dots, r$  is a set of operators that we use to model the variation in  $T(\theta)$ . Furthermore, we use an explicit Gaussian model for  $P(\theta)$ , as specified in Equation 2.7. With these substitutions, Equation 4.7 for  $L_{2,\theta}(\theta', \sigma, \theta|g)$  becomes

$$\begin{aligned} L_{2,\theta}(\theta', \sigma, \theta|g) &= -\text{tr} \left[ \left( T_0^\dagger + \sum_{i=1}^r \theta'^*_i T_i^\dagger \right) N^{-1} \left( T_0 + \sum_{j=1}^r \theta'_j T_j \right) C \right] \\ &\quad - \bar{f}^\dagger \left( T_0^\dagger + \sum_{i=1}^r \theta'^*_i T_i^\dagger \right) N^{-1} \left( T_0 + \sum_{j=1}^r \theta'_j T_j \right) \bar{f} \\ &\quad + \left[ \bar{f}^\dagger \left( T_0^\dagger + \sum_{i=1}^r \theta'^*_i T_i^\dagger \right) N^{-1} g + \text{CC} \right] \\ &\quad - \log[\det(\pi\Lambda)] - \sum_{i,j=1}^r \theta'^*_i \theta'_j (\Lambda^{-1})_{ij} \\ &\quad - [\theta' \rightarrow \theta] \end{aligned} \quad (4.9)$$

Differentiating this result to obtain  $\frac{\partial L_{2,\theta}(\theta', \sigma, \theta|g)}{\partial \theta'^*_i}$  yields

$$\begin{aligned} \frac{\partial}{\partial \theta'^*_i} L_{2,\theta}(\theta', \sigma, \theta|g) &= -\text{tr} \left[ T_i^\dagger N^{-1} \left( T_0 + \sum_{j=1}^r \theta'_j T_j \right) C \right] - \bar{f}^\dagger T_i^\dagger N^{-1} \left( T_0 + \sum_{j=1}^r \theta'_j T_j \right) \bar{f} \\ &\quad + \bar{f}^\dagger T_i^\dagger N^{-1} g - \sum_{j=1}^r \theta'_j (\Lambda^{-1})_{ij} \end{aligned} \quad (4.10)$$

which depends linearly on the quantity of interest  $\theta'$ . We may simplify Equation 4.10 by making the definitions

$$\begin{aligned} A_{ij} &\equiv -\text{tr} [T_i^\dagger N^{-1} T_j C] - \bar{f}^\dagger T_i^\dagger N^{-1} T_j \bar{f} - (\Lambda^{-1})_{ij} \\ b_i &\equiv \text{tr} [T_i^\dagger N^{-1} T_0 C] + \bar{f}^\dagger T_i^\dagger N^{-1} T_0 \bar{f} - \bar{f}^\dagger T_i^\dagger N^{-1} g \end{aligned} \quad (4.11)$$

Note that the matrix  $A$  and the vector  $b$  depend only on the old parameter values  $(\sigma, \theta)$ .

Finally, we obtain the re-estimation equation for  $\theta$  by locating the stationary point  $\frac{\partial L_{2,\theta}(\theta', \sigma, \theta|g)}{\partial \theta'^*_i} = 0$ , which is the solution of the linear matrix equation

$$\sum_{j=1}^r A_{ij} \theta'_j = b_i \quad (4.12)$$

This is a remarkably simple re-estimation formula for the imaging system parameters  $\theta$ , which depends on quantities that are easily computed. For completeness, we quote the analogous result for real-valued imaging parameters  $\theta$  (which have a real symmetric covariance matrix  $\Lambda$ )

$$\begin{aligned} A_{ij} &\equiv \text{Re} \left\{ -\text{tr} \left[ T_i^\dagger N^{-1} T_j C \right] - \bar{f}^\dagger T_i^\dagger N^{-1} T_j \bar{f} \right\} - \frac{1}{2} (\Lambda^{-1})_{ij} \\ b_i &\equiv \text{Re} \left\{ \text{tr} \left[ T_i^\dagger N^{-1} T_0 C \right] + \bar{f}^\dagger T_i^\dagger N^{-1} T_0 \bar{f} - \bar{f}^\dagger T_i^\dagger N^{-1} g \right\} \end{aligned} \quad (4.13)$$

In our numerical simulations in Section V we use a single real imaging parameter, so Equation 4.13 will be the appropriate re-estimation equation to use. In Section V we also explain an important modification to the true noise covariance  $N$  that we found to be necessary in order to obtain convergence in a reasonable number of  $\theta$  re-estimation iterations.

## V. AN EXPLANATORY NUMERICAL EXAMPLE

In this section we apply our super-resolution/autofocus re-estimation method to a variety of one dimensional images (e.g. time series). We wish only to demonstrate the principle of our method, so we provide an explanatory numerical example, rather than extensive numerical simulations and refinement of algorithms. A detailed analysis of a low complexity algorithm for the super-resolution part of our theory can be found in [3].

### A. Imaging system - a defocused lens

For simplicity we consider the case of a scalar imaging parameter  $\theta$ . In order to ensure that our simulations are relevant to a commonly encountered practical situation, we consider the problem of super-resolution in a defocused linear imaging system. Specifically, we consider the case where a lens is defocused by  $\mathcal{O}(\text{depth of focus})$ .

We model a defocused lens as follows

$$\begin{aligned} T(\theta) &= \frac{1}{2c} \int_{-c}^{+c} dk \exp(ikx + i\theta k^2 x^2) \\ &\approx \frac{1}{2c} \int_{-c}^{+c} dk \exp(ikx) [1 + i\theta k^2 x^2] \\ &= T_0 + \theta T_1 \end{aligned} \quad (5.1)$$

where we have defined

$$\begin{aligned} T_0 &\equiv \frac{\sin(cx)}{cx} \\ -iT_1 &\equiv cx \sin(cx) + 2 \cos(cx) - \frac{2 \sin(cx)}{cx} \end{aligned} \quad (5.2)$$

The perturbation expansion in Equation 5.1 is appropriate when  $|\theta c^2 x^2| \leq 1$ , which ensures that the quadratic phase term  $\exp(i\theta k^2 x^2)$  is small at the edges ( $k = \pm c$ ) of the aperture. Physically, this condition requires that the lens be within  $\mathcal{O}(\text{depth of focus})$  of perfect focus. For  $|x| < \frac{\pi}{c}$  (i.e. within the main lobe) we therefore require  $|\theta| < \frac{1}{\pi^2} \simeq 0.1$ . In our numerical simulations we use  $\theta = 0.1$  in order to defocus the lens by  $\mathcal{O}(\text{depth of focus})$ .

We sample the output space of the imaging system defined in Equation 5.1 and Equation 5.2 at intervals  $\Delta x = 0.8 \frac{\pi}{c}$  (i.e. 0.8 of the Nyquist length), and we restrict our attention to an output space containing just five such samples. For simplicity, we do not use a continuous variable for position in input space, instead we merely sample it more finely than the output space. This strategy simplifies the software, without losing the essential properties of super-resolution and autofocusing. For our purposes we choose to sample the input space at intervals  $\Delta x = 0.4 \frac{\pi}{c}$  (i.e. two samples per output sample), which we superimpose on the output sample positions (and the midpoints of the intersample intervals). Thus we have 9 input samples. In summary, our sampling lattices are (in Nyquist units)

$$\begin{aligned} \text{input sample positions} &= \{-1.6, -1.2, -0.8, -0.4, 0.0, 0.4, 0.8, 1.2, 1.6\} \\ \text{output sample positions} &= \{-1.6, -0.8, 0.0, 0.8, 1.6\} \end{aligned} \quad (5.3)$$

These sampling lattices are not very long, but they are

sufficient for us to demonstrate the properties of our re-

estimation method.

### B. One-target case

We now perform the simplest possible numerical simulation to demonstrate autofocus/super-resolution, so our numerical simulation should be regarded as an explanatory example, rather than a detailed practical implementation of our method. Furthermore, we omit the prior probability terms  $P(\sigma)$  and  $P(\theta)$ , to perform a maximum likelihood (rather than maximum posterior probability) fit to the data. We thus rob our method of some of its power in order to produce a simple and uncluttered

demonstration of its effectiveness. It is easy to reintroduce the effect of  $P(\sigma)$  and  $P(\theta)$  into the re-estimation equations when more complicated problems (which *do* require prior knowledge to resolve ambiguous interpretations of the data) need to be solved.

We use the following point target embedded in a weak surrounding

$$\sigma = (1, 1, 1, 1, 1000, 1, 1, 1, 1) \quad (5.4)$$

which can produce a variety of scattered fields  $f$  distributed according to  $P(f|\sigma)$ . The particular realisation that occurred in our simulation happened to be

---


$$\begin{aligned} \text{Re}(f) &= (-0.4812, 0.0352, -0.5281, -0.6611, 23.8763, -0.0822, 0.1760, 0.4499, 0.6768) \\ \text{Im}(f) &= (-0.1526, -0.3091, 0.0117, 0.5673, -15.4640, 0.0822, 0.3951, 0.0117, 1.1462) \\ |f|^2 &= (0.2548, 0.0968, 0.2791, 0.7589, 809.2133, 0.0135, 0.1871, 0.2025, 1.7719) \end{aligned} \quad (5.5)$$


---

where we use the notation  $|f|^2$  informally to denote the vector of modulus squared components of  $f$ . We initialise the defocus parameter to  $\theta = 0.1$ , which is  $\mathcal{O}$ (depth of focus) away from perfect focus, which yields an image

$$\begin{aligned} \text{Re}(g) &= (-10.0906, 2.8887, 23.3954, 5.8459, -9.4112) \\ \text{Im}(g) &= (-6.5631, -4.2681, -15.2647, -4.3982, -5.0146) \\ |g|^2 &= (163.8664, 49.3264, 737.4455, 50.1384, 88.3842) \end{aligned} \quad (5.6)$$

We have not included the effects of noise yet.

The norm of this image is  $|g|^2 = 1119.0$ , and we add complex Gaussian noise to each image sample so that the expected norm of the total noise is  $\frac{|g|^2}{100}$  (i.e. the image signal to noise ratio (SNR) is 100, or equivalently 20dB). Note that we define SNR as a ratio of total energies, because it is this quantity that determines the information content of the data (the conventional way of measuring SNR by using the peak amplitude is not information theoretically meaningful). A SNR of 20dB requires that the variance of each Gaussian noise variate (one for each real and each imaginary part) is 1.1190 ( $= \frac{1119.0}{2 \times 5 \times 100}$ ). The noisy image so generated turned out to be

$$\begin{aligned} \text{Re}(g) &= (-10.6700, 3.2223, 22.3595, 5.7816, -7.9188) \\ \text{Im}(g) &= (-7.0723, -6.2404, -15.4110, -4.0880, -5.0673) \\ |g|^2 &= (163.8664, 49.3264, 737.4455, 50.1384, 88.3842) \end{aligned} \quad (5.7)$$

where we informally use the notation  $|g|^2$  to denote the vector of modulus squared components of  $g$ . This is the image that we use to explore the basic properties of our autofocus/super-resolution method.

In each of our experiments we initialise the cross section to be  $\sigma = (1, 1, 1, 1, 1, 1, 1, 1, 1)$ . This is a featureless first guess in which we do not even bother to initialise the overall normalisation correctly. We perform experiments

under three different focusing conditions: autofocus, defocus, and focus.

1. ‘Autofocus’ initialises  $\theta = 0$  and then recovers both  $\sigma$  and  $\theta$  from the data. This demonstrates the full power of our autofocus/super-resolution method.
2. ‘Defocus’ constrains  $\theta = 0$  (out of focus) and then recovers  $\sigma$  from the data. This demonstrates what super-resolution (without autofocus) would do if an inappropriate imaging operator were used.
3. ‘Focus’ constrains  $\theta = 0.1$  (in focus) and then recovers  $\sigma$  from the data. This demonstrates what super-resolution (without autofocus) would ideally do.

By comparing the results of each of these types of super-resolution we may determine the effectiveness of autofocus/super-resolution.

The re-estimation algorithm that we use in all of our experiments consists firstly of two iterations in which we re-estimate only  $\sigma$  by setting the gradient in Equation 3.16 to zero (and dropping the  $P(\sigma)$  term), followed by one iteration in which we re-estimate only the  $\theta$  by using Equation 4.12 with the results in Equation 4.13 inserted (and dropping the  $\Lambda$  term). Note that because we have only one defocus parameter, Equation 4.12 reduces to a scalar equation. We repeat this prescription of two iterations of  $\sigma$  re-estimation followed by one iteration of  $\theta$  re-estimation until the algorithm converges. Note that any sequence of re-estimations is permitted within our theoretical model, and they are all guaranteed to converge towards a local maximum of  $P(\sigma, \theta|g)$ .

For compactness we denote a  $\sigma$  re-estimation iteration as  $\Sigma$ , and a  $\theta$  re-estimation iteration as  $\Theta$ , so our

algorithm consists of  $\Theta\Sigma^2$  (reading from right to left) repeatedly applied to the data. We discovered this re-estimation algorithm by experimentation: it is one of the few schemes that we discovered that seems to avoid the problems of local maxima (and large flat regions as well) of  $P(\sigma, \theta|g)$ . We do not fully understand the reason why this particular algorithm works, and it is likely that it will need to be extended in order to handle more complicated situations successfully. Note that any re-estimation algorithm based on Equation 4.4 is guaranteed to make steady progress towards a local maximum of  $P(\sigma, \theta|g)$ , because at each stage it locates a greatest lower bound for  $P(\sigma, \theta|g)$ . The problem of designing an algorithm that makes rapid progress towards the global maximum of  $P(\sigma, \theta|g)$  is an extensive research topic in its own right, and it would require that the topography of  $P(\sigma, \theta|g)$  be investigated in detail.

To improve the rate of convergence we artificially increase the noise  $N \rightarrow 100N$  in  $\Theta$  (i.e. we evaluate Equation 4.13 using an artificially boosted noise level), because this permits much larger  $\theta$  updates to occur. If we left the noise at its correct value, then both the old and the new values of  $\theta$  would have to be consistent with the observed image to a high tolerance, which forces the  $\theta$  updates to be minuscule. On the other hand, if we boost the noise level, then consistency with the image need only be true to a low tolerance, so large  $\theta$  updates become possible. We do not bother to change this artificial noise level as the algorithm converges. No doubt some further improvements in the rate of convergence could be won at the cost of making the iteration schedule more complicated. Note that this prescription is not part of our basic theoretical machinery, so it must be used with extreme care. For instance, it is important *not* to use an increased noise level in  $\Sigma$ , because this would destroy our ability to super-resolve.

We show the ‘autofocus’ result in Figure 5, the ‘defocus’ result in Figure 6, and the ‘focus’ result in Figure 7. In each of these figures we present the final result as a bold line, and some of the intermediate results (before convergence) as dotted lines. Reading from right to left, we obtain the ‘autofocus’ result by applying  $\Sigma^2[\Theta\Sigma^2]^4$ , the ‘defocus’ result by applying  $\Sigma^{10}$ , and the ‘focus’ result by applying  $\Sigma^5$ . In Figure 5 the algorithm responds initially by fitting a W-shaped  $\sigma$  to the data, because at this stage  $\theta$  is incorrect. The algorithm then gradually corrects  $\theta$  and simultaneously recovers the correct cross section. In Figure 6 we show what would have happened had we not allowed  $\theta$  to be adjusted as in Figure 5: the W-shaped  $\sigma$  is retained. It is one of the main aims of this paper to remove such artefacts. In Figure 7 we present the ideal result that would be obtained if somehow we knew in advance what  $\theta$  was, and used it throughout the super-resolution iterations without change: the algorithm converges directly towards the correct solution.

Differences between Figure 5 and Figure 7 are largely because we have not run our algorithm all the way to convergence. Figure 5, Figure 6 and Figure 7 pro-

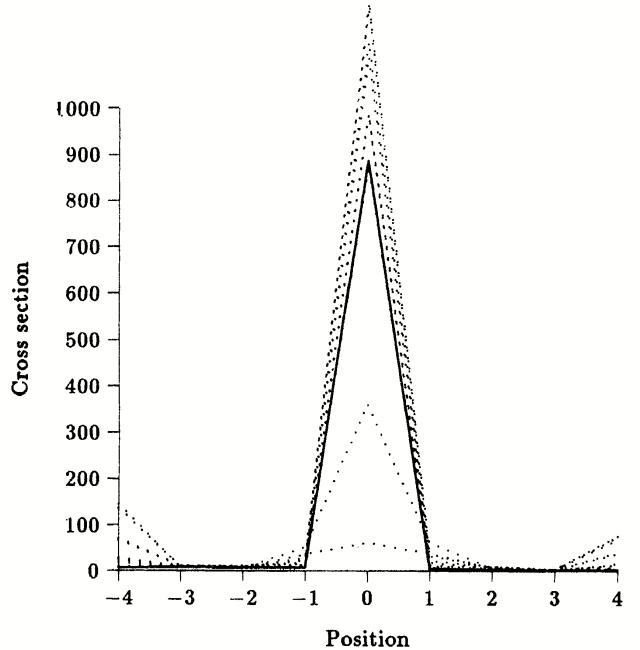


Figure 5: Super-resolve a target (autofocus).

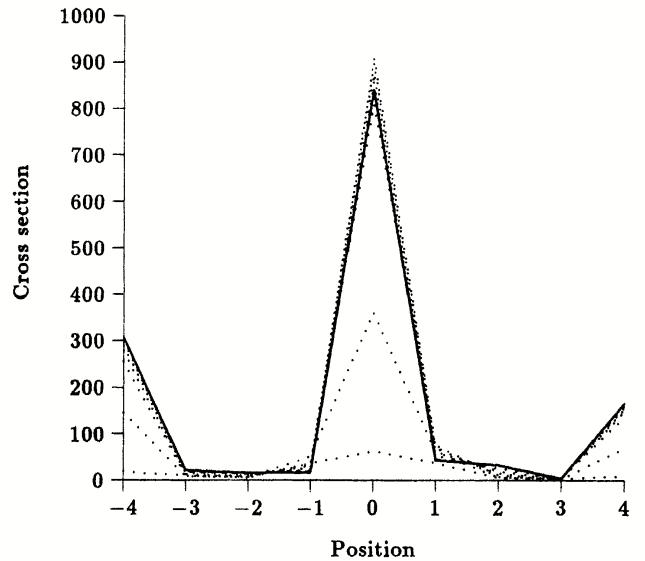


Figure 6: Super-resolve a target (defocus).

vide the simplest demonstration of the utility of our autofocus/super-resolution method. We commented in [10] that a scattered field that consisted of a point target placed between a pair of appropriately chosen point targets could produce a correctly focused image that mimicked the image that we would obtain from a single point target observed through a defocused imaging system. The artefacts that we observe in Figure 6 are consistent with this. If we have available any prior knowl-

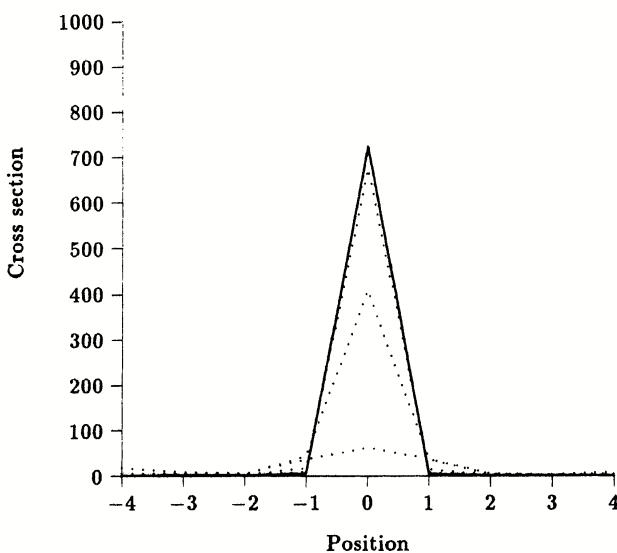


Figure 7: Super-resolve a target (focus).

edge, we can introduce  $P(\sigma)$  and/or  $P(\theta)$  terms into the re-estimation equations in order to reduce the effect of possibly ambiguous interpretations of the data.

### C. Two-target case

We now repeat the above experiment using two point targets embedded in a weak surrounding. The various vectors become

$$\begin{aligned}
 \sigma &= (1, 1, 1, 1000, 1, 1000, 1, 1, 1) \\
 \text{Re}(f) &= (-0.5751, -0.3091, -0.6768, -29.3196, 0.3482, -17.6908, -0.0430, 0.5125, 0.1682) \\
 \text{Im}(f) &= (0.0743, 0.5673, 0.4264, -1.1134, 0.1917, -7.2990, 0.3795, -0.3482, 0.3560) \\
 |f|^2 &= (0.3362, 0.4173, 0.6399, 860.8815, 0.1580, 366.2381, 0.1459, 0.3839, 0.1550) \\
 \text{Re}(g) &= (4.0108, -21.2726, -35.2127, -9.4149, 0.5343) \\
 \text{Im}(g) &= (6.8472, 5.4006, -6.9683, 4.4031, 2.1164) \\
 |g|^2 &= (62.9714, 481.6923, 1288.4935, 108.0277, 4.7648)
 \end{aligned} \tag{5.8}$$

where we have quoted only the noisy version of the image (which has a SNR of 20dB as before). Note that the fields scattered by the two point targets are approximately in phase, and the targets are 0.8 Rayleigh resolution cells apart.

As before we present the results that we obtain under three different focusing conditions. We show the ‘autofocus’ result in Figure 8, the ‘defocus’ result in Figure 9, and the ‘focus’ result in Figure 10. Reading from right to left, we obtain the ‘autofocus’ result by applying  $\Sigma^2[\Theta\Sigma^2]^4$ , the ‘defocus’ result by applying  $\Sigma^{10}$ , and the ‘focus’ result by applying  $\Sigma^{10}$ .

In Figure 8 the algorithm responds initially by constructing a  $\sigma$  that has a featureless bump in the centre. This gradually evolves to a bimodal form, which then converges towards the correct  $\sigma$ . In Figure 9 we do not update  $\theta$ , and the result is a featureless bump. In Figure 10, we apply the correct value of  $\theta$  throughout the super-resolution iterations, and the algorithm converges directly towards the correct solution. Differences between Figure 8 and Figure 10 are largely because we have not run the algorithm all the way to convergence.

These results demonstrate very clearly how

autofocus/super-resolution (Figure 8) can produce superior results to pure super-resolution (Figure 9).

### D. Discussion

The results of our numerical experiments show that our autofocus/super-resolution method performs as expected. In simple cases, we have demonstrated successful super-resolution when the lens is defocused by  $\mathcal{O}(\text{depth of focus})$ .

Our numerical experiments were successful because point targets have images with a lot of contrast, which are easy to autofocus. However, we anticipate that our method will not work in all circumstances. For instance, images containing clutter usually have a lower contrast than a point target image, and might therefore have several alternative feasible interpretations at different setting of the defocus parameter. The same problem would be encountered in any other method that had available the same information, because ambiguities can be resolved only by supplying appropriate additional information. In [10] we discussed some examples of target con-

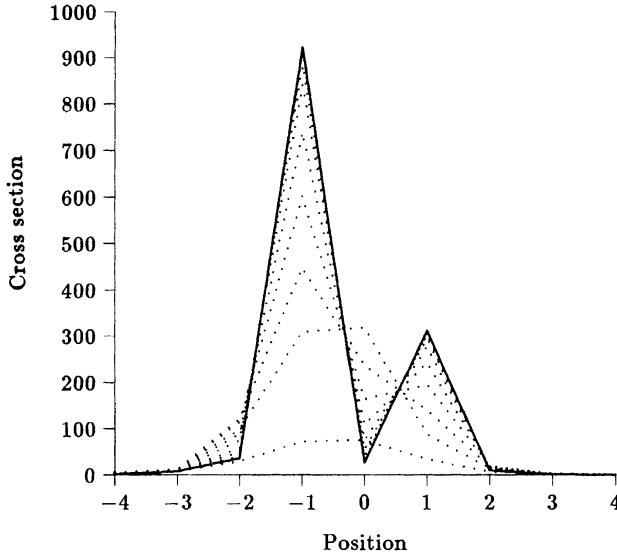


Figure 8: Super-resolve two targets (autofocus, in-phase).

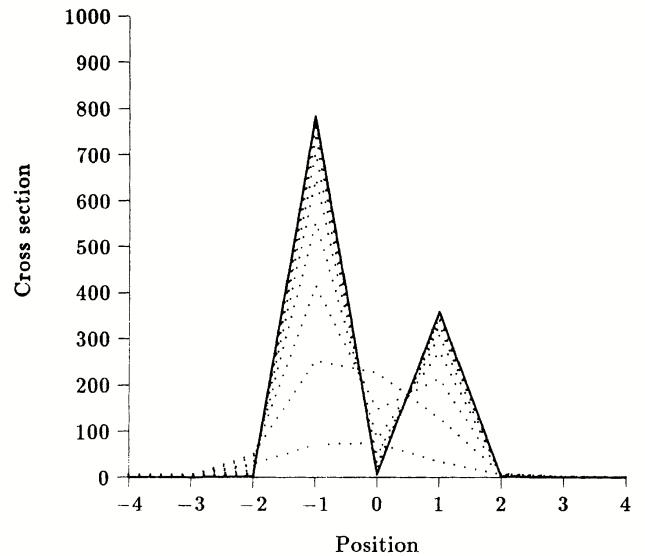


Figure 10: Super-resolve two targets (focus, in-phase).

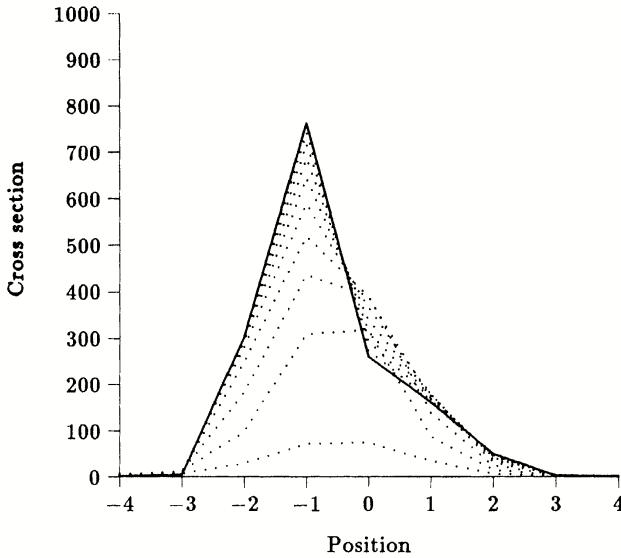


Figure 9: Super-resolve two targets (defocus, in-phase).

figurations that conspire to fool an autofocus algorithm in this way.

The imaging parameter re-estimation method in Equation 4.12 caters for a vector of parameters, so we could extend the model in Equation 5.1 to include the effect of other terms if we wished.

We did not introduce prior knowledge terms  $P(\sigma)$  or  $P(\theta)$  into our numerical simulations. Such terms can only improve the performance of our method, provided that the prior knowledge is correct.

## VI. CONCLUSIONS

Firstly, we have shown how to use Jensen's inequality (in the form of an estimate-maximise (EM) algorithm) to maximise the posterior probability  $P(\sigma|g)$  for recovering a scattering cross section  $\sigma$  from a coherent image  $g$ : we call this 'super-resolution'. Secondly, we have extended this result to maximise the posterior probability  $P(\sigma, \theta|g)$  for simultaneously recovering both the imaging system parameters  $\theta$  and the cross section  $\sigma$ : we call this 'autofocus/super-resolution'. The first result proves the validity of our heuristic iterative super-resolution algorithm [3]. The second result extends this algorithm to compensate for the effects of an uncertain imaging system.

Although our results apply to coherent imaging systems in general, they have the potential to super-resolve synthetic aperture radar (SAR) images. SAR can be modelled as a linear imaging system, and anomalous motion of the transmitter/receiver can be modelled (in simple cases) as a simple defocusing of the type exemplified in Section V. There is a contrast maximisation autofocus method [5, 15] that corrects the focus of a SAR to within  $\mathcal{O}(\text{depth of focus})$  of the correct value, but this is inadequate for successful super-resolution to be attempted [10]. Our current autofocus/super-resolution method shows potential for correcting these residual focusing uncertainties, and thus offers the possibility of a robust super-resolution algorithm for SAR image analysis.

## VII. ACKNOWLEDGEMENTS

I thank J S Bridle for explaining to me the advantages that ‘re-estimation’ methods have over ‘gradient ascent’ methods.

### Appendix A: Miscellaneous derivations

First of all let us make some definitions, whose purpose will become clear later on

$$\begin{aligned} M &\equiv T\sigma T^\dagger + N \\ C^{-1} &\equiv \sigma^{-1} + T^\dagger N^{-1} T \\ \bar{f} &\equiv \sigma T^\dagger M^{-1} g \\ &= CT^\dagger N^{-1} g \end{aligned} \quad (\text{A1})$$

whence

$$\begin{aligned} M^{-1} &= N^{-1} - N^{-1} T C T^\dagger N^{-1} \\ C &= \sigma - \sigma T^\dagger M^{-1} T \sigma \\ \langle \bar{f} \bar{f}^\dagger \rangle &= \sigma T^\dagger M^{-1} T \sigma \end{aligned} \quad (\text{A2})$$

where  $\langle \dots \rangle$  denotes an average over the  $g$  as described by the model probabilities (which is not necessarily the same as the average over observed  $g$ , because one’s model can be, and usually is, wrong).

#### Deriving $P(g|\sigma)$

We derive the probability  $P(g|\sigma)$  that a cross section  $\sigma$  can give rise to dataset  $g$ . Potentially, there are two types of hidden variable to consider: the scattered field  $f$ , and the imaging parameters  $\theta$ , which we showed in Figure 1. Let us assume that  $\theta$  is known, so that it can be removed from the problem, to yield

$$P(g|\sigma) = \int df P(g|f) P(f|\sigma) \quad (\text{A3})$$

Equation A3 represents the top half of Figure 1 with  $f$  integrated out. We obtain  $P(f|\sigma)$  from Equation 2.3 and  $P(g|f)$  from Equation 2.6 (with  $\theta$  removed). The integrand of Equation A3 is Gaussian, so it is easy to perform the integral. The steps are as follows:

$$\begin{aligned} P(g|\sigma) &= \frac{1}{\det(\pi N) \det(\pi \sigma)} \int df \exp \left[ -f \sigma^{-1} f - (g - Tf)^\dagger N^{-1} (g - Tf) \right] \\ &= \frac{1}{\det(\pi N) \det(\pi \sigma)} \int df \exp \left[ -(f - \bar{f})^\dagger C^{-1} (f - \bar{f}) + g^\dagger (-N^{-1} + N^{-1} T C T^\dagger N^{-1}) g \right] \\ &= \frac{\det(\pi C)}{\det(\pi N) \det(\pi \sigma)} \exp [g^\dagger (-N^{-1} + (N^{-1} - M^{-1})) g] \\ &= \frac{1}{\det(\pi M)} \exp (-g^\dagger M^{-1} g) \end{aligned} \quad (\text{A4})$$

In the last step we simplify the normalisation factor by noting (*a posteriori*) that the result must be a correctly normalised probability.  $P(g|\sigma)$  is a zero mean complex Gaussian probability with covariance  $M$ . The form of  $M$  given in Equation A1 is very simple: the  $T\sigma T^\dagger$  piece is the linearly filtered scattered field covariance matrix  $\sigma$ , and the  $N$  piece is the data noise covariance. These terms sum together because the data noise is statistically independent of the scattered field.

#### Deriving $P(f|g, \sigma)$

We derive the posterior probability  $P(f|g, \sigma)$  that a scattered field  $f$  can occur, given that both the scattering cross section  $\sigma$  and the data  $g$  are known. The steps in the derivation are

$$\begin{aligned} P(f|g, \sigma) &= \frac{P(g|f) P(f|\sigma)}{P(g|\sigma)} \\ &= \frac{\det(\pi M)}{\det(\pi N) \det(\pi \sigma)} \exp \left[ -f \sigma^{-1} f - (g - Tf)^\dagger N^{-1} (g - Tf) + g^\dagger M^{-1} g \right] \\ &= \frac{\det(\pi M)}{\det(\pi N) \det(\pi \sigma)} \exp \left[ -(f - \bar{f})^\dagger C^{-1} (f - \bar{f}) + g^\dagger (M^{-1} - N^{-1} + N^{-1} T C T^\dagger N^{-1}) g \right] \\ &= \frac{1}{\det(\pi C)} \exp \left[ -(f - \bar{f})^\dagger C^{-1} (f - \bar{f}) \right] \end{aligned} \quad (\text{A5})$$

In the last step we simplify the normalisation factor by noting (*a posteriori*) that the result must be a correctly normalised probability.  $P(f|g, \sigma)$  is a Gaussian with mean  $\bar{f}$  and covariance  $C$ .

#### Differentiating $\log \det$

We differentiate the logarithm of the determinant of a matrix-valued quantity. We use this in order to differentiate Equation 3.11, so we present the derivation using an appropriate notation.

$$\begin{aligned}
 \delta \log[\det(\pi\sigma')] &\stackrel{1}{=} -\log[\det(\pi(\sigma'^{-1} + \delta\sigma'^{-1}))] + \log[\det(\pi\sigma'^{-1})] \\
 &\stackrel{2}{=} -\text{tr}[\log(\sigma'^{-1}(1 + \sigma'\delta\sigma'^{-1}))] + \text{tr}[\log(\sigma'^{-1})] \\
 &\stackrel{3}{=} -\text{tr}[\log(1 + \sigma'\delta\sigma'^{-1})] \\
 &\stackrel{4}{\approx} -\text{tr}[\sigma'\delta\sigma'^{-1}]
 \end{aligned} \tag{A6}$$

We justify the various stages of this manipulation as follows:

Step 1. Matrix invert  $\sigma'$ , which introduces a minus sign outside the logarithm function. In order to calculate the derivative, write the difference that results from changing  $\sigma'^{-1}$  infinitesimally.

Step 2. Use the identity  $\log[\det(X)] = \text{tr}[\log(X)]$ .

Step 3. Use the identity  $\log(XY) = \log(X) + \log(Y) +$  (commutator terms from the Baker-Hausdorff identity) to obtain  $\text{tr}[\log(XY)] = \text{tr}[\log(X)] + \text{tr}[\log(Y)]$  which causes a pair of terms to cancel, leaving only the infinitesimal part. Note that the trace of any commutator is zero.

Step 4. Expand the logarithm using  $\log(1 + X) = X + \mathcal{O}(X^2)$ .

- [1] L E Baum, *An inequality and associated maximisation technique in statistical estimation for probabilistic functions of Markov processes*, *Inequalities* **3** (1972), no. 1, 1–8.
- [2] R T Cox, *Probability, frequency and reasonable expectation*, *American Journal of Physics* **14** (1946), no. 1, 1–13.
- [3] L M Delves, G Pryde, and S P Luttrell, *A super-resolution algorithm for SAR images*, *Inverse Problems* **4** (1988), no. 3, 681–703.
- [4] A P Dempster, N M Laird, and D B Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, *Journal of the Royal Statistical Society B* **39** (1977), no. 1, 1–38.
- [5] I P Finley and J W Wood, *An investigation of synthetic aperture radar autofocus*, Memorandum 3790, Royal Signals and Radar Establishment, Malvern, 1985.
- [6] S Geman and D Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6** (1984), no. 6, 721–741.
- [7] E Jakeman and P N Pusey, *A model for non-Rayleigh sea echo*, *IEEE Transactions on Antennas and Propagation* **24** (1976), no. 6, 806–814.
- [8] H Jeffreys, *Theory of probability*, Clarendon Press, Oxford, 1939.
- [9] S P Luttrell, *Prior knowledge and object reconstruction using the best linear estimate technique*, *Optica Acta* **32** (1985), no. 6, 703–716.
- [10] ———, *A super-resolution model for synthetic aperture radar*, Memorandum 3785, Royal Signals and Radar Establishment, Malvern, 1985.
- [11] ———, *The relationship between super-resolution and phase imaging of SAR data*, Research Note BS1/41, Royal Signals and Radar Establishment, Malvern, 1987.
- [12] ———, *The inverse cross section problem for complex data*, *Inverse Problems* **5** (1989), no. 1, 35–50.
- [13] ———, *The theory of Bayesian super-resolution of coherent images: a review*, *International Journal of Remote Sensing* **12** (1991), no. 2, 303–314.
- [14] S P Luttrell and C J Oliver, *Prior knowledge in synthetic aperture radar processing*, *Journal of Physics D: Applied Physics* **19** (1986), no. 3, 333–356.
- [15] C J Oliver, *Synthetic aperture radar imaging*, *Journal of Physics D: Applied Physics* **22** (1989), no. 7, 871–890.
- [16] K D Ward, *Compound representation of high resolution sea clutter*, *Electronics Letters* **17** (1981), no. 16, 561–565.



# The Theory of Bayesian Super-Resolution of Coherent Images: A Review \*

S. P. LUTTRELL

*Royal Signals and Radar Establishment, St Andrews Road, Malvern, Worcestershire WR14 3PS, England*

We review some theoretical work on super-resolution of coherent images from a Bayesian point of view. The well known singular value decomposition super-resolution method emerges as a special case, and it is extended in order to derive a practical iterative super-resolution algorithm.

## I. INTRODUCTION

Super-resolution is the name given to techniques which enhance the resolving power of imaging systems beyond that permitted by the Rayleigh resolution criterion. This is achieved at the cost of having to introduce prior knowledge about the object which is being imaged.

In order to take account of prior knowledge in a consistent fashion we shall derive the super-resolution equations from a Bayesian point of view. Denote the scattering and imaging operations as  $\sigma \rightarrow f \rightarrow g$  where the cross-section  $\sigma$  produces a scattered field  $f$ , which is then imaged to produce an image  $g$ . Bayes' theorem may then be used to solve various inverse problems posed in terms of these quantities. We shall study two such problems.

The first is the inverse scattered field problem (recovering  $f$  from  $g$ ) which becomes  $\Pr(f|g) \propto \Pr(g|f) \Pr(f)$  where  $\Pr(f)$  represents our prior knowledge of the scattered field (which depends on our prior estimate of  $\sigma$ ). In particular, when the probability density functions (p.d.f.s) are Gaussian, the peak of  $\Pr(f|g)$  gives the minimum mean square error (MMSE in the Euclidean sense) estimate of  $f$  given  $g$ . This is the basis of a simple and successful super-resolution technique.

The second is the inverse cross-section problem (recovering  $\sigma$  from  $g$ ) which becomes  $\Pr(\sigma|g) \propto \Pr(g|\sigma) \Pr(\sigma)$ . The peak of  $\Pr(\sigma|g)$  can be found by a hill-climbing approach in which inverse problem 1 is solved repeatedly for various estimates of  $\sigma$ .

## II. FUNDAMENTALS

### A. A resumé of Bayesian inference

Throughout this paper we shall adopt a Bayesian approach to super-resolution. The motivation for this is consistency of inference, which is afforded uniquely by Bayesian calculus. The notions of Bayesian calculus are highly non-trivial, so we shall first of all review them. In particular, we shall explain what a Bayesian probability

is, and how it is (usually) different from a frequentist probability [5].

We shall now summarise a simple argument in favour of the Bayesian approach which is due to [18], which is, in turn, based on [2]. This summary is intended to be merely an indication of the simplicity and elegance of the axioms from which all of Bayesian calculus flows. Thus, impose the following consistency conditions on the probabilities related to a pair of states  $f$  and  $g$  of an arbitrary system

$$\text{prefer } f \text{ to } g \text{ AND prefer } g \text{ to } h \Rightarrow \text{prefer } f \text{ to } h$$

$$\begin{aligned} \Pr(f, g) &= \mathcal{F}_1(\Pr(f), \Pr(g|f)) \\ \Pr(f) &= \mathcal{F}_2(\Pr(f)) \end{aligned} \quad (2.1)$$

In Equation 2.1  $\mathcal{F}_1$  and  $\mathcal{F}_2$  are yet to be determined functions. These deceptively simple consistency conditions are all that is required, because they lead to Cox's theorems [2] which obliges us to reason in terms of Bayesian calculus

$$\begin{aligned} \Pr(f) + \Pr(f) &= 1 & 0 \leq \Pr(f) \leq 1 \\ \Pr(\text{falsity}) &= 0 \\ \Pr(\text{certainty}) &= 1 \\ \Pr(f, g) &= \Pr(f) \Pr(g|f) \end{aligned} \quad (2.2)$$

This axiomatic approach to constructing and interpreting probabilities has become known as the Bayesian approach. A Bayesian uses probabilities to describe incomplete information (rather than randomness), and has an epistemological (rather than ontological) view of the universe. On the other hand, a frequentist uses a probability to describe randomness. Unnecessary arguments between Bayesians and frequentists arise because of their different interpretations of probability.

We shall usually be concerned with probabilities of continuous vector-valued quantities, so we shall speak of p.d.f.s. Thus define  $\Pr(Y|X)$  as the p.d.f. of  $Y$  conditioned on  $X$ . We may use  $\Pr(Y|X)$  to proceed deductively from  $X$  to  $Y$ . Bayes' theorem then gives  $\Pr(X|Y)$  in the form

$$\Pr(X|Y) = \frac{\Pr(Y|X) \Pr(X)}{\sum_{X'} \Pr(Y|X') \Pr(X')} \quad (2.3)$$

which we use to proceed inductively from  $Y$  to  $X$ . The most important feature about Equation 2.3 is that  $\Pr(X|Y)$  depends on an additional factor  $\Pr(X)$ , the prior p.d.f. of  $X$ . Thus induction is influenced by prior expectations.

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This paper appeared in International Journal of Remote Sensing, 1991, vol. 12, no. 2, pp. 303-314. © Controller, Her Majesty's Stationery Office, 1991.

## B. A coherent scattering and imaging model

The model which we shall use has been presented before [6, 10]. Let us denote the scattering cross-section (or the object) as  $\sigma$ , the scattered field (or the object field) as  $f$ , and the data as  $g$ . We shall assume coherent illumination, in which case a commonly-used physical scattering model is a random walk process in which a large number of scattering centres (in the vicinity of the field point) each contribute randomly-phased pieces of scattered field, leading to a Gaussian p.d.f. for the total local scattered field

$$\Pr(f|\sigma) = \frac{1}{\det \pi\sigma} \exp(-f^\dagger \sigma^{-1} f) \quad (2.4)$$

From the point of view of interpretations of probability this is a *frequentist* model. We represent the cross-section as a matrix-valued quantity (in this case a field-field covariance), because this permits us to model situations where distinct scatterers produce correlated scattered fields. The covariance (i.e.  $\sigma$ ) of  $f$  is denoted as  $\langle ff^\dagger \rangle_{\Pr(f|\sigma)}$  where  $\langle \dots \rangle$  denotes an average, and the p.d.f. used in the averaging process is indicated outside the  $\langle \dots \rangle$ . We shall restrict our attention to diagonal  $\sigma$  given by  $\sigma = \text{diag}(\sigma_1, \sigma_2, \dots)$  (uncorrelated). This models a conventional uncorrelated scattering process where the matrix nature of  $\sigma$  is physically superfluous, but mathematically convenient.

An alternative derivation of Equation 2.4 may be obtained by using maximum entropy techniques. There is an axiomatic derivation of this approach given in [16, 17] which is based on consistency requirements, and there is a simplified version of their arguments given in [18]. This axiomatic approach does not rely on the frequency interpretation of the entropy of a system being the logarithm of the effective number of states available to that system. Thus suppose that the only knowledge we have concerning the form of  $\Pr(f|\sigma)$  is expressed as:

$$\begin{aligned} \langle f \rangle_{\Pr(f|\sigma)} &= 0 && (\text{mean}) \\ \langle f f^\dagger \rangle_{\Pr(f|\sigma)} &= \sigma && (\text{covariance}) \end{aligned}$$

The maximum entropy p.d.f. consistent with these two constraints is the same as was given in Equation 2.4. The maximum entropy method guarantees that  $\Pr(f|\sigma)$  is the least committal [4] p.d.f. consistent with the constraints, so we can be certain that no spurious information is encoded in the p.d.f. In other words,  $\Pr(f|\sigma)$  expresses precisely our degree of ignorance about  $f$  given  $\sigma$  - this is how one should construct degrees of reasonable belief given knowledge expressed as a set of constraints. In this approach we do not need to introduce a frequentist interpretation of probability, so the p.d.f. is the type of object which a Bayesian prefers to use.

The Gaussian scattering model does not entirely express our knowledge of the physical situation. In regions where there are localised strongly-scattering centres (e.g. targets) the field scattered by a single scatterer might

dominate the scattered amplitude, thus invalidating the simple random walk approach. However, any such additional information about the form of  $\Pr(f|\sigma)$  would lead to a less committal maximum entropy p.d.f., so we err on the side of caution if we use Equation 2.4 instead. We shall adopt the cautious approach of Equation 2.4 because the simplicity of the data processing scheme which eventually emerges justifies this simplification.

We shall model the imaging process using a linear operator  $T$  (this models the system point spread function (PSF)), and we shall assume that the data  $g$  is corrupted by zero mean Gaussian noise  $n$  with covariance matrix  $N$ . Thus

$$g = T f + n \quad (2.5)$$

which is described by the p.d.f.

$$\Pr(g|f) = \frac{1}{\det \pi N} \exp\left[-(g - Tf)^\dagger N^{-1} (g - Tf)\right] \quad (2.6)$$

We shall restrict our attention to isotropic  $N$  (i.e. white noise) with a variance  $\nu$  given by  $N = \nu I$ . This frequentist approach to noise modelling can be replaced by a maximum entropy approach, similar to that used to derive Equation 2.4 and leading to the same result as Equation 2.6.

## III. THE INVERSE OBJECT FIELD PROBLEM

The object of this section is to use Bayes' theorem to obtain the posterior for the scattered field given the data  $g$  and the prior  $\Pr(f|\sigma)$ . We thus use Bayes' theorem in the following form:

$$\begin{aligned} \Pr(g, f, \sigma) &= \Pr(f|g, \sigma) \Pr(g, \sigma) \\ &= \Pr(g|f, \sigma) \Pr(f, \sigma) \end{aligned} \quad (3.1)$$

$$\begin{aligned} \Pr(f|g, \sigma) &= \frac{\Pr(g|f) \Pr(f|\sigma)}{\Pr(g|\sigma)} \\ &= \frac{\Pr(g|f) \Pr(f|\sigma)}{\int df' \Pr(g|f') \Pr(f'|\sigma)} \end{aligned} \quad (3.2)$$

where we have used the Markov chain property of  $\sigma \rightarrow f \rightarrow g$  to obtain  $\Pr(g|f, \sigma) = \Pr(g|f)$  (i.e.  $g$  is conditionally independent of  $\sigma$  given that  $f$  is known).

The  $f$  dependence of Equation 3.2 is contained entirely in the numerator. The posterior reduces to essentially the product of the prior  $\Pr(f|\sigma)$  and the data dependent part  $\Pr(g|f)$ . Thus Bayes' theorem provides a rigorous means of combining the prior knowledge with the new knowledge acquired from a dataset  $g$ . If the data is very noisy then  $\Pr(g|f)$  will depend only weakly on  $f$ , and the prior will dominate the behaviour of the posterior. This is exactly as expected because noisy data contains little information, so our inference must default to the prior data.

### A. Minimum mean square error reconstruction, $f_{\text{rec}}$

We shall now derive the explicit form of  $\Pr(f|g, \sigma)$ . We may now substitute Equation 2.4 and Equation 2.6 into

Equation 3.2 to obtain, after some algebra

$$\Pr(f|g, \sigma) = \frac{1}{\det \pi \mathbf{C}} \exp \left[ - \left( f - \langle f \rangle_{\Pr(f|g, \sigma)} \right)^{\dagger} \mathbf{C}^{-1} \left( f - \langle f \rangle_{\Pr(f|g, \sigma)} \right) \right] \quad (3.3)$$

whose mean and covariance are given by

$$\begin{aligned} \langle f \rangle_{\Pr(f|g, \sigma)} &= \mathbf{C} T^{\dagger} \mathbf{N}^{-1} g \\ &= \sigma T^{\dagger} \mathbf{M}^{-1} g \\ \langle f f^{\dagger} \rangle_{\Pr(f|g, \sigma)} - \langle f \rangle_{\Pr(f|g, \sigma)} \langle f^{\dagger} \rangle_{\Pr(f|g, \sigma)} &= C \end{aligned} \quad (3.4)$$

where we have defined  $\mathbf{C}^{-1}$  and  $\mathbf{M}$  as

$$\begin{aligned} \mathbf{C}^{-1} &\equiv T^{\dagger} \mathbf{N}^{-1} T + \sigma^{-1} \\ \mathbf{M} &\equiv T \sigma T^{\dagger} + \mathbf{N} \end{aligned} \quad (3.5)$$

The posterior mean  $\sigma T^{\dagger} \mathbf{M}^{-1} g$  is non-zero due to the effect of the  $\Pr(g|f)$  term in Equation 3.2. The posterior covariance  $\mathbf{C}$  is not the same as the prior covariance  $\sigma$  for the same reason.

For practical purposes a single representative  $f$  is usually chosen as the outcome of this inference process. The natural choice is

$$f_{\text{rec}} = \langle f \rangle_{\Pr(f|g, \sigma)} \quad (3.6)$$

where the subscript ‘rec’ denotes the reconstructed scattered field. This definition of  $f_{\text{rec}}$  locates it at the peak of the posterior.

Note that  $f_{\text{rec}}$  is generated from  $g$  by applying a linear operator  $\sigma T^{\dagger} \mathbf{M}^{-1}$ . In an alternative approach to obtaining a representative reconstruction of the scattered field we could find the linear operator which filters  $g$  in such a way as to recover the MMSE reconstruction. This is the approach of Wiener which leads to precisely the same linear filter that we have derived here from the Bayesian viewpoint [6]. This equivalence follows from the use of mean and covariance as the only constrained statistics in both cases. However the Bayesian approach

has the advantage of being explicitly related to the consistency conditions in Equation 2.1, and thus is more amenable to consistent generalisation to other  $\Pr(f|\sigma)$  and  $\Pr(g|f)$ .

This reconstruction scheme depends critically on having available complex data  $g$  and accurate knowledge of the PSF  $T$ . Studies of how this Bayesian technique can be applied to synthetic aperture radar (SAR) data have been presented elsewhere: SAR data has a phase which can contain valuable information [9], and the PSF of the Royal Signals and Radar Establishment (RSRE) SAR has been estimated from ‘targets of opportunity’ [8]. This technique may be used to super-resolve the data provided the following two conditions are met [6]:

1. The image of the target must not be larger than a few resolution cells in size.
2. The signal-to-clutter ratio of (the modulus of ) the data must be of order of 10 or more.

These two requirements ensure that the image of the target is a small bright blob.

### B. Information channels

We may gain a deeper understanding of Equation 3.3 by performing a singular value decomposition (SVD). The details of the appropriate weighted SVD have been presented [10], and we reproduce the essential steps in the appendix. This SVD decomposes the linear imaging operator  $T$  into a number of channels, and thus simplifies the detailed analysis of what Bayes’ theorem is achieving. The posterior  $\Pr(f|g, \sigma)$  now becomes

$$\Pr(f|g, \sigma) = \frac{1}{\det \pi \mathbf{C}} \exp \left[ -f_{\text{invis}}^{\dagger} \mathbf{C}^{-1} f_{\text{invis}} \right] \exp \left[ -(f_{\text{vis}} - f_{\text{rec}})^{\dagger} \mathbf{C}^{-1} (f_{\text{vis}} - f_{\text{rec}}) \right] \quad (3.7)$$

where we have decomposed  $f$  as  $f = (f_{\text{vis}}, f_{\text{invis}})$ , and  $f_{\text{vis}}$  ( $f_{\text{invis}}$ ) is the subspace of  $f$  which is ‘(in)visible’ to

the data  $g$ . Such (in)visibility is a consequence of the low dimensionality of the data - we can recover from  $g$  up to

$\dim g$  linear degrees of freedom of  $f$  (they are visible), all other degrees of freedom must remain unknown (they are invisible). In practice there may be fewer than  $\dim g$  visible degrees of freedom because the data is oversampled - this could be inverted and used as a definition of oversampling. The density of visible degrees of freedom can be used to define a generalised resolution cell such that there is, on average, one visible degree of freedom per resolution cell.

In Equation 3.7 we have defined  $f_{\text{invis}}$  to lie in the subspace where the posterior  $\Pr(f|g, \sigma)$  is not influenced by the data: i.e. it depends only on the prior  $\Pr(f|\sigma)$ .  $f_{\text{vis}}$  then occupies the orthogonal complement subspace. This notation is explained in detail in the appendix. The change in (inverse) covariance matrix  $\sigma^{-1} \rightarrow \sigma^{-1} + T^\dagger N^{-1} T$  in passing from  $f_{\text{invis}}$  to  $f_{\text{vis}}$  causes the visible degrees of freedom to be more tightly constrained because  $T^\dagger N^{-1} T$  is a non-negative definite.

Referring to the appendix, an interesting special case is recovered in the limit  $\lambda_m \gg \nu$  (signal energy much greater than noise energy in channel  $m$ ), because then Equation 3.7 contains one or more hard constraints on  $f_{\text{vis}}$ . These hard constraints arise because effectively  $N \approx \mathbf{0}$ , so  $C^{-1}$  will contain some very large eigenvalues (assuming that  $T$  is non-trivial), which in turn implies that the posterior has infinitesimal thickness in certain directions. This leads to  $f_m' \approx \frac{g_m'}{\sqrt{\lambda_m}}$  with a variance  $\frac{\nu}{\lambda_m}$ . This approximation is precisely what is used in conventional non-Bayesian SVD reconstruction methods which reconstruct  $f$  using  $f_{\text{rec}} = \sqrt{\sigma} \sum_{m=0}^M \frac{g_m'}{\sqrt{\lambda_m}} u^m$  where the  $\lambda_m$  are arranged in order of decreasing size, and  $m_0$  is chosen as large as possible but consistent with  $\lambda_{m_0} \gg \nu$  [1]. This sharp cut-off is a crude form of regularisation which can produce results indistinguishable from the full Bayesian approach when the spectrum of  $\lambda_m$  (ordered according to size) indeed cuts off rapidly. The Bayesian approach provides an interpolation between the  $\lambda_m \gg \nu$  and  $\lambda_m \ll \nu$  regimes based on the fundamental consistency requirements of Equation 2.1.

As anticipated in the title of this section we have performed a decomposition of the imaging system into information channels labelled by  $m$ . An important feature is the prior knowledge dependence of these channels [11, 12]:  $\sigma$  plays a central rôle in determining these channels. The interpretation of what we mean by *information* is delicate, so we shall give a detailed theoretical information treatment in Section III C. Intuitively the reason for the prior knowledge dependence of the information channel structure is obvious, because a channel can carry *useful* information about  $f$  only if we have not already received such information from our prior knowledge, by definition.

### C. Mutual information

We shall now make our notion of useful information rigorous [7]. Mutual information is a theoretical information measure that is used to quantify the degree of mutual dependence of variables. It is an objective information measure, because it does not depend on the ordering of the state spaces of the variables. Thus mutual information is a good candidate for a useful information measure, because it allows us to quantify the extent to which knowledge (expressed as a p.d.f.) of the state of one variable constrains our knowledge of another variable.

Define the entropy  $H[r]$  of a variable  $r$  as

$$H[r] = - \int dr \Pr(r) \log [\Pr(r)] \quad (3.8)$$

An explanation of the use of entropy and related quantities in information theory can be found in [15]. We shall paraphrase their interpretation of entropy in frequentist language because it provides an intuitive operational definition of entropy.

1. Draw  $s$  samples independently from  $\Pr(r)$  to form the ordered set  $R$ ,  $R \equiv \{r_1, r_2, \dots, r_s\}$ .
2. Form the set of all such ordered sets  $\mathcal{R}$ ,  $\mathcal{R} \equiv \{R : \text{all distinct sets of samples}\}$ .
3. As  $s \rightarrow \infty$  we find that  $s H[r]$  measures the logarithm of the number of sets  $R$  contained in  $\mathcal{R}$  which have a high probability of occurring (i.e. are not outliers).
4. There are thus  $s H[r]$  distinct ‘messages’ of length  $s$  which are produced by the memoryless ‘source’, which may be used for communicating information.

The interpretation of  $H[r]$  as the logarithm of the effective number of states available to a single sample  $r$  is now apparent.

The mutual information  $I[f; g]$  between a pair of variables  $f$  and  $g$  is defined in three equivalent ways as

$$I[f; g] = H[f] + H[g] - H[f, g] \quad (3.9)$$

$$= H[f] - H[f|g] \quad (3.10)$$

$$= H[g] - H[g|f] \quad (3.11)$$

In the frequency interpretation, each of these expressions is a difference of entropies which measures the logarithm of *the ratio of* the effective number of states available jointly to  $f$  and  $g$ , firstly considered as independent variables then considered as dependent variables. Thus  $I[f; g]$  qualifies as an objective measure of the mutual dependence of  $f$  and  $g$ .

Starting from Equation 3.10 or Equation 3.11 we obtain, respectively

$$\begin{aligned} I[f; g] &= \log \left[ \frac{\det \sigma}{\det C} \right] \\ &= \log \left[ \frac{\det M}{\det N} \right] \end{aligned} \quad (3.12)$$

where we have used the definitions in Equation 3.5. A determinant of a covariance matrix measures the effective (squared) volume occupied by the peak of the corresponding multidimensional complex Gaussian p.d.f., i.e. the product of the eigenvalues (or squared widths) of its covariance matrix. Thus  $\det(\sigma)$  measures the squared volume of the prior  $\Pr(f|\sigma)$ , and  $\det(\mathbf{C})$  measures the squared volume of the posterior  $\Pr(f|g, \sigma)$ . The ratio of these is the factor by which the squared volume of the prior is ‘squeezed’ by the data in order to become the posterior. Similarly,  $\det(\mathbf{M})$  measures the squared volume of the overall (i.e. averaged over all scattered field configurations) data p.d.f.  $\Pr(g|\sigma)$  where

$$\begin{aligned}\Pr(g|\sigma) &= \int df \Pr(g|f) \Pr(f|\sigma) \\ &= \frac{1}{\det \pi \mathbf{M}} \exp [-g^\dagger \mathbf{M}^{-1} g]\end{aligned}\quad (3.13)$$

and  $\det(\mathbf{N})$  measures the squared volume of the p.d.f. of the additive data noise, so the ratio of these is the number of times the noise p.d.f. can fit inside the overall data p.d.f. Both of these are clearly information measures, and they are equal to each other according to Equation 3.12.

We may decompose  $I[f; g]$  into components by applying the SVD of the appendix to yield

$$I[f; g] = \sum_m \log \left[ \frac{\lambda_m + \nu}{\nu} \right] \quad (3.14)$$

This decomposition makes it clear that the SVD (as presented in the appendix) decomposes the imaging system into information channels each of which contributes *independently*  $\log \left[ \frac{\lambda_m}{\nu} + 1 \right]$  to the overall mutual information  $I[f; g]$ . What we informally called information channels in Section III B may therefore now be raised to the status of true information channels, in the technical sense of information as defined in information theory.

#### IV. THE INVERSE CROSS-SECTION PROBLEM

In Section III we studied a Bayesian solution to the inverse field problem. Recovery of the scattered field is almost universally the favoured approach in super-resolution theory. However, this is fundamentally flawed because image interpretation should not be expressed in terms of the object (in this case the scattered field) which *transports* information, rather we should express ourselves in terms of the objects (in this case scattering centres) which are the *source* of the information. We should therefore be concerned with the underlying scattering cross-section, so here we shall present some results which are required in the Bayesian analysis of the inverse cross-section problem.

We shall now use a form of Bayes’ theorem which is appropriate to this new inverse problem,

$$\Pr(\sigma|g) = \frac{\Pr(g|\sigma) \Pr(\sigma)}{\int d\sigma' \Pr(g|\sigma') \Pr(\sigma')} \quad (4.1)$$

where  $\Pr(g|\sigma)$  is the overall data p.d.f. given in Equation 3.13, and  $\Pr(\sigma)$  is a *prior* which specifies our prior knowledge of  $\sigma$ .

The  $\sigma$  dependence of  $\Pr(\sigma|g)$  comes from both  $\Pr(\sigma)$  and the matrix  $\mathbf{M}$  in  $\Pr(g|\sigma)$ , and unfortunately does not reduce to a Gaussian behaviour amenable to linear techniques. We shall postpone the issue of prior knowledge and the  $\Pr(\sigma)$  term for now, in order to concentrate on the structure of the data collection term  $\Pr(g|\sigma)$ . This term is nonlinear in  $\sigma$ , so we shall develop its Taylor expansion in powers of  $\sigma$ . In order to do this we need to differentiate  $\Pr(g|\sigma)$  with respect to  $\sigma$ , and the next two sections are concerned with this task.

##### A. First derivative of $\Pr(g|\sigma)$

For convenience we shall define  $L(g|\sigma)$  (the log likelihood ratio) as

$$\begin{aligned}L(g|\sigma) &\equiv \log [\Pr(g|\sigma)] \\ &= -g^\dagger \mathbf{M}^{-1} g - \log [\det(\mathbf{M})] + \text{constant}\end{aligned}\quad (4.2)$$

In order to differentiate this we need the following results

$$\begin{aligned}\frac{\partial \mathbf{M}^{-1}}{\partial \sigma_i} &= -\mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \mathbf{M}^{-1} \\ \frac{\partial \log [\det(\mathbf{M})]}{\partial \sigma_i} &= \frac{\partial \text{tr} [\log(\mathbf{M})]}{\partial \sigma_i} \\ &= \text{tr} \left[ \frac{\partial \log(\mathbf{M})}{\partial \sigma_i} \right] \\ &= \text{tr} \left[ \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \right]\end{aligned}\quad (4.3)$$

where it is important that we maintain the order of the various powers of  $\mathbf{M}$  because  $\mathbf{M}$  is an operator. Differentiating Equation 4.2 with respect to  $\sigma_i$  and using the results of Equation 4.3 yields

$$\frac{\partial L(g|\sigma)}{\partial \sigma_i} = g^\dagger \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \mathbf{M}^{-1} g - \text{tr} \left[ \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \right] \quad (4.4)$$

We may simplify this by using

$$\begin{aligned}\frac{|f_{\text{rec},i}|^2}{\sigma_i^2} &= g^\dagger \mathbf{M}^{-1} [T T^\dagger]_i \mathbf{M}^{-1} g \\ &= g^\dagger \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \mathbf{M}^{-1} g\end{aligned}$$

$$\begin{aligned}\frac{|f_{\text{rec},i}|^2}{\sigma_i^2} &= g^\dagger \mathbf{M}^{-1} [T T^\dagger]_i \mathbf{M}^{-1} g \\ &= g^\dagger \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \mathbf{M}^{-1} g\end{aligned}\quad (4.5)$$

where  $[TT^\dagger]_i$  is  $TT^\dagger$  with its internal summation restricted to position  $i$  alone. We may average  $|f_{\text{rec},i}|^2$  over data according to  $\Pr(g|\sigma)$  (i.e. for all scattered field and data noise configurations), to yield

$$\frac{\langle |f_{\text{rec},i}|^2 \rangle_{\Pr(g|\sigma)}}{\sigma_i^2} = \text{tr} \left[ \mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \sigma_i} \right] \quad (4.6)$$

where we have used  $\langle gg^\dagger \rangle_{\Pr(g|\sigma)} = \mathbf{M}$  from Equation 3.13, and the cyclic property  $\text{tr}(XY) = \text{tr}(YX)$  of traces. Physically,  $\langle |f_{\text{rec},i}|^2 \rangle_{\Pr(g|\sigma)}$  is the average of the intensity MMSE reconstructions over all possible data which arise from a given cross-section. It will therefore be rather smoother than any of the individual  $|f_{\text{rec},i}|^2$  in Equation 4.5. Essentially this smoothing process is a speckle reduction technique, and it leads to an average intensity reconstruction which closely mimics (a smoothed version of) the underlying cross-section.

Equation 4.4 now becomes (in dimensionless form)

$$\frac{\partial L(g|\sigma)}{\partial \log(\sigma_i)} = \frac{|f_{\text{rec},i}|^2 - \langle |f_{\text{rec},i}|^2 \rangle_{\Pr(g|\sigma)}}{\sigma_i} \quad (4.7)$$

This is appealing because it indicates that we can increase the log likelihood by increasing  $\sigma_i$ , in those regions where the intensity reconstruction is greater than the smoothed intensity reconstruction (and vice versa).

This forms the basis of an iterated inverse cross-section algorithm in which an estimate of the underlying cross-section  $\sigma$  is used to estimate the scattered field  $f_{\text{rec}}$  using Equation 3.6, which in turn is used to update  $\sigma$  using Equation 4.7. Such a scheme is heuristic because it attempts to maximise the likelihood function  $\Pr(g|\sigma)$  rather than the posterior  $\Pr(\sigma|g)$ . Only  $\Pr(\sigma|g)$  includes prior knowledge  $\Pr(\sigma)$  of  $\sigma$  as required of a Bayesian approach. In extensive numerical simulations we find that

omitting the prior knowledge, causes  $\sigma$  to ‘condense out’ into a number of isolated spikes if the algorithm is run through a sufficient number of iterations. This is a consequence of the uncontrolled positive feedback loop which connects  $f_{\text{rec},i}$  and  $\sigma$  in the iterated algorithm: appropriate prior knowledge can temper the effects of this loop. Note that gradient ascent of  $\Pr(\sigma|g)$  would use

$$\frac{\partial \log[\Pr(\sigma|g)]}{\partial \log(\sigma_i)} = \frac{\partial \log[\Pr(g|\sigma)]}{\partial \log(\sigma_i)} + \frac{\partial \log[\Pr(\sigma)]}{\partial \log(\sigma_i)} \quad (4.8)$$

In Equation 4.8 the first term is given by Equation 4.7 and the second term depends on the gradient of the prior. The second term modifies the gradient of the likelihood function, and we can simulate the general effect of this without actually calculating it in detail. Thus, a heuristic form of prior knowledge of  $\sigma$  can be introduced which modifies the result in Equation 4.7 as appropriate. Such an approach has been used successfully to super-resolve imagery in [11, 12].

## B. Higher derivatives of $\Pr(g|\sigma)$

Simple reconstruction algorithms need only use the first derivative of  $L(g|\sigma)$ . However, more sophisticated update procedures can be employed if the second derivative matrix (Hessian) is computed in closed form. We can continue to differentiate  $L(g|\sigma)$  in order to develop higher-order terms in its Taylor expansion, but we shall merely summarise the main result the details can be found in [10]. Define  $D_n L$  as

$$D_n L(g|\sigma) \equiv \left[ \prod_{s=1}^n \frac{\partial}{\partial \log(\sigma_{i_s})} \right] L(g|\sigma) \equiv \partial_{i_1} \partial_{i_2} \cdots \partial_{i_n} L(g|\sigma) \quad (4.9)$$

After some algebra [10] we obtain

---


$$\begin{aligned} D_n L(g|\sigma) &= B_n - \frac{\langle B_n \rangle}{n} \\ B_n &\equiv (-1)^{n-1} \sum_{\text{perms}} g^\dagger \mathbf{M}^{-1} (\partial_{p_1} \mathbf{M}) \mathbf{M}^{-1} (\partial_{p_2} \mathbf{M}) \mathbf{M}^{-1} \cdots \mathbf{M}^{-1} (\partial_{p_n} \mathbf{M}) \mathbf{M}^{-1} g \\ \langle B_n \rangle &\equiv (-1)^{n-1} \sum_{\text{perms}} \text{tr} [\mathbf{M}^{-1} (\partial_{p_1} \mathbf{M}) \mathbf{M}^{-1} (\partial_{p_2} \mathbf{M}) \mathbf{M}^{-1} \cdots \mathbf{M}^{-1} (\partial_{p_n} \mathbf{M})] \end{aligned} \quad (4.10)$$


---

where the summation is over permutations of the indices  $(p_1, p_2, \dots, p_n)$ . Note that Equation 4.10 reduces to Equation 4.4 when  $n = 1$ , as expected.

## V. CONCLUSIONS

We have presented a consistent Bayesian approach to super-resolution of SAR data. One could simply define a Wiener filter and then perform a MMSE reconstruction from the data, but this would deny the wealth of inter-

predictive power that is available to the Bayesian. Why should you minimise the squared error rather than anything else? The Bayesian approach rejects the naive notion of distance measures, and replaces them with physically and information theoretically derived priors and posteriors. In the special case of Gaussian p.d.f.s. this happens to reduce to a MMSE reconstruction prescription. In more subtle cases the advantages of Bayesian calculus become overwhelming, because it alone is guaranteed to provide a consistent approach to inference [2].

The results of Section III and Section IV can be combined to produce a practical super-resolution algorithm [11, 12]. Equation 3.6 provides the MMSE estimate of the scattered field given the data  $g$  and the prior  $\Pr(f|\sigma)$ . This result can be inserted into Equation 4.7 to provide the direction in which to vary  $\sigma$  in order to increase the log likelihood function  $L(g|\sigma)$ . A smoothed version of  $\sigma$  can be used as an estimate of  $\langle |f_{\text{rec},i}|^2 \rangle_{\Pr(g|\sigma)}$  in Equation 4.7 (see Section IV A). Furthermore, by using an iterative approach, we may obtain a fast numerical implementation of this approach to super-resolution [3, 13, 14].

Much work remains to be done in this field. There are uncertainties about the stability of the PSF measured in [8] - it would be desirable to have a super-resolution method which simultaneously recovered both the target and PSF. However, given the limited data which is available, a great deal of prior knowledge about the point spread function would have to be introduced in order to obtain useful results. In the longer term the nature of the prior  $\Pr(\sigma)$  needs further study, because we currently introduce prior knowledge of  $\sigma$  in a somewhat heuristic fashion, rather than by rigorous Bayesian application of  $\Pr(\sigma)$ .

## Appendix A: The singular value decomposition (SVD)

It is very convenient to define a weighted SVD of  $T$  as follows:

$$\begin{aligned} T\sqrt{\sigma}u^m &= \sqrt{\lambda_m}v^m \\ \sqrt{\sigma}T^\dagger v^m &= \sqrt{\lambda_m}u^m \\ u^{\dagger m}u^n &= \delta_{mn} \\ v^{\dagger m}v^n &= \delta_{mn} \end{aligned} \quad (\text{A1})$$

Indices  $m$  and  $n$  label the order of the singular function (or vector). The  $u^m$  singular *functions* are used to

decompose the *continuous* scattered field. These components are mapped to the  $v^m$  which are the corresponding *discrete* data space singular *vectors*. The singular value  $\sqrt{\lambda_m}$  gives the amplitude attenuation factor in channel  $m$ , and the weight  $\sqrt{\sigma}$  provides an amplitude scale for the scattered field.

We may use Equation A1 to write

$$\begin{aligned} T\sqrt{\sigma} &= \sum_m \sqrt{\lambda_m} v^m u^{\dagger m} \\ \sqrt{\sigma}T^\dagger &= \sum_m \sqrt{\lambda_m} u^m v^{\dagger m} \end{aligned} \quad (\text{A2})$$

and we shall expand  $f$  and  $g$  using

$$\begin{aligned} f &= f_{\text{invis}} + \sqrt{\sigma} \sum_m f'_m u^m \\ g &= \sum_m g'_m v^m \\ f'_m &\equiv u^{\dagger m} \frac{1}{\sqrt{\sigma}} f \\ u^{\dagger m} \frac{1}{\sqrt{\sigma}} f_{\text{invis}} &\equiv 0 \\ g'_m &\equiv v^{\dagger m} g \end{aligned} \quad (\text{A3})$$

where  $f_{\text{invis}}$  is the orthogonal complement piece of  $f$  which does not project onto the  $u^m$  (i.e. it is invisible to the data), and the  $f'_m$  constitute the expansion coefficients of the SVD. The subspace spanned by the SVD is all *potentially* visible to the data, provided that  $\sqrt{\lambda_m} \gg \nu, \forall m$ . Whence  $\langle f \rangle_{\Pr(f|g,\sigma)}$  in Equation 3.3 is given by

$$\langle f \rangle_{\Pr(f|g,\sigma)} = \sqrt{\sigma} \sum_m \frac{\sqrt{\lambda_m}}{\lambda_m + \nu} g'_m u^m \quad (\text{A4})$$

and the covariance of the posterior  $C^{-1}$  in Equation 3.5 is given by

$$(\sqrt{\sigma} C^{-1} \sqrt{\sigma})_{ij} = \sum_m \frac{\lambda_m}{\nu} u_i^m u_j^{*m} + \delta_{ij} \quad (\text{A5})$$

where the  $i$  and  $j$  suffixes are used to label object space. Thus the posterior  $\Pr(f|g,\sigma)$  may be simplified by combining the results given in this appendix to yield

---


$$(f - \langle f \rangle_{\Pr(f|g,\sigma)})^\dagger C^{-1} (f - \langle f \rangle_{\Pr(f|g,\sigma)}) = f_{\text{invis}}^\dagger \sigma^{-1} f_{\text{invis}} + \sum_m \left| f'_m - \frac{\sqrt{\lambda_m}}{\lambda_m + \nu} g'_m \right|^2 \frac{\lambda_m + \nu}{\lambda_m} \quad (\text{A6})$$

This separates the contributions from the visible and the invisible subspaces.

- 
- [1] M Bertero and E R Pike, *Resolution in diffraction limited imaging, a singular value analysis I. The case of coherent illumination*, Optica Acta **29** (1982), no. 6, 727–746.
- [2] R T Cox, *Probability, frequency and reasonable expectation*, American Journal of Physics **14** (1946), no. 1, 1–13.
- [3] L M Delves, G Pryde, and S P Luttrell, *A super-resolution algorithm for SAR images*, Inverse Problems **4** (1988), no. 3, 681–703.
- [4] E T Jaynes, *Prior probabilities*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 3, 227–241.
- [5] H Jeffreys, *Theory of probability*, Clarendon Press, Oxford, 1939.
- [6] S P Luttrell, *Prior knowledge and object reconstruction using the best linear estimate technique*, Optica Acta **32** (1985), no. 6, 703–716.
- [7] ———, *The use of transinversion in the design of data sampling schemes for inverse problems*, Inverse Problems **1** (1985), no. 3, 199–218.
- [8] ———, *The complex point spread function of the RSRE SAR*, Memorandum 4079, Royal Signals and Radar Establishment, Malvern, 1987.
- [9] ———, *The relationship between super-resolution and phase imaging of SAR data*, Research Note BS1/41, Royal Signals and Radar Establishment, Malvern, 1987.
- [10] ———, *The inverse cross section problem for complex data*, Inverse Problems **5** (1989), no. 1, 35–50.
- [11] S P Luttrell and C J Oliver, *Prior knowledge in synthetic aperture radar processing*, Journal of Physics D: Applied Physics **19** (1986), no. 3, 333–356.
- [12] ———, *The role of prior knowledge in coherent image processing*, Philosophical Transactions of the Royal Society A **324** (1988), no. 1579, 397–407.
- [13] G C Pryde, L M Delves, and S P Luttrell, *A comparative study of the AMT DAP and of Transputer array architectures for the super-resolution of synthetic aperture radar images*, Proceedings of conference on parallel processing (Cambridge), University Press, 1988, pp. 340–350.
- [14] ———, *The performance of a parallel super-resolution algorithm for synthetic aperture radar images*, Proceedings of IMA conference on mathematics in signal processing (Oxford), University Press, 1988, pp. 739–747.
- [15] C E Shannon and W Weaver, *The mathematical theory of communication*, University of Illinois Press, Urbana, 1949.
- [16] J E Shore and R W Johnson, *Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross entropy*, IEEE Transactions on Information Theory **26** (1980), no. 1, 26–37.
- [17] ———, *Comments on and correction to ‘axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy’ (Jan 80 26–37) (Correspondence)*, IEEE Transactions on Information Theory **29** (1983), no. 6, 942–943.
- [18] J Skilling (ed.), *Maximum entropy and Bayesian methods*, Kluwer Academic Publishers, Dordrecht, 1989.

# Code Vector Density in Topographic Mappings: Scalar Case \*

Stephen P. Luttrell

*Pattern Processing Principles Section. Royal Signals and Radar Establishment,  
St. Andrews Rd., Malvern, WORCS, WR14 3PS, United Kingdom.*

In coding theory one transforms signals from a source representation into an encoded representation that is suitable for transmission through a (possibly noisy) medium, and upon reception one decodes to reconstruct an approximation to the original signal. In autoassociative network theory one reconstructs a signal given incomplete (and possibly noisy) information about it. We derive some new results by combining these two approaches in the form of vector quantisation (VQ) theory and topographic mapping (TM) theory. We use a VQ model (with a noisy transmission medium) to model the processes that occur in TMs, which leads to the standard TM training algorithm, albeit with a slight modification to the encoding process (minimum distortion rather than nearest neighbour encoding). To emphasise this difference we call our model a topographic vector quantiser (TVQ). In the continuum limit of the one-dimensional (scalar) TVQ we find that the density of code vectors is proportional to  $P(x)^\alpha$  ( $\alpha = \frac{1}{3}$ ) (which is the same as the result obtained from a standard scalar quantiser), assuming that the transmission medium introduces additive noise with a zero-mean, symmetric, monotonically decreasing probability density (which is equivalent to using a symmetrically tapered neighbourhood in a TM). Our  $\alpha = \frac{1}{3}$  result is dramatically different from the  $\alpha = \frac{1}{3} \frac{(2n+1)^2}{(n+1)^2+n^2}$  result that is predicted when the standard TM training algorithm is used with a uniform symmetric neighbourhood  $[-n, +n]$ , and we note that this difference arises entirely from using minimum distortion rather than nearest neighbour encoding. We verify our new result by performing a numerical experiment using  $P(x) \propto x$ .

## I. INTRODUCTION

This paper is concerned with the overlap between two related subjects. On the one hand there is much literature on vector quantisation (VQ) and scalar quantisation theory [3, 4, 8], where an encoding/decoding scheme is optimised in such a way as to minimise a distortion measure. On the other hand there is an interesting class of transformations (called topographic mappings (TM) in the neural network literature [2]) that can be trained to perform optimal mappings of high-dimensional input vectors into low-dimensional output vectors.

In [7] and [5] we unified these two subjects by formulating the problem of training a TM in terms of minimising a distortion measure. This required a slight modification of the original TM training algorithm, but the observed side effects of this were minimal. We call this type of mapping a topographic vector quantiser (TVQ) in order to emphasize its close relationship to a standard VQ, and to distinguish it from the TM method. A TVQ has the important property that it encodes information in such a way that it is robust with respect to the damaging effects of a noise process which makes transitions between code indices which lie within the same topographic neighbourhood. A good example of the use of this approach in designing robust codes can be found in [1].

The question of the asymptotic properties of TVQs

(versus those of VQs) naturally arises. In a recent study [9] the asymptotic code vector (CV) density in a scalar TM (one dimension mapped to one dimension) was derived and was found to be proportional to  $P(x)^\alpha$ , where  $\alpha = \frac{1}{3} \frac{(2n+1)^2}{(n+1)^2+n^2}$ ,  $P(x)$  being the probability density of input scalars and  $n$  the half-width of the update neighbourhood used when training the TM. The  $n = 0$  case reduces to  $\alpha = \frac{1}{3}$ , which is consistent with the result that is expected from a scalar quantiser [4].

We wish to derive the corresponding TVQ result for the case where we solve a minimum  $L_2$  distortion problem (as formulated in [7] and [5]), rather than the standard TM problem (as formulated in [2]). For simplicity, we shall restrict our attention in this paper to the scalar case, so our results become an extension of the results that were reported in [9].

In Section II we use a uniform topographic neighbourhood function to derive the asymptotic CV density in a scalar TVQ (this directly corresponds to the scalar TM derivation in [9]). In Section III we extend this result to the more general case of a tapered (i.e., symmetric and monotonically decreasing to zero) topographic neighbourhood function. In Section IV we present the results of some numerical simulations that verify the theoretical results in Section III. In the Appendix we briefly explain the origin of and cure for a stability problem that can arise when training TVQs.

It is important to note that although our derivations refer to the case of one input dimension (i.e. the scalar case), we frequently use the word “vector” when referring to the input. Although this might seem bizarre, it reduces the amount of additional nomenclature that we have to introduce. Thus we speak of a “topographic vec-

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 11, 2019.

This paper appeared in IEEE Trans. Neural Networks, 1991, vol. 2, no. 4, pp. 427-436. Manuscript received August 20, 1990; revised March 14, 1991. © 1991 British Crown Copyright.

tor quantiser" when we should strictly say "topographic scalar quantiser", and similarly "code vector" instead of "code scalar".

## II. DENSITY WITH A UNIFORM TOPOGRAPHIC NEIGHBOURHOOD FUNCTION

In this section we derive conditions for the stationarity (with respect to variation of the encoding/decoding functions) of the distortion caused by a TVQ. We use these results to derive the asymptotic CV density in the case of a uniform topographic neighbourhood function (as used in [9]).

### A. Euclidean distortion

Introduce an  $L_2$  distortion measure  $D_1$  as

$$\begin{aligned} D_1 &= \int dx P(x) (x - x'_{y(x)})^2 \\ &= \sum_y \int_{q_{y-1}}^{q_y} dx P(x) (x - x'_{y'})^2 \end{aligned} \quad (2.1)$$

In Figure 1 we show as a network the various steps that are involved in calculating  $D_1$ : the figure reads from the bottom to the top. The input  $x$  is a scalar which we represent by the horizontal axis at the bottom of Figure 1. An encoding function,  $y(x)$ , maps from the input  $x$  to a code index  $y$ . The interval  $[q_{y-1}, q_y]$  depends on  $y(x)$  as follows:

$$[q_{y-1}, q_y] = \{x : y = y(x)\} \quad (2.2)$$

which we use in Equation 2.1 to partition the range of integration into a set of convenient intervals. The output  $x'_{y'}$  is the decoding function (i.e., the CV) associated with code index  $y$ , and it sits on the horizontal axis  $x'_{y'}$  that we have drawn at the top of Figure 1. The set of  $x'_{y'}$  (ranging over all values of the index  $y$ ) comprises the codebook that is used in this encoding/decoding operation. The overall goal is to choose the encoding  $y(x)$  and decoding  $x'_{y'}$  functions in such a way as to minimise the mean  $L_2$  distortion  $D_1$  between input  $x$  and output  $x'_{y(x)}$ . Note that we have simplified Figure 1 by spacing the CVs at constant intervals for illustrative purposes.

### B. Minimum Euclidean distortion

In order to minimise  $D_1$ , we must differentiate it with respect to the various free parameters: in this case the  $q_y$  (which parameterise the encoding function) and the  $x'_{y'}$  (which parameterise the decoding function  $y(x)$ , see

Equation 2.2). Thus we obtain the partial derivatives as

$$\begin{aligned} \frac{\partial D_1}{\partial q_y} &= P(q_y) \left[ (q_y - x'_{y'})^2 - (q_y - x'_{y+1})^2 \right] \\ &= 2P(q_y) (x'_{y+1} - x'_{y'}) \left( q_y - \frac{x'_{y'} + x'_{y+1}}{2} \right) \end{aligned} \quad (2.3)$$

$$\frac{\partial D_1}{\partial x'_{y'}} = -2 \int_{q_{y-1}}^{q_y} dx P(x) (x - x'_{y'}) \quad (2.4)$$

whence the stationary points of  $D_1$  must satisfy

$$q_y = \frac{x'_{y'} + x'_{y+1}}{2} \quad (2.5)$$

$$x'_{y'} = \frac{\int_{q_{y-1}}^{q_y} dx P(x) x}{\int_{q_{y-1}}^{q_y} dx P(x)} \quad (2.6)$$

Note that Equation 2.5 requires that  $q_y$  lie midway between the adjacent CVs, so it defines a nearest neighbour encoding function  $y(x)$ .

### C. Topographic Euclidean distortion

Now we generalise the  $L_2$  distortion measure of Equation 2.1 to include a neighbourhood function  $\pi_{y',y}$  that specifies the extent to which  $y'$  is in the neighbourhood of  $y$  (later on we define this notion more precisely). Thus introduce the  $L_2$  distortion measure  $D_2$  as

$$\begin{aligned} D_2 &= \int dx P(x) \sum_{y'} \pi_{y',y(x)} (x - x'_{y'})^2 \\ &= \sum_y \int_{q_{y-1}}^{q_y} dx P(x) \sum_{y'} \pi_{y',y} (x - x'_{y'})^2 \end{aligned} \quad (2.7)$$

The  $y(x)$  (and hence the  $q_y$ ) and the  $x'_{y'}$  used in  $D_2$  are to be understood to be different from those used in  $D_1$ . In Figure 2 we show a modified version of Figure 1 in which the effect of the  $\pi_{y',y}$  is represented (for simplicity, we show only  $\pi_{y\pm 1,y}$ ). The action of the  $\pi_{y',y}$  is interposed between the action of  $y(x)$  and the action of  $x'_{y'}$  and it can be interpreted as the relative probability with which index  $y$  is corrupted by some distortion process to become index  $y'$ . With this interpretation in mind, it is evident that minimising  $D_2$  with respect to the choice of  $y(x)$  and  $x'_{y'}$  will cause the encoding/decoding process to become robust with respect to the damaging effects of the distortion process, modelled by [5, 7].

### D. Minimum topographic Euclidean distortion

We may now repeat the derivation of Section II B for  $D_2$  (rather than  $D_1$ ). The partial derivatives are

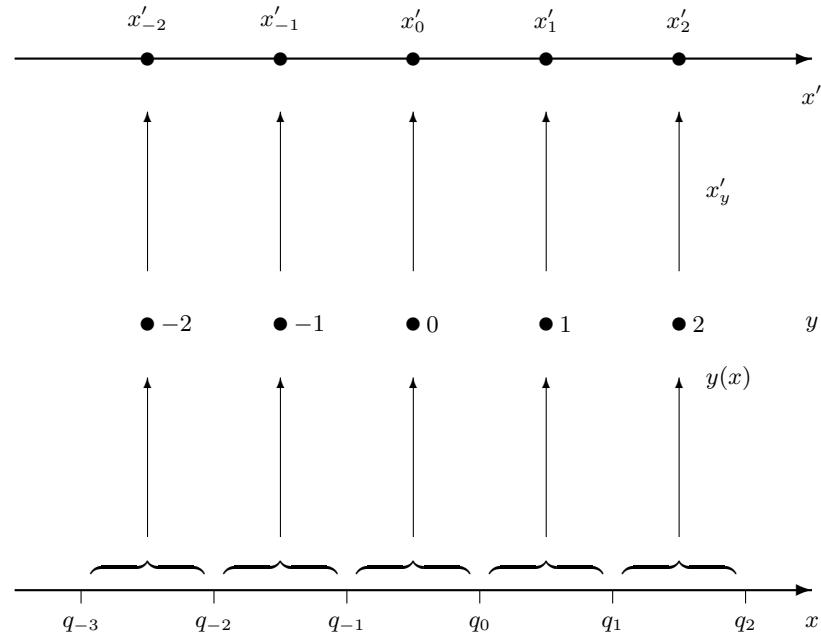


Figure 1: Network representation of encoding an input  $x$  to produce a code  $y$ , followed by decoding to produce a reconstruction  $x'$ .

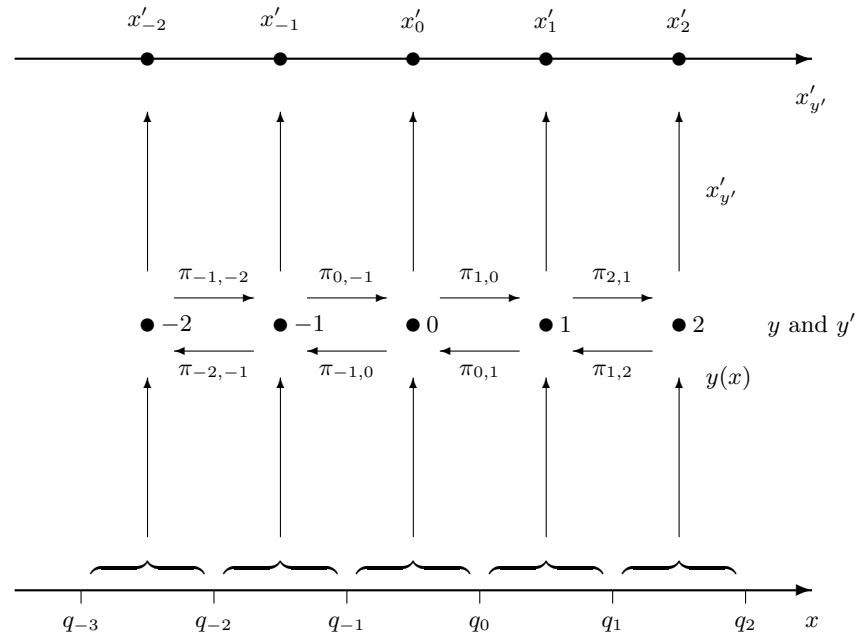


Figure 2: Network representation of encoding an input  $x$  to produce a code  $y$ , followed by corruption of the code by a transition matrix  $\pi$  and then decoding to produce a reconstruction  $x'$ .

$$\frac{\partial D_2}{\partial q_y} = P(q_y) \left[ \sum_{y'} \pi_{y',y} (q_y - x'_{y'})^2 - \sum_{y'} \pi_{y',y+1} (q_y - x'_{y'})^2 \right] \quad (2.8)$$

$$\frac{\partial D_2}{\partial x'_y} = -2 \sum_{y'} \int_{q_{y'-1}}^{q_{y'}} dx P(x) \pi_{y,y'} (x - x'_{y'}) \quad (2.9)$$

so the stationary points of  $D_2$  must satisfy

$$\sum_{y'} \pi_{y',y} (q_y - x'_{y'})^2 = \sum_{y'} \pi_{y',y+1} (q_y - x'_{y'})^2 \quad (2.10)$$

$$x'_{y'} = \frac{\sum_{y'} \int_{q_{y'-1}}^{q_{y'}} dx P(x) \pi_{y,y'} x}{\sum_{y'} \int_{q_{y'-1}}^{q_{y'}} dx P(x) \pi_{y,y'}} \quad (2.11)$$


---

In general, when  $\pi_{y',y} \neq \delta_{y',y}$  in Equation 2.10 we cannot recover the nearest neighbour encoding property that we obtained in Equation 2.5.

In practical applications we have found that the modification of Equation 2.5 to become Equation 2.10 only occasionally produces a different encoding of the input, but these small differences between encoding schemes lead to large differences between the asymptotic CV density in the TVQ (minimum distortion encoding) and TM (nearest neighbour encoding) cases.

### E. Finite differences and derivatives

In this subsection we bring together various useful results that we need in order to derive the asymptotic CV density. In order to make direct contact with the results that were reported in [9] we now assume a specific form for  $\pi_{y',y}$

$$\pi_{y',y} = \begin{cases} 1 & \text{if } |y' - y| \leq n \\ 0 & \text{if } |y' - y| > n \end{cases} \quad (2.12)$$

This  $\pi_{y',y}$  defines a uniform neighbourhood that ranges from  $y - n$  to  $y + n$  in the neighbourhood of code index  $y$ , which we call a  $[-n, +n]$  neighbourhood.

With this assumption we can solve Equation 2.10 to yield

$$q_y = \frac{1}{2} (x'_{y-n} + x'_{y+n+1}) \quad (2.13)$$

which should be compared with the result in Equation 2.5 (which corresponds to making the choice  $\pi_{y',y} = \delta_{y',y}$ , or  $n = 0$ , in Equation 2.13). The effect of the  $[-n, +n]$  neighbourhood is to replace the midpoint of the interval  $[x'_y, x'_{y+1}]$  by the midpoint of the larger interval  $[x'_{y-n}, x'_{y+n+1}]$ . Note that  $x'_y \leq q_y \leq x'_{y+1}$  is not necessarily true when  $n > 0$ .

Now introduce a pair of expressions to relate the finite differences of  $x'_y$  to the derivatives  $\frac{dx'_y}{dy}$  and  $\frac{d^2x'_y}{dy^2}$  of  $x'_y$ :

$$\begin{aligned} x'_{y+k} - x'_{y-k} &= 2k \frac{dx'_y}{dy} + \mathcal{O}\left(k^3 \frac{d^3x'_y}{dy^3}\right) \\ x'_{y+k} + x'_{y-k} - 2x'_y &= k^2 \frac{d^2x'_y}{dy^2} + \mathcal{O}\left(k^4 \frac{d^4x'_y}{dy^4}\right) \end{aligned} \quad (2.14)$$

These two expressions can easily be obtained by Taylor expanding (about the point where the code index has value  $y$ ) the various terms on the left-hand sides of Equation 2.14.

Using Equation 2.13 and Equation 2.14 we may derive the midpoint  $u_y$  and the half-length  $a_y$  of the interval  $[q_{y-n-1}, q_{y+n}]$  as

$$\begin{aligned} u_y &\equiv \frac{1}{2} (q_{y-n-1} + q_{y+n}) \\ &= \frac{1}{4} (x'_{y-2n-1} + x'_{y+2n+1} + 2x'_y) \\ &\simeq x'_y + \frac{(2n+1)^2}{4} \frac{d^2x'_y}{dy^2} \end{aligned} \quad (2.15)$$

$$\begin{aligned} a_y &\equiv \frac{1}{2} (q_{y+n} - q_{y-n-1}) \\ &= \frac{1}{4} (x'_{y+2n+1} - x'_{y-2n-1}) \\ &\simeq \frac{2n+1}{2} \frac{dx'_y}{dy} \end{aligned} \quad (2.16)$$

We now wish to transform our results into the language of density of CVs,  $\rho(x'_y)$ . We can relate  $\rho(x'_y)$  to quantities that we have already introduced as follows:

$$\frac{2n+1}{2} \frac{dx'_y}{dy} \rho(x'_y) = \frac{dy}{dx'_y} \quad (2.17)$$

Thus  $\rho(x'_y)$  is the number of CV indices  $y$  per unit change in CV position  $x'_y$ . In Equation 2.14 we encountered derivatives of  $x'_y$  which we now express in terms of

derivatives of  $\rho(x'_y)$ . Thus,

$$\begin{aligned}\frac{dx'_y}{dy} &= \frac{1}{\rho(x'_y)} \\ \frac{d^2x'_y}{dy^2} &= -\frac{1}{\rho(x'_y)^3} \frac{d\rho(x'_y)}{dy}\end{aligned}\quad (2.18)$$

where we have used  $\frac{d}{dy} = \frac{dx'_y}{dy} \frac{d}{dx'_y}$  to perform the differentiation. We may thus express the results for  $u_y$  (Equation 2.15) and  $a_y$  (Equation 2.16) in terms of derivatives of  $\rho(x'_y)$  as follows:

$$u_y \simeq x'_y - \frac{(2n+1)^2}{4\rho(x'_y)^3} \frac{d\rho(x'_y)}{dy} \quad (2.19)$$

$$a_y \simeq \frac{2n+1}{2\rho(x'_y)} \quad (2.20)$$

#### F. Approximate optimal code vector positions

We now have all the basic theoretical results that are needed to perform a Taylor expansion of Equation 2.11 to relate the derivative of  $P(x)$  to the derivative of  $\rho(x)$ . Insert the  $\pi_{y',y}$  (defined in Equation 2.12), and use the definitions of the midpoint  $u_y$  and half-length  $a_y$  of the interval  $[q_{y-n-1}, q_{y+n}]$  (in Equation 2.15 and Equation 2.16, respectively) in Equation 2.11 to obtain

$$\begin{aligned}x'_y &= \frac{\int_{-a_y}^{+a_y} dx P(x) (u_y + x)}{\int_{-a_y}^{+a_y} dx P(x)} \\ &= u_y + \frac{\int_{-a_y}^{+a_y} dx \sum_{m=0}^{\infty} \frac{P^{(m)}(u_y) x^{m+1}}{m!}}{\int_{-a_y}^{+a_y} dx \sum_{m=0}^{\infty} \frac{P^{(m)}(u_y) x^m}{m!}} \\ &= u_y + \sum_{m \geq 1 \text{ (odd)}} \frac{\frac{P^{(m)}(u_y) a_y^{m+2}}{m! (m+2)}}{\sum_{m \geq 0 \text{ (even)}} \frac{P^{(m)}(u_y) a_y^{m+1}}{(m+1)!}} \\ &= u_y + \frac{\sum_{m \geq 1 \text{ (odd)}} \frac{P^{(m)}(u_y) a_y^{m+1}}{P^{(0)}(u_y) m! (m+2)}}{1 + \sum_{m \geq 2 \text{ (even)}} \frac{P^{(m)}(u_y) a_y^m}{P^{(0)}(u_y) (m+1)!}} \\ &= u_y + a_y \left( \frac{a_y P^{(1)}(u_y)}{3P^{(0)}(u_y)} + \mathcal{O} \left( \frac{a_y^3 P^{(3)}(u_y)}{P^{(0)}(u_y)} \right) + \text{h.o.t.} \right) \left( 1 + \mathcal{O} \left( \frac{a_y^2 P^{(2)}(u_y)}{P^{(0)}(u_y)} \right) + \text{h.o.t.} \right) \\ &= u_y + \frac{a_y^2}{3P(u_y)} \frac{dP(u_y)}{du_y} + \mathcal{O} \left( \frac{a_y^3 P^{(2)}(u_y)}{P^{(0)}(u_y)} \right) + \text{h.o.t.} \\ &= u_y + \frac{a_y^2}{3P(x'_y)} \frac{dP(x'_y)}{dx'_y} + \mathcal{O} \left( \frac{a_y^3 P^{(2)}(x'_y)}{P^{(0)}(x'_y)} \right) + \text{h.o.t.}\end{aligned}\quad (2.21)$$


---

Throughout this derivation we use the notation  $P^{(n)}(u_y)$  to denote the  $n$ th derivative of  $P(u_y)$  with respect to  $u_y$  (and similarly  $P^{(n)}(x'_y)$ ). We assume the dimensionless inequality  $a_y^n \frac{P^{(n)}(u_y)}{P^{(0)}(u_y)} \ll 1$  for  $n \geq 1$ , and we assume that the  $n \geq 3$  terms are negligible in comparison with the leading terms. We can therefore discard terms with squares (and higher powers) of  $a_y^n \frac{P^{(n)}(u_y)}{P^{(0)}(u_y)}$ . In the last

stage of this derivation note that the effect of the change  $u_y \rightarrow x'_y$  appears only in the next to leading order terms.

### G. Code vector density

Finally, inserting the expressions for  $u_y$  and  $a_y$  (from Equation 2.19 and Equation 2.20) into Equation 2.21, we obtain

$$\frac{1}{\rho(x'_y)} \frac{d\rho(x'_y)}{dy} = \frac{1}{3P(x'_y)} \frac{dP(x'_y)}{dy} \quad (2.22)$$

We may solve this to yield asymptotically

$$\rho(x'_y) = P(x'_y)^{1/3} \quad (2.23)$$

This power law dependence is the same as that observed in the equivalent scalar quantiser (which corresponds to a neighbourhood function  $\pi_{y',y} = \delta_{y',y}$ ) but is different from the result that was obtained in [9] for a TM as defined in [2]. These differences arise because we use minimum distortion encoding within the context of our TVQ model, rather than nearest neighbour encoding within the context of the TM model.

### III. DENSITY WITH A TAPERED TOPOGRAPHIC NEIGHBOURHOOD FUNCTION

In this section we extend the results of Section II to the case where  $\pi_{y',y}$  defines a tapered topographic neighbourhood function. Specifically, we shall use a symmetric function that monotonically decreases to zero.

#### A. Topographic Euclidean distortion

We now define the distortion matrix  $\pi_{y',y}$  (used in the definition of the  $L_2$  distortion  $D_2$  in Equation 2.7) in such a way that it satisfies

$$\begin{aligned} \pi_{y'+1,y+1} &= \pi_{y',y} && \text{(Toeplitz matrix)} \\ \pi_{y',y} &= \pi_{y,y'} && \text{(symmetric matrix)} \end{aligned} \quad (3.1)$$

For such matrices it is sufficient to specify the form of a single row (or column) of  $\pi_{y',y}$  as a symmetric function of  $y' - y$ . This type of matrix specifies a distortion that treats each code index  $y$  on an equal footing (the Toeplitz property), and it implies a symmetric topographic neighbourhood (the symmetric property).

For convenience, and to make contact with the derivation presented in Section II, we decompose  $\pi_{y',y}$  as a weighted sum over (symmetric) neighbourhood functions of the type defined in Equation 2.12:

$$\pi_{y',y} = \sum_{s:|y'-y|\leq n_s} h_s \quad (3.2)$$

As discussed in the Appendix, we should ensure that  $\pi_{y',y}$  is a monotonically decreasing function of  $y' - y$  by setting  $h_s > 0$ . In Figure 3 we give an example of the type

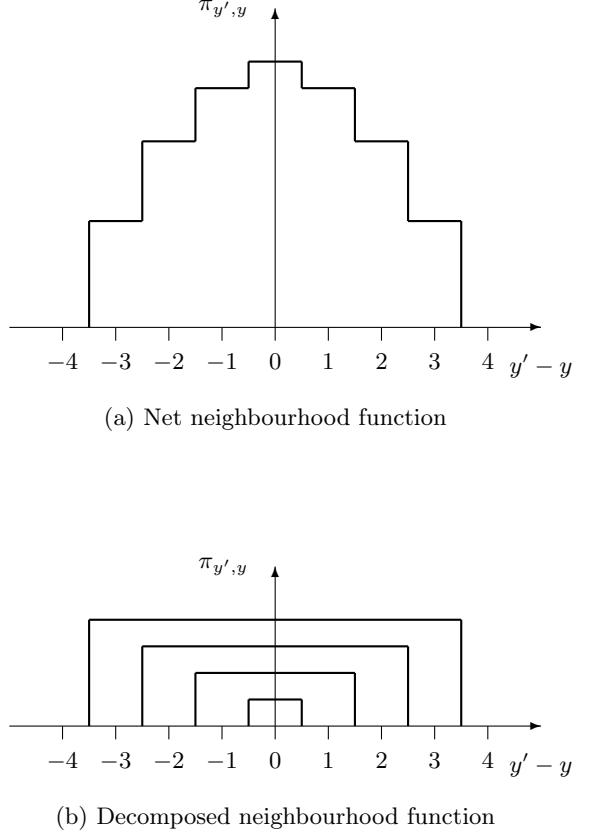


Figure 3: An example of the decomposition of a symmetrically tapered topographic neighbourhood function into a sum of rectangular pieces.

of tapered neighbourhood function that is described by this model. We show in Figure 3(a) a typical  $\pi_{y',y}$  neighbourhood function and in Figure 3(b) its decomposition as a sum over  $[-n_s, +n_s]$  neighbourhoods for various  $s$ .

Using the definition of  $\pi_{y',y}$  in Equation 3.2 we can simplify  $D_2$  in Equation 2.7 to become  $D_3$  given by

$$D_3 = \sum_y \int_{q_{y-1}}^{q_y} dx P(x) \sum_s h_s \sum_{y'=-n_s}^{+n_s} (x - x'_{y+y'})^2 \quad (3.3)$$

#### B. Minimum topographic Euclidean distortion

Now differentiate  $D_3$  to obtain  $\frac{\partial D_3}{\partial x'_{y,y}}$  and  $\frac{\partial D_3}{\partial q_y}$ . The stationary points of  $D_3$  must then satisfy

$$q_y = \frac{\sum_s h_s (x'_{y+n_s+1} - x'_{y-n_s}) (x'_{y+n_s+1} + x'_{y-n_s})}{2 \sum_s h_s (x'_{y+n_s+1} - x'_{y-n_s})} \quad (3.4)$$

$$x'_{y,y} = \frac{\sum_s h_s \int_{q_{y-n_s-1}}^{q_{y+n_s}} dx P(x) x}{\sum_s h_s \int_{q_{y-n_s-1}}^{q_{y+n_s}} dx P(x)} \quad (3.5)$$

Equation 3.4 replaces Equation 2.13, and Equation 3.5 replaces Equation 2.11.

Note that the positivity of the  $h_s$  in Equation 3.2 guarantees the stability of the solution for the  $q_y$  and the  $x'_y$  in Equation 3.4 and Equation 3.5, because the denominators are strictly positive. Note that we assume that the input probability density  $P(x)$  is well behaved in the sense that each code index  $y$  is indeed associated with a finite probability.

Unfortunately, the expression for  $q_y$  in Equation 3.4 is

sufficiently complicated that we have to perform a large amount of algebra to derive the asymptotic relationship between  $P(x)$  and  $\rho(x)$  (i.e., the generalisation of Equation 2.23).

### C. Finite differences and derivatives

Firstly, express  $x'_{y+k} \pm x'_{y+l}$  in terms of  $\frac{dx'_y}{dy}$  and  $\frac{d^2x'_y}{dy^2}$ :

$$\begin{aligned} x'_{y+k} \pm x'_{y+l} &= \frac{1}{2} (x'_{y+k} + x'_{y-k} - 2x'_y) + \frac{1}{2} (x'_{y+k} - x'_{y-k}) \\ &\quad \pm \frac{1}{2} (x'_{y+l} + x'_{y-l} - 2x'_y) \pm \frac{1}{2} (x'_{y+l} - x'_{y-l}) \\ &\quad + x'_y \pm x'_y \\ &\approx \left( \frac{k^2 \pm l^2}{2} \right) \frac{d^2x'_y}{dy^2} + (k \pm l) \frac{dx'_y}{dy} + (x'_y \pm x'_y) \end{aligned} \quad (3.6)$$

where we have used the finite difference expressions in Equation 2.14. Note that we use  $\pm$  signs consistently throughout Equation 3.6 such that, if one were to choose the upper sign in one part of the equation, then one must choose the upper sign throughout the rest of the equation (a similar remark applies to the lower sign). We may use these results to simplify  $q_{y+n_s}$  and  $q_{y-n_s-1}$  to obtain

$$\begin{aligned} q_{y+n_s} &= \frac{\sum_t h_t (\xi_1(t) + \xi_2(s, t)) (\eta_0 + \eta_1(s) + \eta_2(s, t))}{2 \sum_t h_t (\xi_1(t) + \xi_2(s, t))} \\ q_{y-n_s-1} &= \frac{\sum_t h_t (\xi_1(t) - \xi_2(s, t)) (\eta_0 - \eta_1(s) - \eta_2(s, t))}{2 \sum_t h_t (\xi_1(t) - \xi_2(s, t))} \end{aligned} \quad (3.7)$$

where we have defined  $\xi_1(t)$ ,  $\xi_2(s, t)$ ,  $\eta_0$ ,  $\eta_1(s)$  and  $\eta_2(s, t)$  as

$$\begin{aligned} \xi_1(t) &\equiv (2n_t + 1) \frac{dx'_y}{dy} \\ \xi_2(s, t) &\equiv \frac{(n_s + n_t + 1)^2 - (n_s - n_t)^2}{2} \frac{d^2x'_y}{dy^2} \\ \eta_0 &\equiv 2x'_y \\ \eta_1(s) &\equiv (2n_s + 1) \frac{dx'_y}{dy} \\ \eta_2(s, t) &\equiv \frac{(n_s + n_t + 1)^2 + (n_s - n_t)^2}{2} \frac{d^2x'_y}{dy^2} \end{aligned} \quad (3.8)$$

Note that  $\xi_1(t)$  and  $\eta_1(s)$  are trivially related to each other.

We may now introduce the midpoint  $u_y^s$  and half-length  $a_y^s$  of the interval  $[q_{y-n_s-1}, q_{y+n_s}]$  and use Equation 3.7 to simplify their expressions. For compactness, we gather these two results into a single vector equation (where  $u_y^s$  is the upper element and  $a_y^s$  the lower element in a two-component column vector):

$$\begin{aligned}
\begin{pmatrix} u_y^s \\ a_y^s \end{pmatrix} &\equiv \frac{1}{2} \begin{pmatrix} q_{y+n_s} + q_{y-n_s-1} \\ q_{y+n_s} - q_{y-n_s-1} \end{pmatrix} \\
&= \frac{\sum_{t,t'} h_t h_{t'} \left[ (\eta_0 + \eta_2(s, t)) \begin{pmatrix} \xi_1(t) \xi_1(t') - \xi_2(s, t) \xi_2(s, t') \\ \xi_1(t') \xi_2(s, t) - \xi_1(t) \xi_2(s, t') \end{pmatrix} \right.}{2 \sum_{t,t'} h_t h_{t'} (\xi_1(t) \xi_1(t') - \xi_2(s, t) \xi_2(s, t'))} \\
&\quad \left. + \eta_1(s) \begin{pmatrix} \xi_1(t') \xi_2(s, t) - \xi_1(t) \xi_2(s, t') \\ \xi_1(t) \xi_1(t') - \xi_2(s, t) \xi_2(s, t') \end{pmatrix} \right] \tag{3.9}
\end{aligned}$$


---

We have made use of the fact that  $\sum_{t,t'} h_t h_{t'} (\xi_1(t') \xi_2(s, t) - \xi_1(t) \xi_2(s, t')) = 0$  (by symmetry) to simplify the denominator. Now introduce some approximations which are valid in the leading order of the expansion in terms of derivatives of  $x'_y$  with respect to  $y$  (see Equation 3.8):

$$\begin{aligned}
\xi_2(s, t) \xi_2(s, t') &\simeq 0 \\
\xi_2(s, t) (\xi_1(t') \xi_2(s, t) - \xi_1(t) \xi_2(s, t')) &\simeq 0 \tag{3.10}
\end{aligned}$$


---

When we insert these approximations into Equation 3.9 and make use of the relationships in Equation 2.18, we obtain

$$\begin{aligned}
u_y^s &\simeq \frac{\sum_{t,t'} h_t h_{t'} \xi_1(t) \xi_1(t') (\eta_0 + \eta_2(s, t))}{2 \sum_{t,t'} h_t h_{t'} \xi_1(t) \xi_1(t')} \\
&= x'_y + \left[ \frac{\sum_t h_t (2n_t + 1) [(n_s + n_t + 1)^2 + (n_s - n_t)^2]}{4 \sum_t h_t (2n_t + 1)} \right] \frac{d^2 x'_y}{dy^2} \\
&\simeq x'_y - \left[ \frac{(2n_s + 1)^2}{4} + \frac{\sum_t h_t (2n_t + 1) (n_t - n_s) (n_s + n_t + 1)}{2 \sum_t h_t (2n_t + 1)} \right] \frac{1}{\rho(x'_y)^3} \frac{d\rho(x'_y)}{dx'_y} \tag{3.11}
\end{aligned}$$

$$\begin{aligned}
a_y^s &\simeq \frac{\sum_{t,t'} h_t h_{t'} \xi_1(t) \xi_1(t') \eta_1(s)}{2 \sum_{t,t'} h_t h_{t'} \xi_1(t) \xi_1(t')} \\
&\simeq \frac{2n_s + 1}{2\rho(x'_y)} \tag{3.12}
\end{aligned}$$


---

We have expressed these results in such a way that they may readily be compared with the analogous results in Equation 2.19 and Equation 2.20.

When comparing Equation 3.11 with Equation 2.19, note that there is a leading order correction term caused by the presence of more than one component in the tapered neighbourhood function, but note that Equation 2.20 needs no such leading order correction to become Equation 3.12.

#### D. Approximate optimal code vector positions

We now form a Taylor expansion of the integrands in the numerator and denominator of Equation 3.5. The steps in the derivation are analogous to those used to derive the result in Equation 2.21 so we present only the final result, which is

$$x'_y \simeq \frac{\langle a_y^s u_y^s \rangle}{\langle a_y^s \rangle} + \frac{\langle (a_y^s)^3 \rangle}{3 \langle a_y^s \rangle P(x'_y)} \frac{dP(x'_y)}{dx'_y} \tag{3.13}$$

where our angle bracket notation represents an average weighted by the  $h_s$ , which is defined as

$$\langle \dots \rangle \equiv \frac{\sum_s h_s (\dots)}{\sum_s h_s} \quad (3.14)$$

The ratios of weighted averages in Equation 3.13 may be evaluated by taking appropriate averages of the results in Equation 3.11 and Equation 3.12, to yield

$$\begin{aligned} \frac{\langle a_y^s u_y^s \rangle}{\langle a_y^s \rangle} &= x'_y - \frac{\langle (2n_s + 1)^3 \rangle}{4 \langle 2n_s + 1 \rangle \rho(x'_y)^3} \frac{d\rho(x'_y)}{dx'_y} \\ \frac{\langle (a_y^s)^3 \rangle}{\langle a_y^s \rangle} &= \frac{\langle (2n_s + 1)^3 \rangle}{4 \langle 2n_s + 1 \rangle \rho(x'_y)^2} \end{aligned} \quad (3.15)$$

where we note that the contribution of the correction term in Equation 3.11 disappears (by symmetry).

### E. Code vector density

Finally, inserting the results of Equation 3.15 into the leading order Taylor expansion in Equation 3.13 yields (in leading order) the same differential equation that we obtained in Equation 2.22. Thus we have shown that for the class of  $\pi_{y',y}$  corresponding to symmetric monotonically decreasing neighbourhood functions the asymptotic CV density is given by  $\rho(x'_y) = P(x'_y)^{1/3}$  (i.e., the same as Equation 2.23).

## IV. NUMERICAL SIMULATION

In this section we shall present the results of a simple numerical simulation that verifies our theoretical prediction  $\rho(x'_y) = P(x'_y)^{1/3}$  in the one-dimensional case.

### A. Numerical experimental procedure

We now describe a variant of the numerical experiment that was performed in [9].

$$x'_y^{(\text{new})} = x'_y^{(\text{old})} + 0.1 \left( x - x'_y^{(\text{old})} \right) \quad |y - y(x)| \leq n \quad (4.1)$$

1. Define a finite support for  $x : x \in [0, 1]$ .
2. Define a probability density  $P(x)$  of input scalars:  $P(x) = 2x$  (i.e., a ramp).
3. Choose the number  $n_{cv}$ , of CVs (in this case, code scalars) that you wish to use. We use  $n_{cv} = 30$  because we find that this is large enough for the results of the numerical experiment to approximate those that we would expect in the asymptotic (i.e.,  $n_{cv} \rightarrow \infty$ ) case.

4. Choose the number  $n$  that determines the size of the  $[-n, +n]$  neighbourhood, in the form given in Equation 2.12.

5. Adapt the positions of the CVs using the standard training scheme for TMs [2] with nearest neighbour encoding. In Equation 4.1 we use an update step size  $\epsilon = 0.1$ , and we train using 500 000 inputs  $x$  that are sampled independently from  $P(x)$ . It is not clear that this guarantees complete convergence, but the training schedule is good enough to demonstrate the point that we wish to make.

6. Break up the  $[0, 1]$  interval into histogram bins, each of which covers a small interval  $[b - \frac{\Delta}{2}, b + \frac{\Delta}{2}]$  of width  $\Delta$  centred at  $x = b$ . We use ten bins, so  $\Delta = 0.1$  and  $b = 0.05, 0.15, \dots, 0.85, 0.95$ . These bins are used to estimate the relative frequency with which the CVs land in each of the ten intervals.

In our experiments we increment the bins after every  $n_{cv}$  ( $= 30$  in our experiments) training samples. Each bin thus contains a cumulative count of the number of CVs that have appeared in it (summed over all of the “snapshots” taken at intervals of  $n_{cv}$  samples).

This procedure for incrementing the histogram has an infinitely long memory time, so the final histogram (after 500 000 samples) will be the mixture of all histograms that occurred as training progressed toward convergence. This is obviously undesirable, and could be cured by imposing a finite memory by, for instance, making the histogram bins “leaky”. We do not implement such refinements.

7. Do a least squares fit of  $\rho(x)$  versus  $P(x)$  as follows:
  - (a) For histogram bin  $[b - \frac{\Delta}{2}, b + \frac{\Delta}{2}]$  determine two quantities:
    - i. The probability  $P_i$  that  $P(x)$  generates a point lying in bin  $i$ : this can be calculated to be  $2b_i\Delta$ .
    - ii. The probability  $\rho_i$  that a CV lies in bin  $i$ : this is estimated from the outcome of the numerical experiment as the proportion of CVs that land in bin  $i$ .
  - (b) Plot the  $(P_i, \rho_i)$  coordinates from the previous step on a  $(P, \rho)$  graph ( $P$  being the abscissa and  $\rho$  the ordinate).
  - (c) Define a parametric model  $\rho(P) = A P^\alpha$ .
  - (d) Find the  $A$  and  $\alpha$  that minimise the sum of squared errors  $\sum_i (\rho(P_i) - \rho_i)^2$ . Because we use a rather small value of  $n_{cv}$  we find that edge effects (near  $x = 0$  and  $x = 1$ ) adversely affect the number of counts in the  $b = 0.05$  (leftmost) and  $b = 0.95$  (rightmost) histogram

bins. We therefore discard these two bins and use only the central eight (out of ten) bins to estimate  $A$  and  $\alpha$ .

Note that in the fitting process we do not do a least squares fit to a logarithmic plot, because it is the estimated  $\rho_i$  (not the estimated  $\log \rho_i$ ) that has approximately Gaussian errors.

Although our numerical experimental procedure is somewhat cruder than in [9], we nevertheless found that by using the standard TM update procedure we could reproduce the numerical results that are quoted in [9].

We then made two changes to the numerical experiment in order to simulate a TVQ rather than a TM. First, we generalised step 5 of the numerical experiment to permit either minimum distortion encoding or nearest neighbour encoding to be used. Second, we introduced a tapered symmetric neighbourhood of the type shown in Figure 3. In our simulations we used two components: a  $[0, 0]$  neighbourhood term, plus various types of  $[-1, +1]$  neighbourhood terms. This leads to the update equation

$$\begin{aligned} x_{y(x)}'(\text{new}) &= x_{y(x)}'(\text{old}) + 0.1 \left( x - x_{y(x)}'(\text{old}) \right) \\ x_{y(x)\pm 1}'(\text{new}) &= x_{y(x)\pm 1}'(\text{old}) + \epsilon' (x - x_{y(x)\pm 1}'(\text{old})) \end{aligned} \quad (4.2)$$

where one must chose signs consistently, and  $\epsilon'$  takes one of three values  $\epsilon' = 0.025, 0.050, 0.075$ .

## B. Numerical experimental results

The final estimated values of  $\alpha$  (i.e., after 500 000 updates) are shown in Table I. These are the results of a single run, so we have not attempted to quote a standard error. However, the TM case consistently produces an  $\alpha$  which is much larger than the  $\alpha = \frac{1}{3}$  that a standard VQ would produce, but the TVQ case produces a result which is the same as a standard VQ. If our asymptotic theory is correct, then the slight differences (between  $\frac{1}{3}$  in the VQ case and the  $0.31 - 0.35$  observed in the TVQ case) could arise from a combination of factors involving nonvanishing next to leading order corrections, fluctuations in the training, unsophisticated estimation of  $\alpha$ , etc. It is not necessary for us to perform a sophisticated numerical simulation to detect the changes brought about by using minimum distortion rather than nearest neighbour encoding, because the differences between the TM and the TVQ entries in Table I are dramatic enough to swamp any uncertainties that arise from our crude numerical simulation.

We show in Figure 4 some plots of the estimate of  $\alpha$  as a function of (the logarithm to base 10 of) the numbers of training steps. These plots fall into two clearly separated categories which approach (as the number of training steps increases) the results tabulated in Table I. It is clear from Figure 4 that we have not trained all the way to convergence, but the additional effort of either running

Table I: Asymptotic power law  $\alpha$  for various simple neighbourhood functions, showing both the topographic mapping case and the topographic vector quantiser case.

$\epsilon'$	TM	TVQ
0.025	0.51	0.32
0.050	0.51	0.31
0.075	0.52	0.35

our crude algorithm for longer or writing an cleverer algorithm is not justified by the simple point that we are trying to make. We see unequivocally that minimum distortion encoding leads to a dramatically different  $\alpha$  from nearest neighbour encoding, and we see with less certainty (but still fairly convincingly) that asymptotically  $\rho(x)$  has the predicted  $P(x)^{1/3}$  power law dependence.

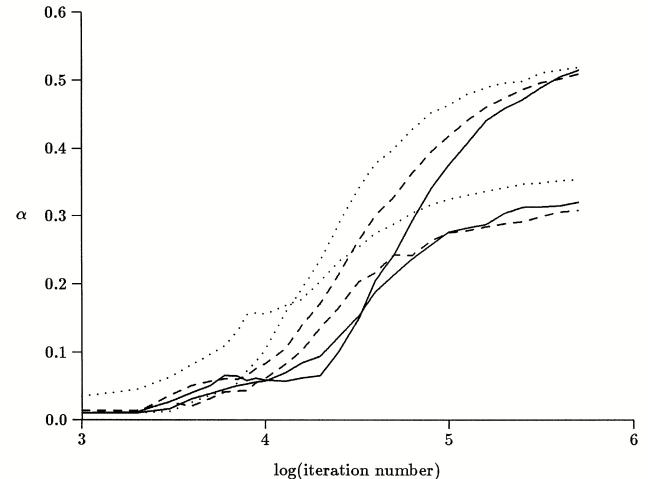


Figure 4: A plot of the power law dependence  $\alpha$  of code vector density  $\rho(x)$  on input vector density  $P(x)$  versus the number of training steps for both the topographic mapping (TM) and the topographic vector quantiser (TVQ) case. The results of using various values of  $\epsilon'$  in the update prescription (see Equation 4.2) are shown. The plots are coded as follows:  $\epsilon' = 0.025$  (solid),  $\epsilon' = 0.050$  (dashes),  $\epsilon' = 0.065$  (dots).

Despite the fact that we choose a rather small value for  $n_{cv}$  (i.e.,  $n_{cv} = 30$ ), we nevertheless obtain a result for  $\alpha$  that is remarkably close to the theoretically predicted asymptotic (i.e.,  $n_{cv} \rightarrow \infty$ ) result. This is very encouraging because we may use our asymptotic result with confidence to predict the density of CVs as a function of the input probability density, apart from some possible edge effects (that we artificially removed from our numerical simulation).

## V. CONCLUSIONS AND DISCUSSION

For topographic mappings the dependence of the asymptotic code vector density  $\rho(x)$  on the density of

inputs  $P(x)$  depends on whether we encode using the method advocated in [2] (i.e., nearest neighbour encoding) or our own method [5, 7] (i.e., minimum distortion encoding). In [9] the result  $\rho(x) \propto P(x)^\alpha$ , where  $\alpha = \frac{1}{3} \frac{(2n+1)^2}{(n+1)^2+n^2}$ , is derived for a scalar quantiser using nearest neighbour encoding, where  $n$  is the half-width of a uniform symmetric topographic neighbourhood. We obtain the result  $\alpha = \frac{1}{3}$  for a scalar quantiser using minimum distortion encoding, both for the uniform symmetric neighbourhood case and for the more general symmetric monotonically decreasing neighbourhood.

We may replace a standard topographic mapping by an equivalent topographic vector quantiser by merely changing the encoding scheme from nearest neighbour to minimum distortion. The distortion that is caused to a given input is thus calculated as a weighted average over the distortions that would have arisen from each of the code vectors in the topographic neighbourhood of the encoded input. Evidently, this is a more complicated scheme than nearest neighbour encoding, but it does lead to a simpler optimisation problem (i.e., minimum  $L_2$  distortion).

It is important to note some restrictions that apply to our new theoretical result (see the Appendix): one must use a topographic neighbourhood that is a decreasing function of distance. As one approaches a uniform topographic neighbourhood function one must use ever smaller updates in order not to destabilise the code vectors. We have not found this to be a restriction in practice.

In practical applications, both nearest neighbour and minimum distortion encoding will lead to topographic mappings that resemble each other superficially except for the different asymptotic power law in  $\rho(x) \propto P(x)^{\frac{1}{3}}$ . The only circumstance where one can be certain that minimum distortion encoding is the correct method to use is when one needs to optimise a codebook to minimise the  $L_2$  distortion that arises after passing the code index through a noisy channel [1]. However, in less well defined problems the decision about which method is best to use is unclear.

In cases where it does not matter which encoding method is used, and where it is required to estimate  $P(x)$  from the unknown positions of the code vectors, minimum distortion encoding has a distinct advantage, because  $\rho(x) \propto P(x)^{\frac{1}{3}}$  irrespective of the shape of the topographic neighbourhood function (subject to mild constraints). Thus  $\rho(x)$  can be estimated from the local density of code vector positions, whence  $P(x)$  can be estimated using  $P(x) \propto \rho(x)^3$  without having to worry about side effects that arise from the presence of a topographic neighbourhood function.

Another advantage of the minimum distortion encoding approach is that it is likely to benefit directly from new results in vector quantiser theory, simply because it is formulated in the language of vector quantisation. For instance, in [3] a coarse-to-fine strategy is suggested for training the codebook of a vector quantiser, which results

in a much more efficient algorithm than the brute-force approach of optimising a fully populated codebook. This idea can be carried over directly to a topographic vector quantiser and by analogy can be applied to train topographic mappings more efficiently than hitherto, as we showed in [6].

Other techniques may be transferred from vector quantisers, via topographic vector quantisers, to topographic mappings. For instance, multilevel (or hierarchical) codebooks that speed up the search time for locating the nearest neighbour from  $\mathcal{O}(n_{cv})$  to  $\mathcal{O}(\log n_{cv})$  can be trained.

## Appendix A: Appendix: Stability Problems

If we implement our TVQ model using the minimum distortion encoding prescription as in Equation 2.5 and Equation 2.6 with the  $[-n, +n]$  neighbourhood defined in Equation 2.12, we rapidly discover that as training proceeds the CVs tend to collapse into clusters, each of which contains  $2n + 1$  CVs. Unfortunately, this type of clustering violates our assumption that we can discard next to leading order terms in our derivation. For instance, the higher derivative terms in Equation 2.14 should not be ignored.

It is easy to convince oneself how this type of observed clustering behavior could arise. Consider the simple case of a uniform input probability density  $P(x) = P = \text{constant}$  with a  $[-n, +n]$  neighbourhood as defined in Equation 2.12. There are two cases to consider.

If we assume that the CVs are uniformly spaced with a separation  $a$ , then the Euclidean distortion  $D_2$  associated with a single CV (see Equation 2.7) reduces to

$$\begin{aligned} D_2 &= P \int_{-a/2}^{+a/2} dx \sum_{k=-n}^{+n} (x - ka)^2 \\ &= \frac{Pa^3}{3} \left[ \sum_{k=-n}^{+n} \frac{1}{4} + 6 \sum_{k=1}^n k^2 \right] \\ &= \frac{Pa^3 (2n+1)^3}{12} \end{aligned} \quad (\text{A1})$$

where we use  $\sum_{k=1}^n k^2 = \frac{1}{6}n(n+1)(2n+1)$ .

If we assume that the CVs are clustered into sets of  $2n + 1$  CVs superimposed on each other and that these clusters are uniformly spaced with a separation  $(2n+1)a$  (to ensure that the overall density of CVs is the same as in the previous case), then the Euclidean distortion  $D_2$  associated with a single cluster reduces to

$$\begin{aligned} D_2 &= P \int_{-(2n+1)a/2}^{+(2n+1)a/2} dx x^2 \\ &= \frac{Pa^3 (2n+1)^3}{12} \end{aligned} \quad (\text{A2})$$

Note that all of this distortion is associated with just one of the  $2n + 1$  CVs in the cluster (the central CV, in fact),

while the remaining  $2n$  CVs act as a buffer of  $n$  CVs on either side of the central CV. This prevents the code index distortion process from moving the code index into one of the two adjacent clusters of  $2n + 1$  CVs, which would have caused a large distortion to occur. Thus the encoding process maps only to the central CV in each cluster, so that subsequently the code index distortion process has no observable effect.

The uniform case in Equation A1 and the clustered cases in Equation A2 both lead to the same overall Euclidean distortion  $D_2$  when all CVs are accounted for. This is a counterexample to our assumption that the optimal CV density is a smoothly varying function of position, because we have presented two different solutions to the problem of minimising  $D_2$ , one of which does not have a smooth CV density.

We may slightly generalise the above result by modifying Equation 2.12 to define a symmetric neighbourhood function as

$$\pi_{y',y} = \begin{cases} \pi_{|y'-y|} & \text{if } |y' - y| \leq n \\ 0 & \text{if } |y' - y| > n \end{cases} \quad (\text{A3})$$

where we impose the normalisation condition  $\sum_{k=-n}^{+n} \pi_k = 2n + 1$  to ensure compatibility with Equation A1 and Equation A2. This change does not affect the clustered case in Equation A2 because the distortion process has no

affect there, but it modifies the uniform case in Equation A1 to become

$$D_2 = \frac{P a^3 \left( 2n + 1 + 24 \sum_{k=1}^n k^2 \pi_k \right)}{12} \quad (\text{A4})$$

where we have made use of the normalisation condition. As expected, Equation A4 reduces to Equation A1 in the case  $\pi_k = 1$ ,  $k = 1, 2, \dots, n$ .

By inspecting Equation A4 we see that  $D_2$  is smaller for uniform spacing than for clustered spacing when  $\pi_k$  is chosen in such a way that it suppresses the contribution from large values of  $k$  in the summation: these terms would otherwise dominate the summation owing to the factor  $k^2$ . This amounts to saying that a “tapered” neighbourhood function ensures the stability of uniformly spaced CVs with respect to clustered CVs.

Although the arguments here have been based entirely on the assumption that  $P(x) = P = \text{constant}$ , we assume that the same general conclusion may be carried over to the more general case of a spatially varying  $P(x)$ . This extension is broadly valid provided that  $P(x)$  varies slowly over lengths of the order of the distance between adjacent CVs. By performing some simple numerical investigations one indeed discovers that the use of tapered neighbourhood functions stabilises the training of our TVQ model.

- 
- [1] D S Bradburn, *Reducing transmission error effects using a self-organising network*, Proceedings of IJCNN international joint conference on neural networks (Washington DC), IEEE, 1989, pp. 531–537.
  - [2] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
  - [3] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
  - [4] S P Lloyd, *Least squares quantisation in PCM*, IEEE Transactions on Information Theory **28** (1982), no. 2, 129–137.
  - [5] S P Luttrell, *Hierarchical self-organising networks*, Proceedings of IEE Conference on Artificial Neural Networks (London), IEE, 1989, pp. 2–6.
  - [6] ———, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
  - [7] ———, *Self-organisation: a derivation from first principles of a class of learning algorithms*, Proceedings of IJCNN international joint conference on neural networks (Washington DC), IEEE, 1989, pp. 495–498.
  - [8] J Max, *Quantising for minimum distortion*, IEEE Transactions on Information Theory **6** (1960), no. 1, 7–12.
  - [9] H Ritter, *Asymptotic level density for a class of vector quantisation processes*, IEEE Transactions on Neural Networks **2** (1991), no. 1, 173–175.

# Self-Supervised Adaptive Networks \*

S. P. Luttrell

*Image Processing Research Section, Defence Research Agency,  
St Andrews Rd, Malvern, Worcs, WR14 3PS, United Kingdom*

A scheme for training multilayer unsupervised networks is presented, in which control signals propagate downwards from the higher layers to influence the optimisation of the lower layers. Because there is no external teacher involved, this is called self-supervised training. The author demonstrates both theoretically and numerically how self-supervision emerges when a simple network built out of vector quantisers is optimised.

## I. INTRODUCTION

The supervised training of adaptive networks can solve problems which require an optimal nonlinear transformation from an input to an output space. Conversely, unsupervised training of adaptive networks can solve problems which require an optimal encoding of an input space. Supervised training requires an external teacher, whereas unsupervised training does not.

There is a useful middle ground between the extremes of supervised and unsupervised networks. Assume that a multilayer unsupervised network is a stack of single-layer unsupervised networks. There are two distinct types of cost function that we could introduce: global and local. A local cost function might ensure that each layer strives to code the output of the previous layer, so that it can be reconstructed with low distortion, as in a single-layer unsupervised network. A global cost function might ensure that when the input is passed through all the layers, and then passed all the way back again, it still forms a low-distortion reconstruction. To achieve global optimisation, we must train the layers cooperatively so that each layer of the network internally supervises other layers. We call this mode of operation ‘self-supervision’, and in this paper, which is an extended version of [8], we aim to demonstrate this type of network training.

Although training a one-layer unsupervised network (input → code) is equivalent to training an appropriate two-layer supervised network (input → hidden → output) in which the output is required to reproduce the input (i.e. an autoassociative supervised network), this equivalence does not generalise to multilayer unsupervised networks. The local cost functions that connect adjacent pairs of layers in multilayer unsupervised networks cannot be accounted for in standard autoassociative network models, which have a cost function defined in terms of the input and output only.

Self-supervision is important in low-level vision applications, such as the extraction of features from images.

To ensure that useful information arrives at a given layer of a vision network, we must ensure that lower (or earlier) network layers process their inputs appropriately. This type of global cost function leads naturally to self-supervised training schemes. A similar application is data fusion of the information collected by separate sensors, where a multilayer system progressively transforms and collates the sensor information in a macroscopic version of low-level vision. Again, self-supervision emerges naturally from the global cost function.

In [7] we discuss a hierarchical network that computes the maximum relative entropy estimate of the joint probability density function of its own inputs. We show that this reduces to maximising the sum of mutual informations between various internal components of the networks; this is a global cost function. This training scheme makes different regions of the hierarchical network mutually supervise each other, and is thus an example of a self-supervised network.

Because the goal of this paper is to explain the principles that underlie self-supervision, we limit our attention to the case of a simple two-layer unsupervised network, built out of vector quantisers [1, 4, 5].

## II. VECTOR QUANTISATION MODEL

In this section we present a résumé of vector quantisation theory (e.g. the so-called LBG algorithm [3]), its extension to vector quantisation for communication over a noisy communication channel [2, 4, 5], and its further extension to mutually interfering communication channels [8]. The noisy communication channel model is formally equivalent to the model that is implicit in the theory of topographic mappings [1].

Throughout this paper we make use of functions such as  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$ , whose ‘input’ and ‘output’ spaces we treat as continuous, rather than discrete. We do this for two reasons. First, it makes our derivations easier to perform using the rich language of continuum notation. Secondly, it demonstrates that our results are general statements about transformations between input and output spaces, and are thus not limited to vector quantisers (continuous  $\mathbf{x}$ , discrete  $\mathbf{y}$ ).

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 11, 2019.

This paper appeared in IEE Proceedings-F, 1992, vol. 139, no. 6, pp. 371-377. Paper 9135F (C4, E5), first received 29th January and in revised form 17th August 1992. © British Crown Copyright 1992/MOD.

### A. Vector quantisation for noiseless and noisy channels

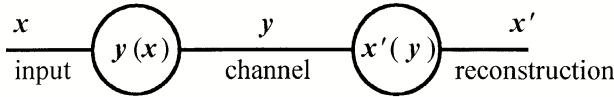


Figure 1: A standard vector quantiser.  $y(\mathbf{x})$  is an encoder that transforms the input  $\mathbf{x}$  into a code  $\mathbf{y}$ .  $\mathbf{x}'(\mathbf{y})$  is a decoder that transforms the code  $\mathbf{y}$  into a reconstruction  $\mathbf{x}'$ , which approximates the original input  $\mathbf{x}$ .

Consider vector quantisation for transmission along a noiseless communication channel. Define  $\mathbf{x}$  as the input data,  $\mathbf{y}$  as the encoded data, and  $\mathbf{x}'$  as the reconstruction of the input data. Define  $y(\mathbf{x})$  as the encoding operation  $\mathbf{x} \rightarrow \mathbf{y}$ , and  $x'(\mathbf{y})$  as the decoding operation  $\mathbf{y} \rightarrow \mathbf{x}'$ , which yields overall  $\mathbf{x}' = x'(\mathbf{y}(\mathbf{x}))$  (Figure 1). We may combine these quantities to obtain the average  $L_2$  (i.e. Euclidean) distortion  $D_1$

$$D_1 = \int d\mathbf{x} P(\mathbf{x}) \|x'(\mathbf{y}(\mathbf{x})) - \mathbf{x}\|^2 \quad (2.1)$$

To minimise  $D_1$ , we need to optimise the pair of functions  $(\mathbf{y}(\mathbf{x}), \mathbf{x}'(\mathbf{y}))$ ; the LBG algorithm [3] is an effective solution to this.

Now consider a noisy communication channel, which we may model as a generalisation of Equation 2.1

$$D_2 = \int d\mathbf{x} P(\mathbf{x}) \int d\mathbf{y}' \pi(\mathbf{y}' - \mathbf{y}(\mathbf{x})) \|x'(\mathbf{y}') - \mathbf{x}\|^2 \quad (2.2)$$

In Equation 2.2 we assume that  $\mathbf{y}' = \mathbf{y}(\mathbf{x}) + \mathbf{n}$ , where  $\mathbf{n}$  is an additive random noise variable with PDF  $\pi(\mathbf{n})$ . Furthermore, we assume that  $\mathbf{x}$  and  $\mathbf{n}$  are statistically independent, so that  $P(\mathbf{x}, \mathbf{n}) = P(\mathbf{x})\pi(\mathbf{n})$ . A generalisation of the LBG algorithm [2, 5] is an effective solution to this optimisation problem, as we explain below.

### B. Nearest neighbour and minimum distortion encoding

In order to minimise  $D_2$  we functionally differentiate it and then locate the zeros of  $\frac{\delta D_2}{\delta \mathbf{y}(\mathbf{x})}$  and  $\frac{\delta D_2}{\delta \mathbf{x}'(\mathbf{y})}$ . After some calculation we obtain

$$\begin{aligned} \mathbf{x}'(\mathbf{y}) &= \frac{\int d\mathbf{x} P(\mathbf{x}) \pi(\mathbf{y} - \mathbf{y}(\mathbf{x})) \mathbf{x}}{\int d\mathbf{x} P(\mathbf{x}) \pi(\mathbf{y} - \mathbf{y}(\mathbf{x}))} & (a) \\ \Delta \mathbf{x}'(\mathbf{y}) &= \varepsilon \pi(\mathbf{y} - \mathbf{y}(\mathbf{x})) (\mathbf{x} - \mathbf{x}'(\mathbf{y})) & (b) \end{aligned} \quad (2.3)$$

$$\mathbf{y}(\mathbf{x}) = \arg \min_{\mathbf{y}} \int d\mathbf{y}' \pi(\mathbf{y}' - \mathbf{y}) \|x'(\mathbf{y}') - \mathbf{x}\|^2 \quad (2.4)$$

where  $\arg \min \mathbf{y}(\cdot)$  means ‘select the value of  $\mathbf{y}$  that minimises  $(\cdot)$ ’. These results are a compact specification of a pair of vector quantiser training algorithms:

1. Batch update (Equation 2.3a). This is the generalisation to noisy communication channels of one cycle of the LBG algorithm [3].
2. Continuous update (Equation 2.3b). This is identical to the topographical mapping training algorithm [1], apart from a subtle difference in the encoding prescription that we discuss below.  $\pi(\mathbf{n})$  can therefore be interpreted as a neighbourhood function.

In Equation 2.4 there are two distinct cases to consider:

1. Nearest neighbour encoding (NN): when  $\pi(\mathbf{n}) = \delta(\mathbf{n})$  (i.e. the noiseless channel case) Equation 2.4 specifies a nearest neighbour encoding prescription that we denote as  $\mathbf{y}^0(\mathbf{x})$ . This prescription selects the encoded version of  $\mathbf{x}$  to be the  $\mathbf{y}$  that minimises the distortion  $\|x'(\mathbf{y}) - \mathbf{x}\|^2$ .

2. Minimum distortion encoding (MD): when  $\pi(\mathbf{n}) \neq \delta(\mathbf{n})$  Equation 2.4 specifies a minimum distortion encoding prescription  $\mathbf{y}(\mathbf{x})$ , that anticipates the subsequent effect of possible channel distortions. This prescription selects the encoded version of  $\mathbf{x}$  to be the  $\mathbf{y}$  that minimises the expected distortion  $\int d\mathbf{y}' \pi(\mathbf{y}' - \mathbf{y}) \|x'(\mathbf{y}') - \mathbf{x}\|^2$ .

NN encoding is sometimes a suitable approximation to MD encoding when  $\pi(\mathbf{n}) \neq \delta(\mathbf{n})$ , and it is widely used in the literature as the standard encoding prescription irrespective of whether or not it minimises a meaningful distortion.

### C. Relationship to unsupervised neural networks

The vector quantiser in Figure 1 can be interpreted as a ‘winner-take-all’ neural network, as we show in Figure 2, where we fold the vector quantiser diagram to make the relationship clearer.

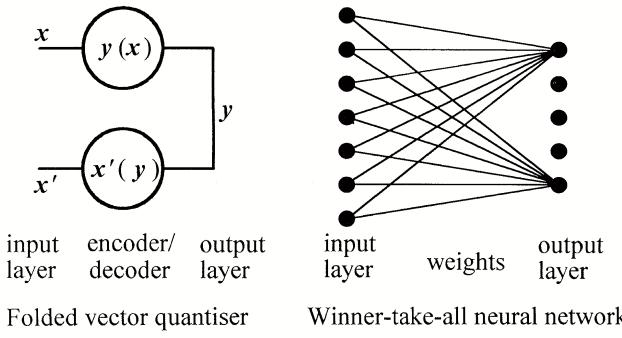


Figure 2: Equivalence of a folded quantiser and a single-layer ‘winner-take-all’ neural network. The correspondence between the diagrams is as follows:  $\mathbf{x}$  is the pattern of activity in the input layer,  $\mathbf{y}(\mathbf{x})$  is the prescription for locating the winner in the output layer,  $\mathbf{y}$  is the location of the winning node in the output layer (which would be a two-component vector if the output layer were a two-dimensional sheet of nodes, for instance), and  $\mathbf{x}'(\mathbf{y})$  is the weight vector attached to output node  $\mathbf{y}$ . The nearest neighbour prescription is widely used for  $\mathbf{y}(\mathbf{x})$  in the neural network literature. The neural network operates simultaneously to perform encoding (i.e. forward pass) and decoding (i.e. backward pass).

The noisy communication channel version of the vector quantiser is more complicated. The noise process  $\pi(\mathbf{n})$  causes confusion between different values of  $\mathbf{y}$  via the transformation  $\mathbf{y} \rightarrow \mathbf{y} + \mathbf{n}$ . The vector quantiser re-

sponds by ensuring that  $\mathbf{x}'(\mathbf{y})$  is not too sensitive to the types of change in the value of  $\mathbf{y}$  that this noise process brings about. If  $\pi(\mathbf{n})$  is a localised function of  $\mathbf{n}$ , such as a zero-mean Gaussian, then  $\mathbf{x}'(\mathbf{y})$  must become a slowly varying function of  $\mathbf{y}$ .

In the neural network picture this would translate into a smooth variation of weight vector as we pass from node to node in the output layer. This is exactly what we observe in a topographical mapping neural network using neighbourhood function  $\pi(\mathbf{n})$  [1]. The exact details of the standard topographical mapping training algorithm are not precisely the same as Equation 2.3b; we use minimum distortion encoding, whereas topographic mapping networks use nearest neighbour encoding. In [6] we demonstrate that using minimum distortion encoding in preference to nearest neighbour encoding, actually simplifies some of the properties of topographical mappings. In particular, the density of weight vectors acquires a functional form that is independent of the details of the neighbourhood function.

#### D. Vector quantisation for coupled channels

Now consider a pair of mutually coupled communication channels, which we may model as a generalisation of Equation 2.2. The Euclidean distortion in such a system is given by

$$D_3 = \int d\mathbf{x}_1 d\mathbf{x}_2 P(\mathbf{x}_1, \mathbf{x}_2) \int d\mathbf{y}_1' P(\mathbf{y}_1' | \mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2)) \|\mathbf{x}_1'(\mathbf{y}_1') - \mathbf{x}_1\|^2 + (1 \longleftrightarrow 2) \quad (2.5)$$

We may interpret the various contributions to this expression for  $D_3$  as follows:

1. The  $\int d\mathbf{x}_1 d\mathbf{x}_2 P(\mathbf{x}_1, \mathbf{x}_2) (\dots)$  integration averages over all pairs of inputs  $(\mathbf{x}_1, \mathbf{x}_2)$  to channels 1 and 2, and  $P(\mathbf{x}_1, \mathbf{x}_2)$  specifies the probability density with which each pair occurs.
2. The  $\int d\mathbf{y}_1' P(\mathbf{y}_1' | \mathbf{y}_1, \mathbf{y}_2)$  integration averages over all possible distortions of channel 1, due to the mutual coupling of channels 1 and 2.

3.  $\|\mathbf{x}_1'(\mathbf{y}_1') - \mathbf{x}_1\|^2$  is the Euclidean distance between the input vector  $\mathbf{x}_1$  and its reconstruction  $\mathbf{x}_1'(\mathbf{y}_1')$  from the distorted version of channel 1.

4.  $(1 \longleftrightarrow 2)$  denotes an analogous term for channel 2.

By comparing Equation 2.5 with Equation 2.2, we see that the marginals  $P(\mathbf{y}_1' | \mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2' | \mathbf{y}_1, \mathbf{y}_2)$  now play the part of the noise PDF  $\pi(\mathbf{y} - \mathbf{y}(\mathbf{x}))$ .

We functionally differentiate  $D_3$  to obtain ( $k = 1$  and 2)

$$\mathbf{x}_k'(\mathbf{y}_k') = \frac{\int d\mathbf{x}_1 d\mathbf{x}_2 P(\mathbf{x}_1, \mathbf{x}_2) P(\mathbf{y}_k' | \mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2)) \mathbf{x}_k}{\int d\mathbf{x}_1 d\mathbf{x}_2 P(\mathbf{x}_1, \mathbf{x}_2) P(\mathbf{y}_k' | \mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2))} \quad (a)$$

$$\Delta \mathbf{x}_k'(\mathbf{y}_k') = \varepsilon P(\mathbf{y}_k' | \mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2)) (\mathbf{x}_k - \mathbf{x}_k'(\mathbf{y}_k)) \quad (b)$$

$$(\mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2)) = \arg \min_{(\mathbf{y}_1, \mathbf{y}_2)} \left( \int d\mathbf{y}_1' P(\mathbf{y}_1' | \mathbf{y}_1, \mathbf{y}_2) \|\mathbf{x}_1'(\mathbf{y}_1') - \mathbf{x}_1\|^2 + (1 \longleftrightarrow 2) \right) \quad (2.7)$$

Strictly speaking, when we functionally differentiate  $D_3$  to determine its dependence on  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$ , we should include any derivatives that arise from the implicit dependence of  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  on  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$ . However, we shall assume that  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  are time-averaged quantities, and so their instantaneous dependence on the functions  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$  is weak enough to be ignored. This is a type of ‘mean field’ approximation.

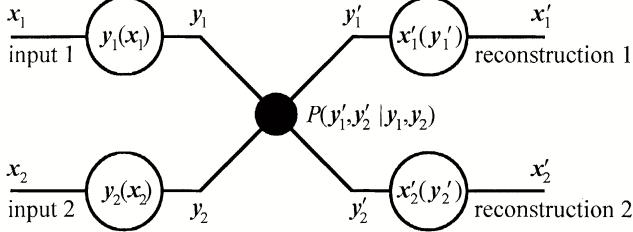


Figure 3: A pair of coupled vector quantisers of the type shown in Figure 1. The conditional  $P(\mathbf{y}_1', \mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  describes the distortion that arises due to coupling of the channels. If the channels were conditionally independent then  $P(\mathbf{y}_1'|\mathbf{y}_1)$  and  $P(\mathbf{y}_2'|\mathbf{y}_2)$  would suffice. Many distortion processes can be modelled using  $P(\mathbf{y}_1', \mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$ , but in the Appendix we restrict our attention to the distorting effects of a second stage of vector quantisation

Equation 2.7 specifies a minimum distortion prescription in which we simultaneously optimise  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$ ; this is ‘cooperative encoding’. Note that, for simplicity, in Figure 3 we omit any reference to the fact that the encoding functions  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$  are implicitly coupled via Equation 2.7. If we ignore the coupling between the channels, we replace  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  with  $P(\mathbf{y}_1'|\mathbf{y}_1)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  with  $P(\mathbf{y}_2'|\mathbf{y}_2)$ ; this is ‘independent encoding’. Furthermore, in both the cooperative and the independent encoding cases, we could use the nearest neighbour prescription, instead of the full minimum distortion prescription. In Table I we summarise the various modes in which the two channels operate.

### E. A specific model of the channel coupling

In Figure 4 we represent diagrammatically in  $(\mathbf{y}_1, \mathbf{y}_2)$ -space (and  $(\mathbf{y}_1', \mathbf{y}_2')$ -space) the role of the various PDFs in Equation 2.5. We show in the Appendix that, if we pass the output pair  $(\mathbf{y}_1, \mathbf{y}_2)$  through another stage of vector quantisation, then the bias of  $P(\mathbf{y}_1', \mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  is towards higher values of  $P(\mathbf{y}_1, \mathbf{y}_2)$ . The overall effect is to bias the two marginals  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  (as shown) in a data-dependent way. This type of effect cannot be accounted for in the original additive noise model using  $\pi(\mathbf{y} - \mathbf{y}(\mathbf{x}))$ .

In the Appendix we derive the first-order approximation to the effect of a second-stage of vector quantisation

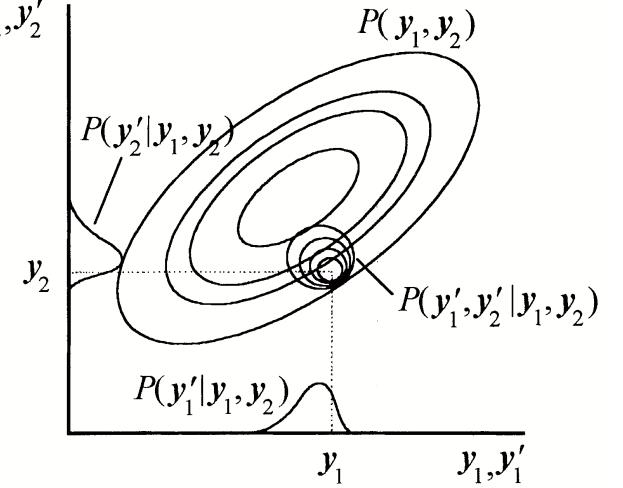


Figure 4: Contours of the PDFs in a typical joint distortion situation. As we show in the Appendix, the distortion PDF  $P(\mathbf{y}_1', \mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  has a simple dependence on the joint density of the vector quantiser outputs. Typically, the bias of  $P(\mathbf{y}_1', \mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  is in the direction of increasing  $P(\mathbf{y}_1, \mathbf{y}_2)$ , as we show in the diagram. The neighbourhood functions for  $\mathbf{y}_1$  and  $\mathbf{y}_2$  are the marginal PDFs  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  of  $P(\mathbf{y}_1', \mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$ . The bias of both of these neighbourhood functions depends jointly on the values of  $\mathbf{y}_1$  and  $\mathbf{y}_2$ , and so the two channels mutually influence each other.

in the form

$$P(\mathbf{y}'|\mathbf{y}) \propto \rho(\mathbf{y}') \exp \left( -\frac{\alpha_N \rho(\mathbf{y}) \| \mathbf{y}' - \mathbf{y} \|^N}{N} \right) \quad (2.8)$$

where we use the notation  $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)$ , and we use  $\rho(\mathbf{y})$  to denote the density of code vectors of the second-stage vector quantiser. For scalar  $\mathbf{y}_1$  and  $\mathbf{y}_2$  (i.e.  $\mathbf{y}_1 = y_1$  and  $\mathbf{y}_2 = y_2$ ) we can readily marginalise this result to obtain

$$P(y_1'|\mathbf{y}_1, \mathbf{y}_2) \propto \rho(y_1', y_2) \exp \left( -\pi \rho(y_1, y_2) (y_1' - y_1)^2 \right) \quad (2.9)$$

A similar result holds for  $P(y_2'|\mathbf{y}_1, \mathbf{y}_2)$ . The mean and standard deviations of  $y_1' - y_1$  are

$$\begin{aligned} \text{mean} &\approx \frac{1}{2\pi\rho^2(y_1, y_2)} \frac{\partial \rho(y_1, y_2)}{\partial y_1} \\ \text{SD} &\approx \frac{1}{\sqrt{2\pi\rho(y_1, y_2)}} \end{aligned} \quad (2.10)$$

A similar pair of results holds for  $y_2' - y_2$ . Note that in Figure 4 the bias of the neighbourhood functions is consistent with Equation 2.10. Because  $\rho(y_1, y_2) \propto \sqrt{P(y_1, y_2)}$  (for an optimised vector quantiser) [9] we may also use Equation 2.9 to relate the neighbourhood functions  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  of the two channels to their joint PDF  $P(\mathbf{y}_1, \mathbf{y}_2)$ .

Table I: Various encoding and channel modes. NN = nearest neighbour encoding. MD = minimum distortion encoding. I = independent channels. C = correlated channels.

	Nearest neighbour encoding	Minimum distortion encoding
Independent channels	NN/I	MD/I
Correlated channels	NN/C	MD/C

### F. Self-supervision

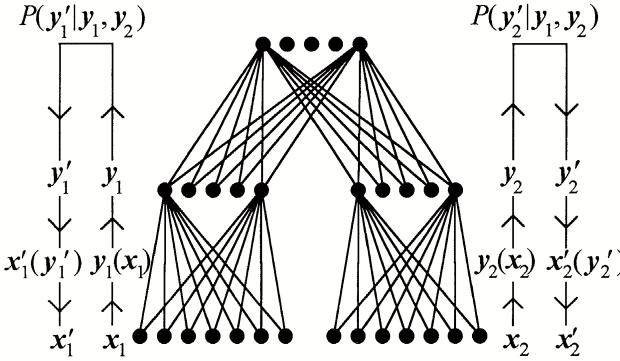


Figure 5: Two-layer self-supervised neural network. The orientation of this diagram is 90° clockwise relative to Figure 2. The bottom left and right parts of the diagram correspond to a pair of folded vector quantisers in channels 1 and 2, respectively (i.e. two copies of Figure 2). The top part of the diagram implements the joint distortion  $P(\mathbf{y}_1', \mathbf{y}_2' | \mathbf{y}_1, \mathbf{y}_2)$  that converts  $(\mathbf{y}_1, \mathbf{y}_2)$  into  $(\mathbf{y}_1', \mathbf{y}_2')$ ; this distortion acts after encoding (i.e. forward pass) but before decoding (i.e. backward pass) in the bottom part of the diagram. The top part of the diagram is only a schematic representation of top-down control of lower layers by higher layers of a multilayer network; the top-down control could come from any source.

A subtle side effect of channel coupling is ‘self-supervision’. This phenomenon is easiest to understand in terms of the neural network viewpoint. When we optimise the network in Figure 5, the interaction between the layers causes the optimisation of the bottom left (channel 1) and bottom right (channel 2) parts of the network to become indirectly coupled through the top (the coupled distortion) part of the network; this is ‘self-supervision’. If the top part of the network were absent, then it would reduce to two separate unsupervised networks (corresponding to channels 1 and 2).

The self-supervision principle is quite general: whenever we build a multilayer unsupervised network, the various layers must cooperate to minimise the global (and local) distortion function. This cooperation manifests itself as control signals that the higher layers of the network pass down to the lower layers of the network. The similarity to the back-propagating signals that we use to train a supervised network is not accidental. Indeed, we can build a hybrid network in which an external teacher

gives rise to standard supervised backpropagating sig-

Table II: Values of  $\pi_-$  and  $\pi_0$  that we use in four separate numerical experiments. In experiment 1 we use an unbiased distortion, whereas in experiments 2-4 we use a biased distortion, as indicated in Equation 2.9.

	$\pi_-$	$\pi_0$	$\pi_+$
1	0.20	0.60	0.20
2	0.15	0.60	0.25
3	0.10	0.60	0.30
4	0.05	0.60	0.35

nals, and the internal layers of the network give rise to self-supervised back-propagating signals.

## III. NUMERICAL EXPERIMENTS

In this Section we present some numerical simulations that demonstrate self-supervision in a simple network of the type shown in Figure 3 (whose neural network interpretation is shown in Figure 5). We run all of our numerical simulations using a four-dimensional input vector  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) = (x_{11}, x_{12}, x_{21}, x_{22})$ , and with scalar outputs from the encoders  $y_1(\mathbf{x}_1)$  and  $y_2(\mathbf{x}_2)$ . This is the minimal network that we can use to demonstrate self-supervision. In realistic applications more complicated networks would arise, but they would all operate according to the same general principles.

### A. Basic network operation

$$P_{\text{approx}}(y_1' | y_1, y_2) = \begin{cases} \pi(y_1' - y_1) & \partial\rho(y_1, y_2)/\partial y_1 \geq 0 \\ \pi(y_1 - y_1') & \partial\rho(y_1, y_2)/\partial y_1 < 0 \end{cases} \quad (3.1)$$

$$\pi(\Delta y) \equiv \begin{cases} \pi_- & \Delta y = -1 \\ \pi_0 & \Delta y = 0 \\ \pi_+ & \Delta y = +1 \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

$$D(\mathbf{x}) = \int dy_1' P(y_1'|y_1, y_2) \left( (x_{11}'(y_1) - x_{11})^2 + (x_{12}'(y_1) - x_{12})^2 \right) + (1 \longleftrightarrow 2) \quad (3.3)$$

1. Clamp the inputs. We generate  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) = (x_{11}, x_{12}, x_{21}, x_{22})$  as follows: we choose  $(x_{11}, x_{12})$  to be a uniformly distributed random vector in a disc-shaped region, and then generate  $(x_{21}, x_{22})$  by rotating  $(x_{11}, x_{12})$  about the disc's centre by a random angle uniformly sampled from the interval  $[-\theta, +\theta]$ . We use this prescription to ensure that the marginal PDFs  $P(x_{21}, x_{22})$  and  $P(x_{21}, x_{22})$  are the same, and to control the degree of correlation between  $(x_{11}, x_{12})$  and  $(x_{21}, x_{22})$ . We use  $\theta = 0.5$  and  $\theta = 1.0$  in our simulations.

2. Compute the nearest neighbours. This yields  $\mathbf{y}^0 = (y_1^0, y_2^0)$ .

3. Compute the distortion PDFs. Although we couch the following discussion in terms of  $y_1$  alone, the same remarks also apply to  $y_2$ . We generate a convenient numerical approximation to  $P(y_1'|y_1, y_2)$  as follows. First, we use the asymptotic result  $\rho \propto P^{\frac{N}{N+2}} \propto P^{1/2}$  (for the case of  $N = 2$  dimensions) in Equation 2.9 [9]. Secondly, we approximate  $P(y_1'|y_1, y_2)$  as in Equation 3.1 where, for discrete-valued  $y$ , we define  $\pi(y_1' - y_1)$  as in Equation 3.23.2 that  $P_{\text{approx}}(y_1'|y_1, y_2)$  is a fairly crude approximation, but nevertheless it can correctly preserve the sign of the bias of  $P(y_1'|y_1, y_2)$ , which is crucial to the success of our self-supervision demonstration.

More sophisticated  $P_{\text{approx}}(y_1'|y_1, y_2)$  yield the same ranking when we compare the distortions that occur for each of the alternatives in Table I. Table II lists the values of  $\pi_\pm$  and  $\pi_0$  that we use in four separate numerical simulations.

If we are not using minimum distortion encoding, there is no need to refine the nearest neighbour encoding  $\mathbf{y}^0$  and so we now jump to step 6.

4. Compare the expected reconstruction error. We may write the expected reconstruction error  $D(\mathbf{x})$  for the current input vector  $\mathbf{x}$  as in Equation 3.3. We evaluate the integral over  $y_1'$  using the representation of  $P(y_1'|y_1, y_2)$  that we determined in step 3. An analogous result also holds for the integral over  $y_2'$ .

5. Adjust the encoding to reduce the error. We must now investigate how  $D(\mathbf{x})$  varies in the vicinity of our initial guess  $(y_1^0, y_2^0)$  to locate the local minimum  $(y_1, y_2)$  of  $D(\mathbf{x})$ , which in general is not equal to  $(y_1^0, y_2^0)$  (i.e. minimum distortion encoding is not the same as nearest neighbour encoding).

Note that, for each alternative value of  $(y_1, y_2)$  that we investigate, we must repeat step 3 and

step 4 to determine the corresponding value of  $D(\mathbf{x})$ , because the pair of neighbourhood functions  $P(y_1'|y_1, y_2)$  and  $P(y_2'|y_1, y_2)$  depend on  $(y_1, y_2)$ . In our simulations we explore only the immediate neighbourhood of  $(y_1^0, y_2^0)$ , as specified by  $y_1 \in \{y_1^0, y_1^0 \pm 1\}$ ,  $y_2 \in \{y_2^0, y_2^0 \pm 1\}$ . Ideally we should perform an exhaustive search of all possible pairs  $(y_1, y_2)$  to find the one that yields the minimum distortion, but this is too costly.

6. Update the code vectors. The code vector update prescription is the scalar version of Equation 2.6b. We do not gradually reduce  $\varepsilon$  to zero as is usually the case, but instead use  $\varepsilon = 0.1$  throughout the optimisation.

We implement a refinement of Equation 2.6b in which we start with only two code vectors, which we then use to initialise a more refined optimisation using four code vectors, and so on. We optimise each generation of code vectors using 50 training vectors per code vector, before initialising the next generation. In our simulations we stop at eight code vectors. This 'coarse to fine' strategy is very effective, and rapidly produces an optimum set of code vectors. We discuss this whole training procedure in detail in [4].

7. Update the histogram. We update a leaky histogram representation of  $P(y_1, y_2)$  by both topping up and leaking away the contents of each bin of an  $8 \times 8$  bin histogram. In order to limit the computational cost, we top up the appropriate bin at every update step, but leak away all the bins (by multiplying by 0.5) on every 100th update step.

## B. Experimental results

Figure 6 presents the results of several numerical simulations. Each result is the average of the value of the Euclidean distortion that we obtain from 16 independent simulations (in each simulation we accumulate statistics for 256 test set samples).

1. When we use a symmetrical neighbourhood function (i.e. entry number 1 in Table II) we obtain approximately the same distortion in all four cases. We would expect NN/I = NN/C = MD/I = MD/C in the limit of a zero-width neighbourhood function, and NN/I = NN/C > MD/I = MD/C for a finite-width symmetrical neighbourhood function. The slight improvement that MD encoding gives

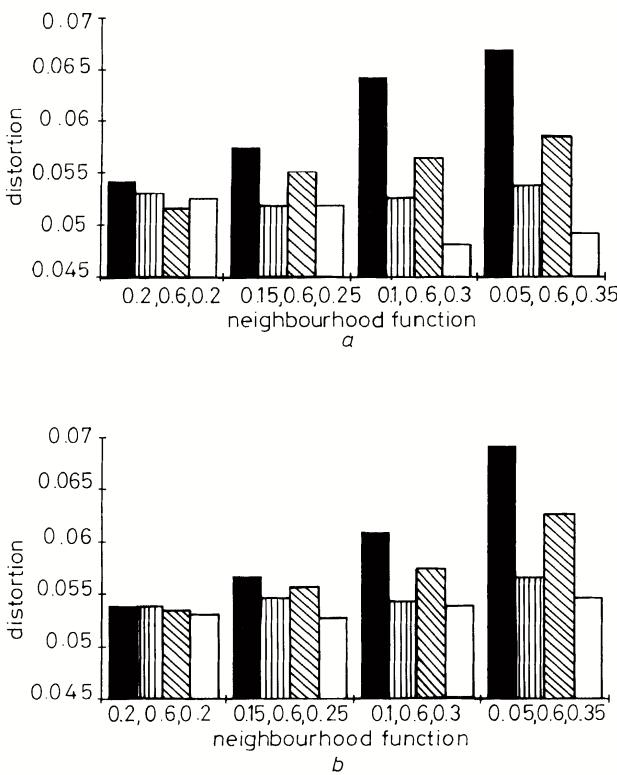


Figure 6: Results of the numerical experiments presented together for comparison. The error bars in all cases are approximately 0.001. There are two separate batches of experiments using different degrees of correlation between the channels ( $\theta = 0.5$  in *a* and  $\theta = 1.0$  in *b*). Within each batch there are four different choices for the neighbourhood function  $\pi$  (see Table II) which are listed along the horizontal axis. For each  $\pi$  there are four results (NN/I, NN/C, MD/I and MD/C) corresponding to the two encoding modes and the two channel modes (see Table I).

compared to NN encoding shows that MD encoding yields measurable effects even for symmetrical neighbourhood functions [6].

2. When we use a biased neighbourhood function, we find that MD encoding systematically produces a smaller distortion than the corresponding NN encoding. We would expect this result because MD anticipates the effect (on average) of the channel distortion, whereas NN does not.
3. When we compare the different channel modes, we find that the C channels systematically produce a smaller distortion than the corresponding I channels. We would expect this, because correlated noise causes less damage than uncorrelated noise.
4. When we compare the two different degrees of correlation  $\theta$ , we find that the C channels systematically produce a smaller distortion for the higher degree of correlation (i.e. the smaller value of  $\theta$ ).

The I channels do not distinguish between the different degrees of correlation.

This behaviour demonstrates that using neighbourhood functions  $P(y_1'|y_1, y_2)$  and  $P(y_2'|y_1, y_2)$  that depend on the *pair* of outputs  $(y_1, y_2)$ , is better than using  $P(y_1'|y_1)$  and  $P(y_2'|y_2)$  that depend on only *one* output; this result is true whether we use NN or MD encoding. The cooperative behaviour that  $P(y_1'|y_1, y_2)$  and  $P(y_2'|y_1, y_2)$  induce can be described in a number of ways, but we prefer to use the multilayer unsupervised network viewpoint, and thus describe it as ‘self-supervision’.

#### IV. CONCLUSIONS

The main purpose of this paper is to explain how self-supervision arises in multilayer networks. Thus we consider the problem of minimising the distortion (between input and output) that occurs when a pair of communication channels mutually interfere with each other. This system is equivalent to an unsupervised winner-take-all neural network. We demonstrate how a set of neighbourhood functions (for ordering the vector quantiser codebooks) emerges from the analysis. Furthermore, we observe that each channel influences the other’s neighbourhood function - an effect that we call self-supervision.

Self-supervision of multilayer unsupervised networks is important, because different parts of such a network co-operate to process the input data in a coordinated fashion, without the need to supply an external teacher. In general, an unsupervised multilayer network supervises its own internal operation by passing control signals back from the higher layers to lower layers, which in turn cause the lower layers to process their inputs more effectively.

It is also possible to construct hybrid multilayer networks in which both external-supervision and self-supervision terms contribute. The external-supervision terms arise from the choice of the supervised contribution to the network cost function (e.g. minimum average squared output error), whereas the self-supervision terms arise from the choice of the unsupervised contribution to the network cost function (e.g. minimum average squared input reconstruction error).

#### Appendix A

In this appendix we present an analytically solvable model of the neighbourhood functions that arise in the vector quantiser system that we discuss in Figure 4, where we use  $P(\mathbf{y}_1', \mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  to describe the effect of a second stage of vector quantisation. We use  $\mathbf{y}$  to denote  $(\mathbf{y}_1, \mathbf{y}_2)$  and  $\mathbf{y}'$  to denote  $(\mathbf{y}_1', \mathbf{y}_2')$ ; we refer to  $P(\mathbf{y}_1', \mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  as a ‘transition probability’, and we use  $\rho(\mathbf{y}_1, \mathbf{y}_2)$  to denote the density of code vectors of the second stage of vector quantisation. Note that  $\rho(\mathbf{y}_1, \mathbf{y}_2)$  contains no information about correlations between the

positions of the code vectors; it models the average properties of an ensemble of optimised second-stage vector quantisers, assuming that each has a large codebook.

We may write down an integral equation that relates  $P(\mathbf{y}'|\mathbf{y})$  to  $\rho(\mathbf{y})$  as

$$P(\mathbf{y}'|\mathbf{y}) \delta\mathbf{y}' = \left( 1 - \int_{\|\mathbf{w}-\mathbf{y}\| \leq \|\mathbf{y}'-\mathbf{y}\|} d\mathbf{w} P(\mathbf{w}|\mathbf{y}) \right) \rho(y') \delta\mathbf{y}' \quad (\text{A1})$$

The first term on the right-hand side of Equation A1 is the probability that there is no nearest neighbour code vector within the sphere of radius  $\|\mathbf{y}' - \mathbf{y}\|$  centred on  $\mathbf{y}$ , and the second term is the probability of finding a code vector in the volume  $\delta\mathbf{y}'$  located at  $\mathbf{y}'$ . The product of these two terms gives the probability of finding the

nearest neighbour code vector in the volume  $\delta\mathbf{y}'$  at  $\mathbf{y}'$ .

We now solve Equation A1 for the case  $\rho(\mathbf{y}) = \rho_0 = \text{constant}$ . The nearest neighbour code vector is then equally likely to lie in any direction from  $\mathbf{y}$ , so  $P(\mathbf{y}'|\mathbf{y})$  must be a radial distribution function, which we shall denote as  $P(\|\mathbf{y}' - \mathbf{y}\|)$ , which depends only on radial distance  $\|\mathbf{y}' - \mathbf{y}\|$ , and which therefore satisfies the integral equation

$$P(\|\mathbf{y}' - \mathbf{y}\|) = \left( 1 - \int_{\|\mathbf{w}-\mathbf{y}\| \leq \|\mathbf{y}'-\mathbf{y}\|} d\mathbf{w} P(\|\mathbf{w}-\mathbf{y}\|) \right) \rho_0 \quad (\text{A2})$$

The integrand is spherically symmetrical, and so we may use the transformation

---


$$\int_{\|\mathbf{w}-\mathbf{y}\| \leq \|\mathbf{y}'-\mathbf{y}\|} d\mathbf{w} P(\|\mathbf{w}-\mathbf{y}\|) = \alpha_N \int_{\|\mathbf{w}\| \leq \|\mathbf{y}'-\mathbf{y}\|} d\|\mathbf{w}\| \|\mathbf{w}\|^{N-1} P(\|\mathbf{w}\|) \quad (\text{A3})$$


---

where  $\alpha_N$  is a constant deriving from the angular integration in  $N$  dimensions ( $\alpha_2 = 2\pi$  for  $N = 2$ ). Now differentiate Equation A2 with respect to the upper limit  $\|\mathbf{y}' - \mathbf{y}\|$  of the  $\|\mathbf{w}\|$  integration to yield

$$\frac{dP(\|\mathbf{y}' - \mathbf{y}\|)}{d\|\mathbf{y}' - \mathbf{y}\|} = -\alpha_N \rho_0 \|\mathbf{y}' - \mathbf{y}\|^{N-1} P(\|\mathbf{y}' - \mathbf{y}\|) \quad (\text{A4})$$

and finally integrate to yield

$$P(\|\mathbf{y}' - \mathbf{y}\|) = P_0 \exp\left(-\frac{\alpha_N \rho_0 \|\mathbf{y}' - \mathbf{y}\|^N}{N}\right) \quad (\text{A5})$$

where we must adjust the integration constant  $P_0$  in order

to normalise  $P(\|\mathbf{y}' - \mathbf{y}\|)$ . The  $N = 2$  case reduces to a Gaussian distribution with  $P_0 = \rho_0$ .

We now extend the previous results to the case

$$\rho(y') \approx \rho(\mathbf{y}) + (\mathbf{y}' - \mathbf{y})^T \cdot \nabla \rho(\mathbf{y}) \quad (\text{A6})$$

which is a first-order Taylor expansion of  $\rho(y')$  about the point  $\mathbf{y}' = \mathbf{y}$ . Throughout this section we use approximation signs to indicate where we omit second- (and higher)-order terms in Taylor expansions. We anticipate that the first-order expansion of  $P(\mathbf{y}'|\mathbf{y})$  has the form of Equation A5, with an extra factor to account for the angular dependence in Equation A6. Thus we approximate  $P(\mathbf{y}'|\mathbf{y})$  as

---


$$P(\mathbf{y}'|\mathbf{y}) \approx P_0(\mathbf{y}) \left( 1 + (\mathbf{y}' - \mathbf{y})^T \cdot \mathbf{a}(\mathbf{y}) \right) \exp\left(-\frac{\alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^N}{N}\right) \quad (\text{A7})$$

where we have yet to determine  $\mathbf{a}(\mathbf{y})$ .

To solve for  $P(\mathbf{y}'|\mathbf{y})$  we must first differentiate Equation A1 with respect to  $\mathbf{y}'$  to obtain

---


$$\frac{\partial P(\mathbf{y}'|\mathbf{y})}{\partial \mathbf{y}'} = - \left( \frac{\partial}{\partial \mathbf{y}'} \int_{\|\mathbf{w}-\mathbf{y}\| \leq \|\mathbf{y}'-\mathbf{y}\|} d\mathbf{w} P(\mathbf{w}|\mathbf{y}) \right) \rho(y') + \frac{P(\mathbf{y}'|\mathbf{y})}{\rho(y')} \frac{\partial \rho(y')}{\partial \mathbf{y}'} \quad (\text{A8})$$


---

The first term depends on the derivative with respect to the upper limit of the  $\mathbf{w}$  integral, and the second term depends directly on the gradient of the code vector den-

sity.

To simplify the three terms that appear in Equation A8, we need the following three results:

$$\begin{aligned} \frac{\partial P(\mathbf{y}'|\mathbf{y})}{\partial \mathbf{y}'} &\approx \frac{\partial}{\partial \mathbf{y}'} \left( P_0(\mathbf{y}) \left( 1 + (\mathbf{y}' - \mathbf{y})^T \cdot \mathbf{a}(\mathbf{y}) \right) \exp \left( -\frac{\alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^N}{N} \right) \right) \\ &= P_0(\mathbf{y}) \left( \mathbf{a}(\mathbf{y}) - \alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^{N-2} (\mathbf{y}' - \mathbf{y}) \left( 1 + (\mathbf{y}' - \mathbf{y})^T \cdot \mathbf{a}(\mathbf{y}) \right) \right) \\ &\quad \times \exp \left( -\frac{\alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^N}{N} \right) \end{aligned} \quad (\text{A9})$$

$$\begin{aligned} \frac{\partial}{\partial \mathbf{y}'} \int_{\|\mathbf{w}-\mathbf{y}\| \leq \|\mathbf{y}'-\mathbf{y}\|} d\mathbf{w} P(\mathbf{w}|\mathbf{y}) &= \frac{\mathbf{y}' - \mathbf{y}}{\|\mathbf{y}' - \mathbf{y}\|} \oint_{\|\mathbf{w}-\mathbf{y}\| \leq \|\mathbf{y}'-\mathbf{y}\|} dS P(\mathbf{w}|\mathbf{y}) \\ &\approx P_0(\mathbf{y}) \alpha_N \|\mathbf{y}' - \mathbf{y}\|^{N-2} (\mathbf{y}' - \mathbf{y}) \exp \left( -\frac{\alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^N}{N} \right) \\ \frac{\partial \rho(y')}{\partial \mathbf{y}'} &= \nabla \rho(y') \end{aligned} \quad (\text{A10})$$

We use these three results to simplify Equation A8 into the form

$$\begin{aligned} \mathbf{a}(\mathbf{y}) - \alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^{N-2} (\mathbf{y}' - \mathbf{y}) \left( 1 + (\mathbf{y}' - \mathbf{y})^T \cdot \mathbf{a}(\mathbf{y}) \right) &\approx \\ - \alpha_N \|\mathbf{y}' - \mathbf{y}\|^{N-2} (\mathbf{y}' - \mathbf{y}) + \left( 1 + (\mathbf{y}' - \mathbf{y})^T \cdot \mathbf{a}(\mathbf{y}) \right) \frac{\nabla \rho(y')}{\rho(y')} & \end{aligned} \quad (\text{A11})$$

We then use  $\rho(y') \approx \rho(\mathbf{y})(1 + (\mathbf{y}' - \mathbf{y})^T \cdot \frac{\nabla \rho(\mathbf{y})}{\rho(\mathbf{y})})$  and  $\frac{\nabla \rho(y')}{\rho(y')} \approx \frac{\nabla \rho(\mathbf{y})}{\rho(\mathbf{y})}$  to simplify this result into the form

$$\left( \mathbf{a}(\mathbf{y}) - \frac{\nabla \rho(\mathbf{y})}{\rho(\mathbf{y})} \right) \approx \alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^{N-2} (\mathbf{y}' - \mathbf{y}) (\mathbf{y}' - \mathbf{y})^T \cdot \left( \mathbf{a}(\mathbf{y}) - \frac{\nabla \rho(\mathbf{y})}{\rho(\mathbf{y})} \right) \quad (\text{A12})$$

where we omit second (and higher)-order terms.

Finally, we solve this equation by choosing  $\mathbf{a}(\mathbf{y}) = \frac{\nabla \rho(\mathbf{y})}{\rho(\mathbf{y})}$ , and so we obtain the generalisation of Equation A5 as

$$\begin{aligned} P(\mathbf{y}'|\mathbf{y}) &\approx P_0(\mathbf{y}) \left( 1 + \frac{(\mathbf{y}' - \mathbf{y})^T \cdot \nabla \rho(\mathbf{y})}{\rho(\mathbf{y})} \right) \exp \left( -\frac{\alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^N}{N} \right) \\ &\propto \rho(y') \exp \left( -\frac{\alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^N}{N} \right) \end{aligned} \quad (\text{A13})$$

- [1] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [2] H Kumazawa, M Kasahara, and T Namekawa, *A construction of vector quantisers for noisy channels*, Electronics and Engineering in Japan **67** (1984), no. 4, 39–47.
- [3] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [4] S P Luttrell, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
- [5] ———, *Derivation of a class of training algorithms*, IEEE Transactions on Neural Networks **1** (1990), no. 2, 229–232.
- [6] ———, *Code vector density in topographic mappings: scalar case*, IEEE Transactions on Neural Networks **2** (1991), no. 4, 427–436.
- [7] ———, *A hierarchical network for clutter and texture modelling*, Proceedings of SPIE conference on adaptive signal processing (San Diego), SPIE, 1991, pp. 518–528.
- [8] ———, *Self-supervised training of hierarchical vector quantisers*, Proceedings of IEE conference on artificial neural networks (Bournemouth), IEE, 1991, pp. 5–9.
- [9] P L Zador, *Asymptotic quantisation error of continuous signals and the quantisation dimension*, IEEE Transactions on Information Theory **28** (1982), no. 2, 139–149.



# Partitioned Mixture Distribution: An Adaptive Bayesian Network for Low-Level Image Processing \*

S. P. Luttrell

*Adaptive Systems Theory Section, Defence Research Agency,  
St Andrews Rd, Malvern, Worcestershire WR14 3PS, United Kingdom*

Bayesian methods are used to analyse the problem of training a model to make predictions about the probability distribution of data that has yet to be received. Mixture distributions emerge naturally from this framework, but are not ideally matched to the density estimation problems that arise in image processing. An extension, called a partitioned mixture distribution is presented, which is essentially a set of overlapping mixture distributions. An expectation-maximisation training algorithm is derived for optimising partitioned mixture distributions according to the maximum likelihood prescription. Finally, the results of some numerical simulations are presented, which demonstrate that lateral inhibition arises naturally in partitioned mixture distributions, and that the nodes in a partitioned mixture distribution network co-operate in such a way that each mixture distribution in the partitioned mixture distribution receives its necessary complement of computing machinery.

## I. INTRODUCTION

This paper is an expanded version of a previous presentation [5], and its purpose is to introduce a novel adaptive network for performing low-level image processing operations. Unlike conventional solutions to this problem in which image operators are hand crafted to extract information from images, the adaptive approach taken in this paper discovers for itself how best to process images. Furthermore, to ensure that the operation of the image processing network is principled, rather than *ad hoc*, it is derived from Bayesian techniques of information processing, in which probability density functions (PDFs) are used to represent information (for example, see [1]).

Mixture distribution (MD) models emerge very naturally in the Bayesian approach, see [6], and also Section II below. However, in low-level image processing applications each patch of an image requires a separate MD to model its marginal PDF, which leads to a rather clumsy processing architecture. It is better to embed the multiple MD models into a single unified architecture, called a partitioned mixture distribution (PMD) [4]. This type of architecture corresponds to a locally connected neural network, as will be explained. Also, several numerical experiments will be presented to demonstrate some of the more interesting properties of PMDs.

## II. BAYESIAN APPROACH

In this Section the use of PDFs as a vehicle for solving inference problems is discussed. The Bayesian approach

to building a PDF model is characterised by several standard steps that will be described below. Note that the  $Q(\dots)$  notation is used generically to denote ‘the PDF of  $\dots$ ’, and should not be taken to imply that  $Q(\dots)$  is a fixed function of its arguments. The notation will be made more explicit only where its meaning is not clear from the context.

### A. Step 1: Select a state space

Choose a state space to describe the system under study, and denote a vector in this space as  $\mathbf{x}$ . For instance, in the case of image processing, the state space might be the space of all images, in which case the components of  $\mathbf{x}$  would be the pixel values of a single image.

### B. Step 2: Select a PDF model

Define a PDF  $Q(\mathbf{x})$  to model the joint probability density of the components of the vector  $\mathbf{x}$ .  $Q(\mathbf{x})$  will have one or more internal variables (or parameters), which may be represented as a vector  $\mathbf{s}$  which is unknown at the outset, so  $Q(\mathbf{x})$  may be expanded out as follows

$$Q(\mathbf{x}) = \int d\mathbf{s} Q(\mathbf{x}|\mathbf{s}) Q(\mathbf{s}) \quad (2.1)$$

where  $Q(\mathbf{x}|\mathbf{s})$  is the PDF given that  $\mathbf{s}$  is known, and  $Q(\mathbf{s})$  is the prior PDF of  $\mathbf{s}$ . Thus the PDF  $Q(\mathbf{x})$  is a weighted average over the various alternative contributing PDFs  $Q(\mathbf{x}|\mathbf{s})$ .

### C. Step 3: Select a training set

Define a training set  $\mathbf{X}$  of vectors drawn from the state space of the system. If there are  $M$  such vectors then  $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$ . If the training samples are

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 22, 2019.

This paper appeared in IEE Proceedings on Vision, Image, and Signal Processing, 1994, vol. 141, no. 4, pp. 251-260. Paper 1316K (E4, C4), first received 29th October 1993 and in revised form 9th May 1994. © Crown copyright, 1994.

drawn independently, then the likelihood of the training set  $\mathbf{X}$  given a particular choice of the parameter vector  $\mathbf{s}$  is given by

$$Q(\mathbf{X}|\mathbf{s}) = \prod_{i=1}^M Q(\mathbf{x}_i|\mathbf{s}) \quad (2.2)$$

and the logarithmic likelihood is given by

$$\begin{aligned} L(\mathbf{X}|\mathbf{s}) &\equiv \log Q(\mathbf{X}|\mathbf{s}) \\ &= \sum_{i=1}^M \log Q(\mathbf{x}_i|\mathbf{s}) \end{aligned} \quad (2.3)$$

#### D. Step 4: Train the PDF model

Present the training data  $\mathbf{X}$  to the model  $Q(\mathbf{x})$  in order that the model can learn something about the true state of its internal parameter vector  $\mathbf{s}$ . The overall effect of this is to convert the prior PDF  $Q(\mathbf{s})$  into a posterior PDF  $Q(\mathbf{s}|\mathbf{X})$  that embodies the prior information contained in  $Q(\mathbf{s})$  plus the additional information contained in  $\mathbf{X}$ . This conversion is implemented using Bayes' theorem thus

$$Q(\mathbf{s}|\mathbf{X}) = \frac{Q(\mathbf{X}|\mathbf{s}) Q(\mathbf{s})}{\int d\mathbf{s}' Q(\mathbf{X}|\mathbf{s}') Q(\mathbf{s}')} \quad (2.4)$$

The training process converts  $Q(\mathbf{x})$  into  $Q(\mathbf{x}|\mathbf{X})$  thus

$$Q(\mathbf{x}|\mathbf{X}) = \int d\mathbf{s} Q(\mathbf{x}|\mathbf{s}) Q(\mathbf{s}|\mathbf{X}) \quad (2.5)$$

which is a weighted average over the possible contributing PDFs  $Q(\mathbf{x}|\mathbf{s})$ , with the untrained prior PDF  $Q(\mathbf{s})$  replaced by the trained posterior PDF  $Q(\mathbf{s}|\mathbf{X})$ . This result should be compared with Equation 2.1.

#### E. Step 5: Approximate the PDF model

$Q(\mathbf{x}|\mathbf{X})$  is usually difficult to calculate because it requires an integration over alternative states of the parameter vector  $\mathbf{s}$ . There are many ways in which approximations to this integral can be made; the optimal choice depends both on the form of the integrand and on the type of problem in which the integral appeared in the first place. However, if the integral is dominated by contributions from the integrand in the vicinity of a single value  $\mathbf{s}(\mathbf{X})$  of  $\mathbf{s}$ , then  $Q(\mathbf{s}|\mathbf{X})$  can be approximated by a point PDF  $\delta[\mathbf{s} - \mathbf{s}(\mathbf{X})]$  located at  $\mathbf{s}(\mathbf{X})$ . This simplifies  $Q(\mathbf{x}|\mathbf{X})$  as follows

$$\begin{aligned} Q(\mathbf{x}|\mathbf{X}) &\approx \int d\mathbf{s} Q(\mathbf{x}|\mathbf{s}) \delta[\mathbf{s} - \mathbf{s}(\mathbf{X})] \\ &= Q[\mathbf{x}|\mathbf{s}(\mathbf{X})] \end{aligned} \quad (2.6)$$

The optimum location  $\mathbf{s}(\mathbf{X})$  of this point PDF  $\delta[\mathbf{s} - \mathbf{s}(\mathbf{X})]$  is debatable. Two popular schemes are

$$\mathbf{s}(\mathbf{X}) = \begin{cases} \arg \max_{\mathbf{s}} Q(\mathbf{X}|\mathbf{s}) & \text{maximum likelihood} \\ \arg \max_{\mathbf{s}} Q(\mathbf{s}|\mathbf{X}) & \text{maximum posterior probability} \end{cases} \quad (2.7)$$

Note that the maximum likelihood prescription can be written out explicitly as

$$\mathbf{s}(\mathbf{X}) = \arg \max_{\mathbf{s}} \sum_{i=1}^M \log Q(\mathbf{x}_i|\mathbf{s}) \quad (2.8)$$

If the point PDF approximation is replaced by a weighted sum of point PDFs then Equation 2.6 becomes

$$\begin{aligned} Q(\mathbf{x}|\mathbf{X}) &\approx \int d\mathbf{s} Q(\mathbf{x}|\mathbf{s}) \sum_{c=1}^n Q(c|\mathbf{X}) \delta[\mathbf{s} - \mathbf{s}_c(\mathbf{X})] \\ &= \sum_{c=1}^n Q[\mathbf{x}|\mathbf{s}_c(\mathbf{X})] Q(c|\mathbf{X}) \end{aligned} \quad (2.9)$$

where the  $Q(c|\mathbf{X})$  are normalised as  $\sum Q(c|\mathbf{X}) = 1$ . This type of approximation is called a MD because the PDF  $Q(\mathbf{x}|\mathbf{X})$  is approximated by a weighted mixture of  $n$  separate PDFs  $Q[\mathbf{x}|\mathbf{s}_c(\mathbf{X})]$ . The optimal choice of  $\mathbf{s}_c(\mathbf{X})$  and  $Q(c|\mathbf{X})$  depends on what types of error can be tolerated in the MD approximation to  $Q(\mathbf{x}|\mathbf{X})$ . In this paper the maximum likelihood prescription will be used.

In the usual terminology of density estimation theory  $c$  is usually referred to as a class label, in which case  $Q(c|\mathbf{X})$  is the prior probability of class  $c$ , and  $Q[\mathbf{x}|\mathbf{s}_c(\mathbf{X})]$  is the likelihood that data vector  $\mathbf{x}$  could be produced by class  $c$ .

### III. ADAPTIVE MIXTURE DISTRIBUTION NETWORKS

In this Section MD networks will be discussed in much greater detail, and the generalisation to PMDs will be presented.

#### A. Neural network interpretation of a mixture distribution

A MD can be interpreted in ‘neural network’ language as follows. The parameters  $\mathbf{s}_c(\mathbf{X})$  and  $Q(c|\mathbf{X})$  correspond to the neural network weights (or whatever) that emerge from the maximum likelihood training process. Subsequent computation of the MD approximation to  $Q(\mathbf{x}|\mathbf{X})$  corresponds to what is commonly called ‘testing’ the neural network. This correspondence between PDF models and neural networks will be tacitly assumed throughout this paper.

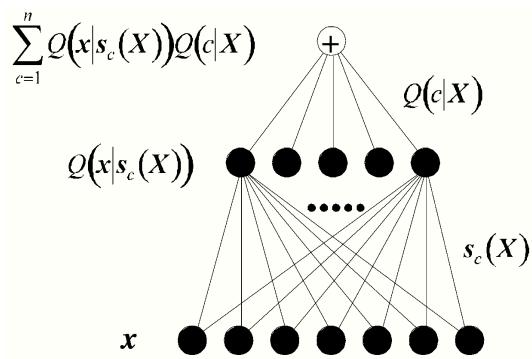


Figure 1: MD represented as a neural network. Bottom layer represents the components of the input vector  $\mathbf{x}$ . Middle layer represents  $Q[\mathbf{x}|\mathbf{s}_c(\mathbf{X})]$  for each possible value of the class label  $c$ ; these are the class likelihoods. Top layer is the network output, which is a linear combination (or mixture) of the various  $Q[\mathbf{x}|\mathbf{s}_c(\mathbf{X})]$  weighted by the class prior probabilities  $Q(c|\mathbf{X})$ . The connections between the middle and output layers have weights that correspond to the  $Q(c|\mathbf{X})$ . Connections between input and middle layers indicate that each  $Q[\mathbf{x}|\mathbf{s}_c(\mathbf{X})]$  depends on all of the components of the input vector  $\mathbf{x}$ ; they are not simply weighted connections.

In Figure 1 a MD is drawn as if it were a neural network. The input layer is transformed through a set of nonlinear functions to produce a pattern of hidden layer outputs  $Q[\mathbf{x}|\mathbf{s}_c(\mathbf{X})]$  indexed by the label  $c$ , which are then transformed via a linear summation operation weighted by the prior probabilities  $Q(c|\mathbf{X})$  to produce the MD  $\sum Q[\mathbf{x}|\mathbf{s}_c(\mathbf{X})] Q(c|\mathbf{X})$ . The goal is to maximise the likelihood of the training data with respect to  $\mathbf{s}_c(\mathbf{X})$  and  $Q(c|\mathbf{X})$  thus

$$\arg \max_{\{\mathbf{s}_c(\mathbf{X}), Q(c|\mathbf{X}) : c = 1, 2, \dots, n\}} \sum_{i=1}^M \log \left[ \sum_{c'=1}^n Q[\mathbf{x}_i|\mathbf{s}_{c'}] Q(c') \right] \quad (3.1)$$

which maximises the average over the  $M$  samples in the training set of the logarithm of the network output. This is an example of an unsupervised neural network training procedure.

#### B. Multiple independent mixture distributions

In situations where the dimensionality,  $\dim \mathbf{x}$ , of the input vector,  $\mathbf{x}$ , is large it is necessary to make some simplifying assumptions in order to use the MD approach. For instance, an independence assumption can be made where  $\mathbf{x}$  is partitioned into  $N$  subspaces as  $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^N)$ , and likewise the training data  $\mathbf{X} = (\mathbf{X}^1, \mathbf{X}^2, \dots, \mathbf{X}^N)$ , and a MD  $Q^k(\mathbf{x}^k|\mathbf{X}^k)$  is constructed independently for each subspace. The over-

all PDF  $Q(\mathbf{x}|\mathbf{X})$  is then the product of the individual  $Q^k(\mathbf{x}^k|\mathbf{X}^k)$ .

$$Q(\mathbf{x}|\mathbf{X}) = \prod_{k=1}^N Q^k(\mathbf{x}^k|\mathbf{X}^k) \quad (3.2)$$

This expression does not model any correlations that might exist between the various  $\mathbf{x}^k$  for different values of  $k$ .

Each  $Q^k(\mathbf{x}^k|\mathbf{X}^k)$  in this product could be approximated using a MD network of the type shown in Figure 1, and the outputs from the  $N$  such networks could be combined using a multiplier node to create the following

overall approximation to  $Q(\mathbf{x}|\mathbf{X})$

$$Q(\mathbf{x}|\mathbf{X}) \approx \prod_{k=1}^N \left\{ \sum_{c^k=1}^{n^k} Q[x^k|s_{c^k}(X^k)] Q(c^k|X^k) \right\} \quad (3.3)$$

where an obvious generalisation of the earlier notation has been used. The maximum likelihood prescription then leads to  $N$  independent optimisation problems of the form given in Equation 3.1.

For  $N = 2$  the neural network representation of this approximation is shown in Figure 2. This network can readily be generalised to arbitrary values of  $N$ .

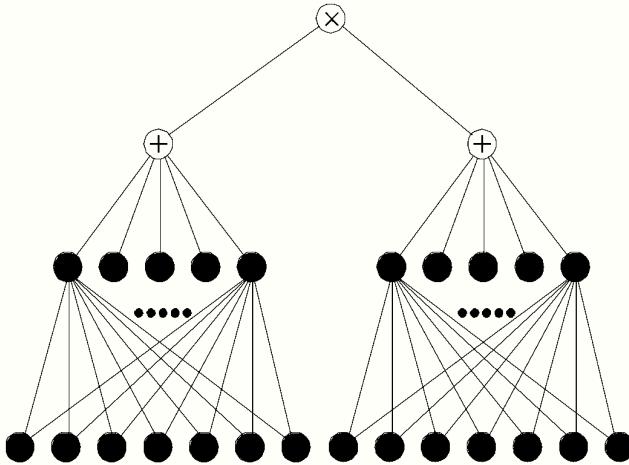


Figure 2: A pair of independent MD networks. The final output is the product of the output of the pair of mixture distribution networks. These networks compute a pair of mixture distributions for the left and right halves of the overall input vector. The final network output is then the product of the outputs of the left and right half networks; this procedure assumes that the left and right halves of the input vector are independent, or, at least, that any correlations between left and right may be ignored.

### C. Multiple overlapping mixture distributions

The type of network shown in Figure 2 is undesirable for data which does not clearly divide into a number of separate subspaces. What is required is a network structure that treats each patch of data on an equal footing, unlike Figure 2 which partitions the input vector  $\mathbf{x}$  into left and right halves.

In the following subsections a two stage derivation of such a network will be presented, starting with a standard MD network, and finishing with what will be called a partitioned mixture distribution (PMD) network [4]. For concreteness, this derivation will refer to the network in Figure 2, but the technique may be applied to arbitrary  $N$ .

### 1. Modify the mixture distribution network

In the first stage of the derivation the structure of the network is changed so as to make it possible to construct translation invariant networks when  $N$  is large. The basic trick is to modify the MD network as shown in Figure 3.

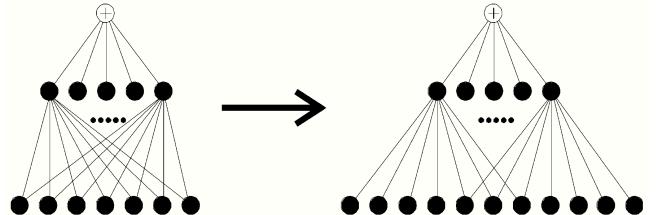


Figure 3: Modification of the basic MD network. In the standard MD each mixture component (or class) sees the whole of the input space, whereas in the modified version each class sees only part of the input space, as indicated by the connections drawn in the right hand diagram. This has the effect that as the class label is varied the field of view of the corresponding mixture component moves across the input data.

In the modified MD the different components of the input vector  $\mathbf{x}$  receive a different degree of attention by the MD. For instance, the components at the far left and far right of the diagram in Figure 3 are each seen by only one component of the MD, whereas the central three components of  $\mathbf{x}$  are seen by all five components of the MD. Therefore, the scope for tailoring the shape of the MD  $Q(\mathbf{x}|\mathbf{X})$  is such that the PDF of the central components of the input vector is better modelled than those at the edges. This limitation is not present in the standard MD.

With this modification, the network in Figure 2 would become as shown in Figure 4.

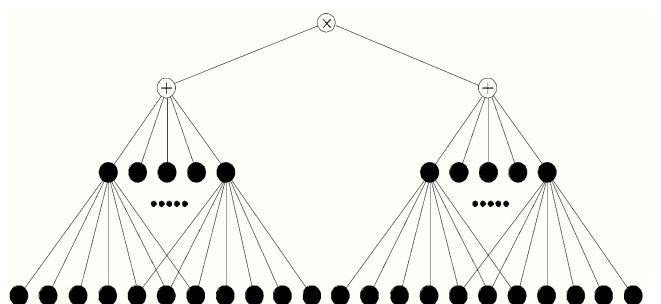


Figure 4: A pair of independent MD networks. This is as in Figure 2, but using the modified MD network shown in Figure 3.

### 2. Multiple overlapping mixture distribution networks

In the second stage of the derivation a very high dimensional input vector is considered. A network of the

type shown in Figure 3 can see only a subspace of such an input vector, but different networks could be used to view different subspaces, such as in Figure 4. It is convenient to build a composite network that implements simultaneously all possible placements of Figure 3 with respect to the input vector. This is called a partitioned mixture distribution (PMD), and its corresponding network is shown in Figure 5.

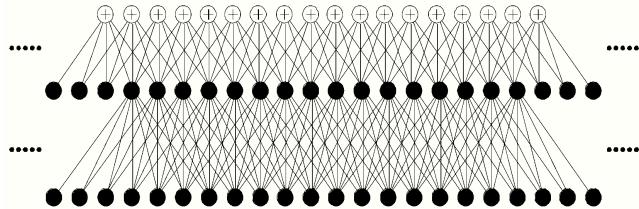


Figure 5: A union of MD networks of the type shown in Figure 4. Because the input vector is assumed to be high-dimensional, the dots on either side of the diagram indicate that the basic structure is to be replicated across the input vector. This network will be referred to as a partitioned mixture distribution (PMD) network.

If a network of the type shown in Figure 4 is required, then it may be obtained by taking the product of the appropriate output nodes of the network shown in Figure 5. It is important to note that because of the overlap between the various subnetworks in Figure 5 their parameter vectors are not independent. In the neural network literature this is usually referred to as ‘weight sharing’.

The PMD network in Figure 5 can be extended later-

ally without encountering any scaling problems because its connectivity is local, and it computes as many MDs as there are network outputs. Any non-overlapping subset of these MDs may be used to construct an approximation to  $Q(\mathbf{x}|\mathbf{X})$  as in Equation 3.3, so there are many different possible approximate versions of  $Q(\mathbf{x}|\mathbf{X})$ . The PMD network thus provides the raw material for constructing a PDF model; although it does not specify a unique way in which to combine its MDs.

The purpose of this paper is to investigate the properties of the PMD network in Figure 5, rather than the PDF model(s) that might eventually be constructed from it, so the exact form of the PDF model will not be discussed further.

#### IV. THEORY OF PARTITIONED MIXTURE DISTRIBUTIONS

In this Section the theory of PMDs will be presented, and an expectation-maximisation (EM) method of training a PMD will be derived. Finally, these results will be applied to the special case of a Gaussian PMD. The diagram in Figure 5 shows the network structure that is implicitly used throughout this Section.

##### A. Basic theory of partitioned mixture distributions

For a PMD the basic definition of a MD (see Equation 2.9) becomes

$$\begin{aligned} Q_c(\mathbf{x}|\mathbf{X}) &= \sum_{c'=c-n/2}^{c+n/2} Q(\mathbf{x}, c'|\mathbf{X}) \\ &= \frac{\sum_{c'=c-n/2}^{c+n/2} Q(\mathbf{x}|\mathbf{s}_{c'}(\mathbf{X})) Q(c'|\mathbf{X})}{\sum_{c''=c-n/2}^{c+n/2} Q(c''|\mathbf{X})} \quad c = 1, 2, \dots, N \end{aligned} \quad (4.1)$$

where edge effects are ignored.  $N$  is the number of nodes across the diagram in Figure 5, the index  $c$  selects which of the  $N$  MDs is the focus of attention, the halfwidth parameter  $\frac{n}{2}$  is used to determine the number of components in each MD ( $n$  is called the ‘mixture window width’, and if  $n$  is odd then  $\frac{n}{2}$  is replaced by the largest integer less than  $\frac{n}{2}$ ), and the normalisation factor  $\sum_{c'=c-n/2}^{c+n/2} Q(c'|\mathbf{X})$  ensures that each MD is normalised so that its integral over  $\mathbf{x}$  is unity. Because the limited mixture window size ensures that each MD uses only part of the full range of possible values of the class

label, the set of overlapping MDs will be referred to as a partitioned mixture distribution (PMD). In Figure 5 each class is sensitive to only a limited number of the inputs, which will be called the ‘input window width’.

For convenience, an augmented set  $\mathcal{S}$  of parameters will be defined as

$$\mathcal{S} = \{\mathbf{s}_c, Q(c) : c = 1, 2, \dots, N\} \quad (4.2)$$

and the notation  $\mathcal{S}(\mathbf{X})$  will be used to denote a single choice of  $\mathcal{S}$  (such as the maximum likelihood choice) given the training set  $\mathbf{X}$ . The maximum likelihood pro-

cedure for optimising  $\mathbf{S}$  for a set of  $N$  independent MDs must now be generalised to a form that is appropriate for Figure 5. The simplest choice is to require that the geometric mean of the likelihoods (i.e., sum of the logarithmic likelihoods) of the various MDs in Figure 5 be maximised,

$$\mathbf{S}(\mathbf{X}) = \arg \max_{\mathbf{S}} \sum_{c=1}^N \sum_{i=1}^M \log Q_c(\mathbf{x}_i | \mathbf{S}) \quad (4.3)$$

where

$$Q_c(\mathbf{x}_i | \mathbf{S}) = \frac{\sum_{c'=c-n/2}^{c+n/2} Q(\mathbf{x}_i | \mathbf{s}_{c'}) Q(c')}{\sum_{c''=c-n/2}^{c+n/2} Q(c'')} \quad (4.4)$$

This result should be compared with Equation 2.8. The various parameter vectors must now be adjusted in order to maximise the average logarithmic likelihood. In the limit of a large training set size Equation 4.3 is equivalent to maximising the average logarithmic likelihood.

$$\mathbf{S}(\mathbf{X}) = \arg \max_{\mathbf{S}} \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log Q_c(\mathbf{x} | \mathbf{S}) \quad (4.5)$$

where the average over training data has been replaced by a shorthand notation in which the PDF  $P(\mathbf{x})$  is used

to represent the density of training vectors, as follows

$$\frac{1}{M} \sum_{i=1}^M (\dots) \longrightarrow \int d\mathbf{x} P(\mathbf{x}) (\dots) \quad (4.6)$$

In PMD calculations it is useful to write the joint probability for the MD centred on class  $c$  as

$$Q_c(\mathbf{x}, c' | \mathbf{S}) = \begin{cases} \frac{Q(\mathbf{x} | \mathbf{s}_{c'}) Q(c')}{\sum_{c''=c-n/2}^{c+n/2} Q(c'')} & |c' - c| \leq \frac{n}{2} \\ 0 & |c' - c| > \frac{n}{2} \end{cases} \quad (4.7)$$

and the corresponding posterior probability as

$$Q_c(c' | \mathbf{x}, \mathbf{S}) = \begin{cases} \frac{Q(\mathbf{x} | \mathbf{s}_{c'}) Q(c')}{\sum_{c''=c-n/2}^{c+n/2} Q(\mathbf{x} | \mathbf{s}_{c''}) Q(c'')} & |c' - c| \leq \frac{n}{2} \\ 0 & |c' - c| > \frac{n}{2} \end{cases} \quad (4.8)$$

which satisfies the normalisation condition  $\sum_{c'=c-n/2}^{c+n/2} Q_c(c' | \mathbf{x}, \mathbf{S}) = 1$ . In a PMD there are  $N$  separate MDs, each with its own set of posterior probabilities, which occupy class labels  $c - \frac{n}{2}, c - \frac{n}{2} + 1, \dots, c + \frac{n}{2}$  for the  $c$ th MD. These posterior probabilities overlap each other, and it turns out to be useful to define the sum over posterior probabilities at each class label

---


$$\begin{aligned} \tilde{Q}(c | \mathbf{x}, \mathbf{S}) &\equiv \sum_{c'=c-n/2}^{c+n/2} Q_{c'}(c | \mathbf{x}, \mathbf{S}) \\ &= Q(\mathbf{x} | \mathbf{s}_c) Q(c) \sum_{c'=c-n/2}^{c+n/2} \frac{1}{\sum_{c''=c'-n/2}^{c'+n/2} Q(\mathbf{x} | \mathbf{s}_{c''}) Q(c'')} \end{aligned} \quad (4.9)$$


---

This definition implies the normalisation condition  $\sum_{c=1}^N \tilde{Q}(c | \mathbf{x}, \mathbf{S}) = N$ , so it could be used as a simple way of converting any function of  $c$  into a normalised function. Normalisation to  $N$ , rather than 1, is used to ensure that the overall scale of  $\tilde{Q}(c | \mathbf{x}, \mathbf{S})$  does not depend on the dimensionality  $N$  of the input data. A similar definition for the sum of the prior probabilities also turns out to be useful

$$\tilde{Q}(c | \mathbf{S}) \equiv Q(c) \sum_{c'=c-n/2}^{c+n/2} \frac{1}{\sum_{c''=c'-n/2}^{c'+n/2} Q(c'')} \quad (4.10)$$

which satisfies  $\sum_{c=1}^N \tilde{Q}(c | \mathbf{S}) = 1$ .

## B. Expectation-maximisation (EM) method of training partitioned mixture distributions

The PMD optimisation prescription in Equation 4.3 needs to be converted into an explicit training algorithm. There are two basic types of algorithm: the expectation-maximisation (EM) method (see [2]), and various gradient ascent methods. The former is suitable for batch training, whereas the latter is suitable for online training. In this paper only batch training will be considered, so gradient ascent methods will not be discussed.

The EM method is rather subtle, so a pictorial summary is given in Figure 6.

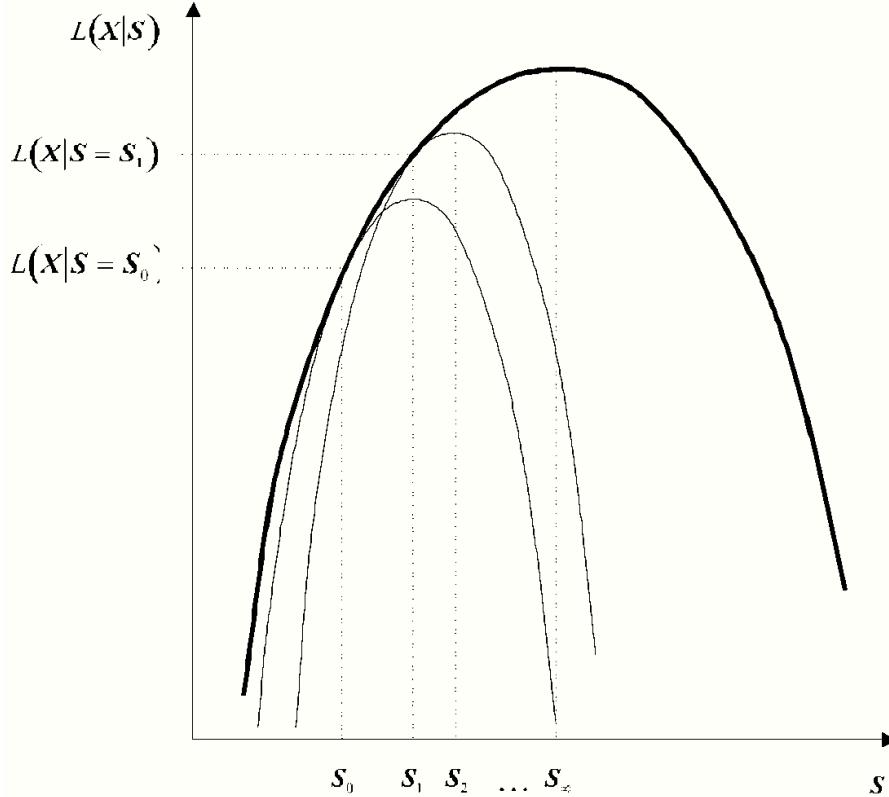


Figure 6: An example of a logarithmic likelihood function  $L(\mathbf{X}|\mathbf{S})$ , shown in bold. Assume that the starting value of the parameter vector is  $\mathbf{S} = \mathbf{S}_0$ , and the goal is to locate the position  $\mathbf{S}_\infty$  of the peak of the  $L(\mathbf{X}|\mathbf{S})$  curve. Each iteration of the EM method has 2 steps: (i) fit a lower bound function beneath the  $L(\mathbf{X}|\mathbf{S})$  curve, which touches it at the point  $\mathbf{S} = \mathbf{S}_0$ , as shown in the diagram, and (ii) locate the position of the peak  $\mathbf{S}_1$  of this lower bound function. Some progress has been made towards the maximum of  $L(\mathbf{X}|\mathbf{S})$ , as can be seen by inspecting the diagram. This procedure is then repeated with the new starting value  $\mathbf{S} = \mathbf{S}_1$ . After several iterations the value of  $\mathbf{S}$  approaches the required location  $\mathbf{S}_\infty$  of the peak of the logarithmic likelihood function  $L(\mathbf{X}|\mathbf{S})$ , as can readily be seen in the diagram. The key is to choose the lower bound curve so that it is easy to locate its maximum; each iteration of the EM algorithm can then readily be implemented.

The main purpose of the following derivation is to obtain a lower bound function as discussed in the caption to Figure 6. First of all define the difference  $\Delta L(\mathbf{S}, \mathbf{S}_0; \mathbf{X})$  between the logarithmic likelihood evaluated for two different choices,  $\mathbf{S} \equiv \{\mathbf{s}_c, Q(c) : c = 1, 2, \dots, N\}$  and  $\mathbf{S}_0 \equiv \{\mathbf{s}_{0,c}, Q_0(c) : c = 1, 2, \dots, N\}$ , of the parameters

$$\begin{aligned}\Delta L(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) &\equiv L(\mathbf{X}|\mathbf{S}) - L(\mathbf{X}|\mathbf{S}_0) \\ &= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log \left[ \frac{Q_c(\mathbf{x}|\mathbf{S})}{Q_c(\mathbf{x}|\mathbf{S}_0)} \right]\end{aligned}\quad (4.11)$$

The steps of the derivation are as follows:

$$\begin{aligned}
\Delta L(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) &= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log \left[ \frac{\sum_{c'=c-n/2}^{c+n/2} Q(\mathbf{x}|\mathbf{s}_{c'}) Q(c')}{Q_c(\mathbf{x}|\mathbf{S}_0) \sum_{c''=c-n/2}^{c+n/2} Q(c'')} \right] \\
&= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log \left[ \sum_{c'=c-n/2}^{c+n/2} Q_c(c'|\mathbf{x}, \mathbf{S}_0) \frac{Q(\mathbf{x}|\mathbf{s}_{c'}) Q(c')}{Q_c(\mathbf{x}, c'|\mathbf{S}_0) \sum_{c''=c-n/2}^{c+n/2} Q(c'')} \right] \\
&\geq \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \sum_{c'=c-n/2}^{c+n/2} Q_c(c'|\mathbf{x}, \mathbf{S}_0) \log \left[ \frac{Q(\mathbf{x}|\mathbf{s}_{c'}) Q(c') \sum_{c'''=c-n/2}^{c+n/2} Q_0(c''')}{Q(\mathbf{x}|\mathbf{s}_{0,c'}) Q_0(c') \sum_{c''=c-n/2}^{c+n/2} Q(c'')} \right] \\
&= \Delta L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X})
\end{aligned} \tag{4.12}$$

where  $\Delta L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X})$  is defined as

$$\begin{aligned}
\Delta L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) &\equiv \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0) \log [Q(\mathbf{x}|\mathbf{s}_c) Q(c)] - \log \left[ \sum_{c'=c-n/2}^{c+n/2} Q(c') \right] \\
&\quad + \text{a term that is independent of } \mathbf{S}
\end{aligned} \tag{4.13}$$

The penultimate step of the derivation in Equation 4.12 was obtained by using Jensen's inequality for convex functions, and the last step was obtained by using the result  $\sum_{c'=c-n/2}^{c+n/2} Q_c(c'|\mathbf{x}, \mathbf{S}) = 1$  and the result

$$\begin{aligned}
\sum_{c=1}^N \sum_{c'=c-n/2}^{c+n/2} Q_c(c'|\mathbf{x}, \mathbf{S}) (\dots) &= \sum_{c'=1}^N \sum_{c=c'-n/2}^{c'+n/2} Q_c(c'|\mathbf{x}, \mathbf{S}) (\dots) \\
&= \sum_{c'=1}^N \tilde{Q}(c'|\mathbf{x}, \mathbf{S}) (\dots)
\end{aligned} \tag{4.14}$$

where the term  $(\dots)$  does not depend on  $c$ , and edge effects are ignored.. Combining the above results yields

$$L(\mathbf{X}|\mathbf{S}) \geq \Delta L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) + L(\mathbf{X}|\mathbf{S}_0) \tag{4.15}$$

where  $\Delta L_0(\mathbf{S}, \mathbf{S}; \mathbf{X}) = 0$ .

$\Delta L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) + L(\mathbf{X}|\mathbf{S}_0)$  satisfies all of the requirements of the lower bound function discussed in the caption to Figure 6, so the first step of the EM algorithm is thus given by

$$\mathbf{S}_1 = \arg \max_{\mathbf{S}} \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \left[ \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0) \log Q(\mathbf{x}, c|\mathbf{S}) - \log \sum_{c'=c-n/2}^{c+n/2} Q(c'|\mathbf{S}) \right] \tag{4.16}$$

where  $\mathbf{S}$  and  $\mathbf{S}_0$  have been carefully used on the right hand side to indicate exactly which terms are varied in the maximisation process (i.e., those containing  $\mathbf{S}$ ). Subsequent iterations of the EM algorithm are analogously defined. Note that each update step is not obliged to update all of the parameters in  $\mathbf{S}$ . For instance, the  $Q(c)$  might be held constant, and only the  $\mathbf{s}_c$  updated, as will be the case in the numerical simulations later.

Provided that the likelihood function has no local maxima, then the EM update equation may be iterated to yield the sequence  $\mathbf{S}_0, \mathbf{S}_1, \dots, \mathbf{S}_\infty$  of successively improved estimates of the required  $\mathbf{S}(\mathbf{X})$ .

### C. Gaussian partitioned mixture distribution

A popular model to use for  $Q(\mathbf{x}|\mathbf{s}_c)$  is a Gaussian PDF with mean  $\mathbf{m}_c$  and covariance  $\mathbf{A}_c$ .

$$\begin{aligned} Q(\mathbf{x}|\mathbf{s}_c) &\longrightarrow Q(x|m_c, \mathbf{A}_c) \\ &\equiv \frac{1}{\sqrt{(\det 2\pi\mathbf{A}_c)}} \exp \left[ -\frac{1}{2}(\mathbf{x} - \mathbf{m}_c)^T \mathbf{A}_c^{-1} (\mathbf{x} - \mathbf{m}_c) \right] \end{aligned} \quad (4.17)$$


---

The likelihoods  $Q_c(\mathbf{x}|\mathbf{S})$ , joint probabilities  $Q_c(\mathbf{x}, c'|\mathbf{S})$ , posterior probabilities  $Q_c(c'|\mathbf{x}, \mathbf{S})$ , and averaged posterior probabilities  $\tilde{Q}(c|\mathbf{x}, \mathbf{S})$  may readily be obtained from  $Q(\mathbf{x}|\mathbf{s}_c)$  and  $Q(c)$  using the theory outlined earlier.

The EM method may be used to solve for the maximum likelihood choice of  $\mathbf{S}$  as follows. Differentiate the right hand side of the EM update equation in Equation 4.16 with respect to  $Q(c|\mathbf{S})$ , set the result to zero, interchange the order of summations assuming that edge effects can be ignored, and use the definition in Equation 4.10 to obtain an implicit equation for the updated  $Q(c|\mathbf{S})$ .

$$\tilde{Q}(c|\mathbf{S}_1) = \int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0) \quad (4.18)$$


---

Similarly, differentiate with respect to  $\mathbf{m}_c$  (assuming that  $\mathbf{A}_c^{-1}$  is full rank) to obtain the updated  $\mathbf{m}_c$

$$\mathbf{m}_c(\mathbf{S}_1) = \frac{\int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0) \mathbf{x}}{\int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0)} \quad (4.19)$$

Similarly, differentiate with respect to  $\mathbf{A}_c^{-1}$  (using  $\frac{\partial \log \det \mathbf{A}_c^{-1}}{\partial \mathbf{A}_c^{-1}} = \mathbf{A}_c^T$ ) to obtain the updated  $\mathbf{A}_c$

$$\mathbf{A}_c(\mathbf{S}_1) = \frac{\int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0) [\mathbf{x} - \mathbf{m}_c(\mathbf{S}_1)] [\mathbf{x} - \mathbf{m}_c(\mathbf{S}_1)]^T}{\int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0)} \quad (4.20)$$


---

where the right hand side depends on the updated  $\mathbf{m}_c$ . In order to simplify notation define the first three moments of the posterior probability weighted training data as

$$\begin{aligned} M_c^0(\mathbf{S}_0) &= \int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0) \\ M_c^1(\mathbf{S}_0) &= \int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0) \mathbf{x} \\ M_c^2(\mathbf{S}_0) &= \int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0) \mathbf{x} \mathbf{x}^T \end{aligned} \quad (4.21)$$

in which case the solutions to the EM update equation may be expressed as

$$\begin{aligned} \tilde{Q}(c|\mathbf{S}_1) &= M_c^0(\mathbf{S}_0) \\ \mathbf{m}_c(\mathbf{S}_1) &= \frac{M_c^1(\mathbf{S}_0)}{M_c^0(\mathbf{S}_0)} \\ \mathbf{A}_c(\mathbf{S}_1) &= \frac{M_c^2(\mathbf{S}_0)}{M_c^0(\mathbf{S}_0)} - \mathbf{m}_c(\mathbf{S}_1) \mathbf{m}_c(\mathbf{S}_1)^T \end{aligned} \quad (4.22)$$

The updated prior probabilities  $Q(c|\mathbf{S}_1)$  may be found by solving the first of these equations iteratively, but the means  $\mathbf{m}_c(\mathbf{S}_1)$  and covariances  $\mathbf{A}_c(\mathbf{S}_1)$  may be obtained directly in closed form using the second and third equations, respectively. In the numerical simulations presented in the following Section the prior probabilities are

fixed to be constant and equal (i.e., they are not updated), and the covariance matrices are assumed to be diagonal.

These results for a Gaussian PMD reduce to the standard results for the corresponding Gaussian MD when the mixture window width  $n$  is wide enough to include the full range of values (i.e.,  $N$ ) of the class label; the PMD is then effectively  $N$  copies of the same MD.

## V. PROPERTIES OF PARTITIONED MIXTURE DISTRIBUTIONS

In this Section a few of the more interesting properties of PMD networks will be demonstrated by numerical simulation. The simulations were carried out mostly on a one-dimensional PMD with  $N = 21$ , an input window width of 7, a mixture window width  $n = 7$ ; the structure of this type of PMD is given in the diagram in Figure 5. The prior probabilities  $Q(c)$  were held constant and equal, and diagonal covariance matrices  $\mathbf{A}_c$  were used.

An interesting variation of the training algorithm (not derived above) was also sometimes used. Thus the averaged posterior probabilities  $\tilde{Q}(c|\mathbf{x}, \mathbf{S})$  were modified as follows

$$\tilde{Q}(c|\mathbf{x}, \mathbf{S}) \longrightarrow (1 - 2\varepsilon) \tilde{Q}(c|\mathbf{x}, \mathbf{S}) + \varepsilon \left[ \tilde{Q}(c-1|\mathbf{x}, \mathbf{S}) + \tilde{Q}(c+1|\mathbf{x}, \mathbf{S}) \right] \quad (5.1)$$

where  $0 \leq \varepsilon \leq 0.5$ . In effect node  $c$  leaks some of its activity onto its neighbours. This leakage leads to a similar type of ordering of the properties of adjacent nodes to that observed in topographic mappings (see [3]). In the limit  $\varepsilon \rightarrow 0$  this reduces to the unmodified training algorithm.

#### A. Lateral inhibition in an iterated partitioned mixture distribution

Lateral inhibition is the term used to describe the tendency for an active node in a neural network to suppress the activity of nearby nodes in the same network layer. This effect can be demonstrated in a PMD with activity leakage when it is driven by a static input vector, and the effect can be greatly enhanced by iterating the PMD, as described below. In an iterated PMD the input data  $\mathbf{x}$  is held constant whilst the prior probabilities  $Q(c|\mathbf{S})$  are temporarily adjusted by an iterative process, as follows. The initial presentation of data  $\mathbf{x}$  causes the PMD to output a set of posterior probabilities  $\tilde{Q}(c|\mathbf{x}, \mathbf{S})$  at each node (exactly as in a standard PMD), which are then used as the prior probabilities  $Q(c|\mathbf{S})$  during the next iteration of the PMD, and so on. This feedback loop causes the peaks and troughs in the posterior probabilities to become exaggerated, which makes the lateral inhibition effect easier to observe when the pattern of numbers  $\{\tilde{Q}(c|\mathbf{x}, \mathbf{S}) : c = 1, 2, \dots, N\}$  is plotted out for successive iterations.<sup>1</sup> For the purpose of this simulation it is not necessary to introduce explicit likelihood functions  $Q(\mathbf{x}|\mathbf{s}_c)$ ; it is sufficient to use input data that is itself the pattern of numbers  $\{Q(\mathbf{x}|\mathbf{s}_c) : c = 1, 2, \dots, N\}$ .

In Figure 7a and b the point stimulus leads to a strong response from the stimulated node, and a suppressed response from the surrounding nodes. This lateral inhibition effect arises from the form of the expression used to define the posterior probability in a PMD. Because the node responses are actually posterior probabilities, if one node has a larger than average share of probability, then its neighbours will typically have a smaller than average share. Note how the effect of node activity leakage is to damp down the response.

In Figure 7c and d the two equal point stimuli lead to equal responses because neither leads to a single node gaining supremacy. However in Figure 7e and f the two

unequal point stimuli upset the balance so that the lateral inhibition from the node responding to the stronger stimulus suppresses the response to the smaller stimulus. The effect is highly non-linear; the ratio of the amplitudes of the two stimuli and background is 1.2 : 1.1 : 1.0, whereas the ratio of the responses is such that the largest is greatly exaggerated. The effect of node activity leakage is to damp out this behaviour somewhat.

#### B. Symmetry breaking and completeness

Another interesting effect that PMDs exhibit is symmetry breaking, such that even if the statistical properties of the training data are translation invariant (i.e., each node is treated equivalently), the stable solutions for the parameter vectors  $\mathbf{s}_c(\mathbf{X})$  break this symmetry (i.e., the nodes do inequivalent jobs). Furthermore, the symmetry is broken in a very specific way, as will become clear below.

The results of Figure 8 show that a Gaussian PMD is very sensitive to the presence of activity leakage from a node to its neighbours. Only four iterations of the EM algorithm are shown (which is actually insufficient to obtain convergence), but nevertheless some interesting observations can be made. The effect of leakage is to order the Gaussian means in such a way that they change progressively from node to node along the PMD; this ordering is conspicuously absent in the PMD without leakage. Furthermore, the progression fluctuates regularly as indicated by the dotted line that connects the means. Note that within each mixture window (which has width 7) there is approximately one cycle of this fluctuation. In fact each mixture window possesses a complete repertoire of Gaussians with which to process the data that it sees in its input window, irrespective of whether there is node activity leakage or not. This will be referred to as the ‘completeness’ property.

When the EM algorithm is run for 10 iterations and the variances are updated as well as the means, then the simulation with leakage leads to the result shown in Figure 9. The dotted line shows that the fluctuations of the means have settled down to an almost perfect sinusoidal form with approximately one cycle per mixture window. The  $\frac{1}{4}$  and  $\frac{1}{2}$  half standard deviation curves are shown to indicate the thickness of the ‘Gaussian tubes’. These tubes are thinner where they are clustered closer together at the top and bottom of the figure. This effect occurs because the training data density is greater where the randomly phased sinusoids in the training set have turning points.

In Figure 10 the result of a 2-dimensional simulation

<sup>1</sup> Iterating a PMD is not as ad hoc as it might first appear to be; it is a particular implementation of a PMD that incorporates discrete-time dynamics.

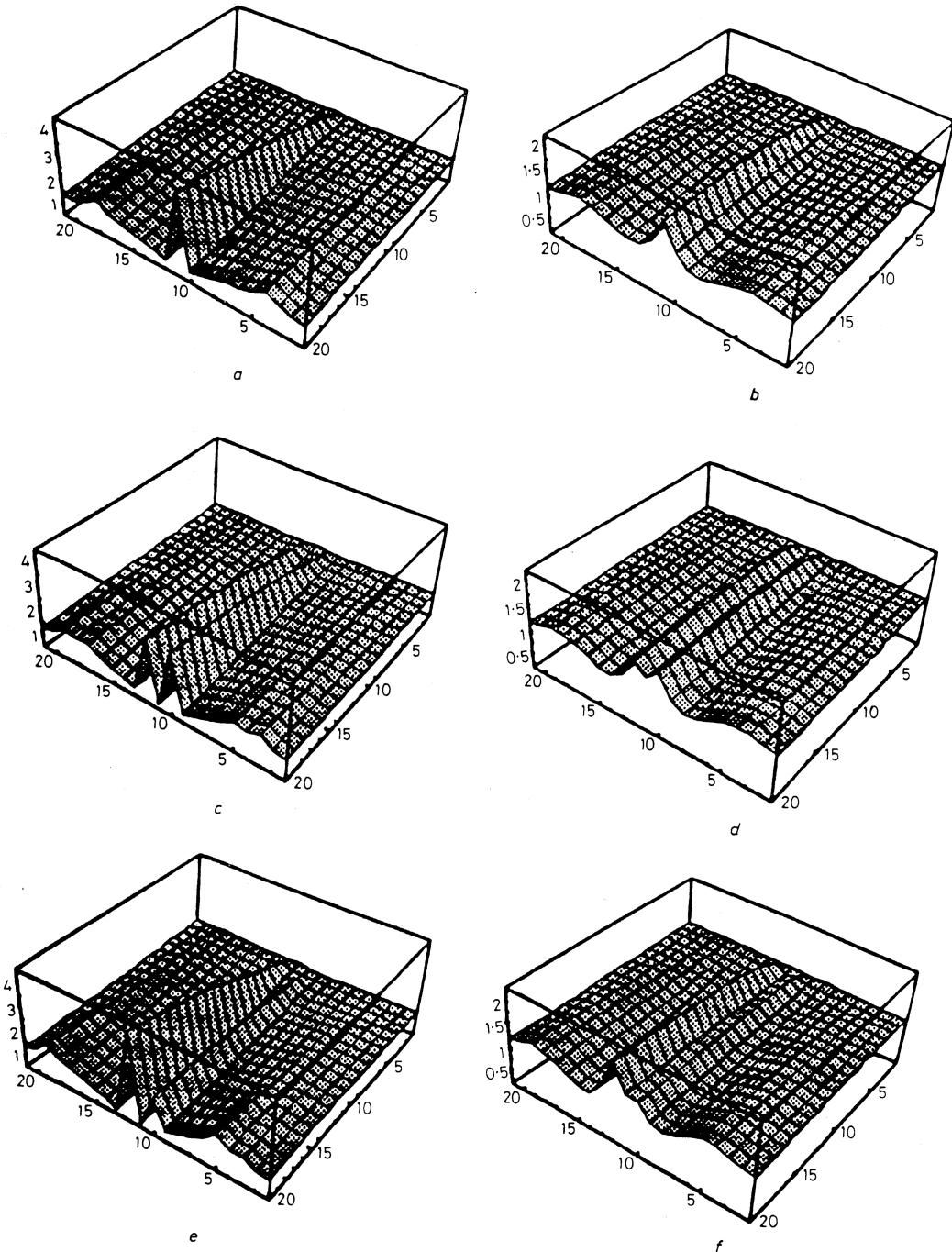


Figure 7: Simulation of PMD having a one-dimensional array of 21 nodes with a mixture window width of 7. Each row of the Figure uses a different pattern of likelihoods: *a* and *b* have a point stimulus of amplitude 1.2 embedded in a constant background stimulus of amplitude 1, *c* and *d* have two equal point stimuli of amplitude 1.15 separated by one node embedded in a constant background stimulus of amplitude 1, and *e* and *f* have two unequal point stimuli of amplitude 1.1 and 1.2 separated by one node embedded in a constant background stimulus of amplitude 1. The left hand column is an iterated PMD with no activity leakage, whereas the right hand column has activity leakage such that 10% of node's activity leaks onto each of its immediate left and right hand neighbours. In each simulation the activity was initialised to a constant 1, and then developed by iterating the PMD 20 times. The history of the resulting patterns of activity is displayed.

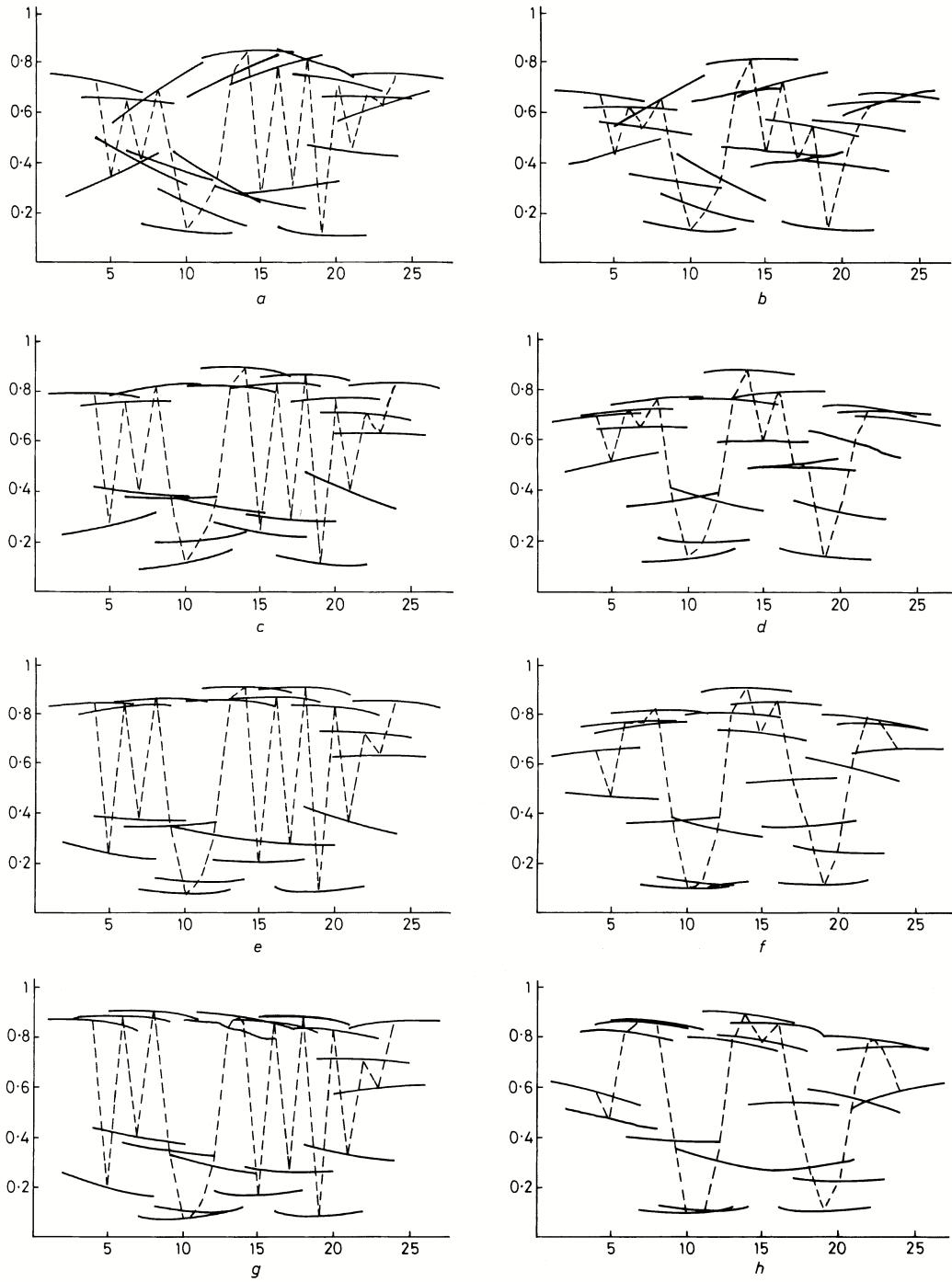


Figure 8: Simulation of PMD having a one-dimensional array of 21 nodes with a mixture window width of 7 and an input window width of 7. The likelihood functions are Gaussian, each with a constant variance of 0.1, and a trainable mean whose components are initialised to be random numbers uniformly drawn from the interval [0,1]. Each training vector was a 27 (= 21 + 3 + 3 to prevent the input window from falling off the ends of the input vector) dimensional vector comprising a sampled sinusoid advancing at 0.1 radians per sample. The starting phase of the sinusoid was randomly chosen for each training vector, and the height of sinusoid was rescaled to the interval [0,1] (i.e., add 1 and divide by 2). Each successive row of the figure shows the Gaussian means after one further iteration of the EM algorithm; four iterations are shown, starting with the result of the first EM update. The left hand column is a standard PMD, whereas the right hand column has node activity leakage such that 10% of each node's activity leaks onto each of its immediate left and right hand neighbours. The dotted lines join the centre of each Gaussian mean to its neighbours' means.

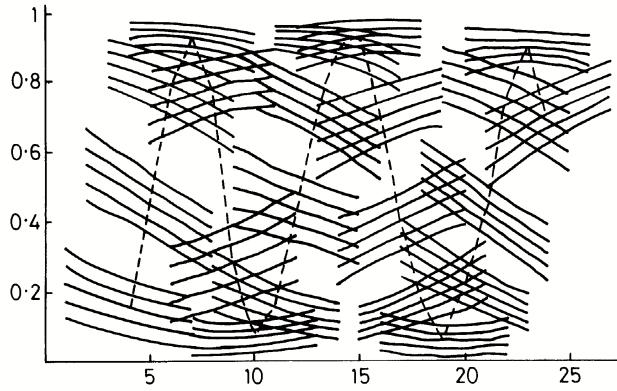


Figure 9: Result of 10 iterations of the EM algorithm. The parameters are the same as for the simulation with leakage in Figure 8. In this simulation both the variances and the means are updated. Each mean and variance is shown as 5 curves: mean, mean  $\pm 0.25\sqrt{\text{variance}}$  and mean  $\pm 0.5\sqrt{\text{variance}}$ .

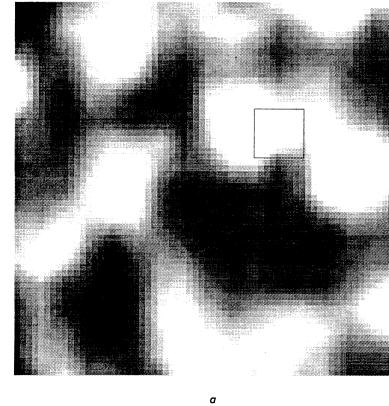
is shown for comparison with Figure 9. The fluctuations appear as they did in the one-dimensional simulation. The completeness property may be observed by inspecting the part of the montage that is contained in the superimposed square. The size of this square is the same as a MD window, and it contains a complete repertoire of Gaussians (i.e., it exhibits completeness). This completeness property is observed for all possible locations of the square on the montage.

## VI. CONCLUSIONS

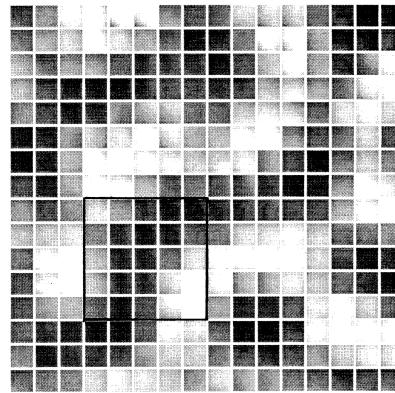
Mixture distributions are a natural candidate for modelling the PDFs that arise in Bayesian processing. In situations where many marginal PDFs need to be modelled simultaneously, such as in image processing applications, the partitioned mixture distribution approach introduced in this paper proves to be more natural than the standard mixture distribution approach. A partitioned mixture distribution can be viewed as a two-layer neural network with a limited input window for each output node, and a limited range of lateral interaction between the output nodes.

Two interesting behaviours of partitioned mixture distributions have been demonstrated in this paper. First, lateral inhibition arises such that an active output node tends to suppress activity in its neighbouring output nodes. Second, partitioned mixture distributions self-organise to yield automatically a number of overlapping mixture distribution models.

Finally, partitioned mixture distribution networks have a locally connected structure that is amenable to hardware implementation. This opens up the possibility of constructing a fast low-level image processing engine based entirely on rigorous Bayesian principles.



a



b

Figure 10: Result of a two-dimensional PMD simulation. This PMD is a  $16 \times 16$  array of nodes, each input window is  $9 \times 9$ , and each mixture window is  $5 \times 5$ . (a) Example of a training image, where the superimposed square indicates the size of an input window. (b) A montage formed from the Gaussian means after training, where the superimposed square indicates the size of a mixture window.

- [1] R T Cox, *Probability, frequency and reasonable expectation*, American Journal of Physics **14** (1946), no. 1, 1–13.
- [2] A P Dempster, N M Laird, and D B Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society B **39** (1977), no. 1, 1–38.
- [3] T Kohonen, *The self-organising map*, Proceedings of the IEEE **78** (1992), no. 9, 1464–1480.
- [4] S P Luttrell, *Partitioned mixture distributions: an introduction*, Memorandum 4671, Defence Research Agency, 1992.
- [5] \_\_\_\_\_, *An adaptive Bayesian network for low-level image processing*, Proceedings of IEE conference on artificial neural networks (Brighton), IEE, 1993, pp. 61–65.
- [6] D M Titterington, A F Smith, and U E Makov, *Statistical analysis of finite mixture distributions*, Wiley, New York, 1985.

# A Bayesian Analysis of Self-Organising Maps \*

Stephen P Luttrell

*Adaptive Systems Theory Section, Defence Research Agency,  
St Andrews Rd., Malvern, Worcestershire, WR14 3PS, United Kingdom*

In this paper Bayesian methods are used to analyse some of the properties of a special type of Markov chain. The forward transitions through the chain are followed by inverse transitions (using Bayes' theorem) backwards through a copy of the same chain; this will be called a folded Markov chain. If an appropriately defined Euclidean error (between the original input and its "reconstruction" via Bayes' theorem) is minimised with respect to the choice of Markov chain transition probabilities, then the familiar theories of both vector quantisers and self-organising maps emerge. This approach is also used to derive the theory of self-supervision, in which the higher layers of a multi-layer network supervise the lower layers, even though overall there is no external teacher.

## I. INTRODUCTION

A self-organising map (SOM) is an adaptive function that transforms (or maps) from an input vector space to an output vector space, where the adaptation is driven entirely by signals derived from the input space. In the context of neural networks an SOM would therefore be realised as an unsupervised network whose training algorithm acted to minimise a suitably defined error function in the network's input space. The aim of this paper is to develop a theoretical framework that unifies several different strands of unsupervised network theory, and for this purpose it is best to develop the theory anew, rather than to build a hybrid theory out of an assortment of existing theories.

### A. Variational formulation of SOM optimisation

In order to create a theoretically clean framework it is necessary to express the optimisation of an SOM in terms of a variational principle. Thus define a scalar functional  $D$  as follows

$$D \equiv \int d\mathbf{x} P(\mathbf{x}) d[\mathbf{x}, \mathbf{y}(\mathbf{x}), \mathbf{x}'(\mathbf{y})] \quad (1.1)$$

where the vectors  $\mathbf{x}$  and  $\mathbf{y}$  sit in the input and output spaces respectively, and the functions  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  transform from  $\mathbf{x}$ -space to  $\mathbf{y}$ -space and vice-versa respectively. These are the functions that must be optimised so as to minimise the value of  $D$ . The scalar function  $d[\mathbf{x}, \mathbf{y}(\mathbf{x}), \mathbf{x}'(\mathbf{y})]$  contributes to the functional  $D$  an amount that depends on the currently selected input vector  $\mathbf{x}$ , and on the functions  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$ . The function  $P(\mathbf{x})$  is a probability density (or measure) that weights the  $\mathbf{x}$  integral non-uniformly. In practice  $P(\mathbf{x})$

might be used to represent the relative frequency of occurrence of each input vector  $\mathbf{x}$  in a representative set of input vectors, in which case  $D$  would be the average of  $d[\mathbf{x}, \mathbf{y}(\mathbf{x}), \mathbf{x}'(\mathbf{y})]$  over that set of vectors.

In Equation 1.1 the functional dependence of  $d[\mathbf{x}, \mathbf{y}(\mathbf{x}), \mathbf{x}'(\mathbf{y})]$  can be simplified to  $d\{\mathbf{x}, \mathbf{x}'[\mathbf{y}(\mathbf{x})]\}$ , so  $D$  simplifies to

$$D = \int d\mathbf{x} P(\mathbf{x}) d\{\mathbf{x}, \mathbf{x}'[\mathbf{y}(\mathbf{x})]\} \quad (1.2)$$

This is possible because in  $d[\mathbf{x}, \mathbf{y}(\mathbf{x}), \mathbf{x}'(\mathbf{y})]$  the vector  $\mathbf{y}$  in the function  $\mathbf{x}'(\mathbf{y})$  is a placeholder that receives the output of the function  $\mathbf{y}(\mathbf{x})$ . The variational problem is then to find functions  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  that minimise  $D$  as defined in Equation 1.2.

Equation 1.2 can readily be extended to include the effects of an additive noise process acting in  $\mathbf{y}$ -space. Thus  $D$  becomes

$$D = \int d\mathbf{x} P(\mathbf{x}) \int d\mathbf{n} \pi(\mathbf{n}) d\{\mathbf{x}, \mathbf{x}'[\mathbf{y}(\mathbf{x}) + \mathbf{n}]\} \quad (1.3)$$

where  $\pi(\mathbf{n})$  is the probability density of the noise vector  $\mathbf{n}$ . The variational problem is then to find functions  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  that minimise this augmented expression for  $D$ . A side effect of including the noise process is that the information that is stored in the output vector  $\mathbf{y}$  (produced by the action of  $\mathbf{y}(\mathbf{x})$  transforming the input vector  $\mathbf{x}$ ) is represented in such a way that it is robust with respect to the damaging effects of the noise process.

### B. Variational principle versus unsupervised network notation

By making the replacement  $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2$  (i.e., a Euclidean distance) in Equation 1.3 this variational formulation can be shown to lead to a type of SOM that is similar to, but not precisely the same as, the unsupervised network that is known as the Kohonen map (see [3]) with the noise probability density  $\pi(\mathbf{n})$  playing the role of the SOM neighbourhood function. This type of analysis was also introduced in a non-neural context to design

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 22, 2019.

This paper appeared in Neural Computation, 1994, vol. 6, pp. 767-794. Communicated by Graeme Mitchison. Received May 19, 1993; accepted September 7, 1993. © 1994 British Crown Copyright/DRA.

an optimal vector quantiser (VQ) codebook for encoding data for transmission along a noisy channel [1, 2, 4].

The above variational principle can be related to a corresponding unsupervised network as shown in the table below.

Note that in practice the output  $\mathbf{y}$  of an unsupervised network is usually the index of the “winning node”, which is a discrete-valued quantity. Whether the output is continuous or discrete is unimportant to the variational approach, but in this paper a continuum notation is used because it is more compact.

The basic derivations of the variational formulation are in [6, 9, 10], a simple application to time series compression is in [7], the compression of synthetic aperture radar images is in [8], an analysis of the density of reference vectors is in [11], and an extension of unsupervised networks to multi-layer networks in which higher layers supervise lower layers (self-supervised networks) is in [12][13].

### C. Encoding prescriptions

It has been noted [2, 6, 9, 10] that the above Euclidean error function is not minimised when the nearest neighbour encoding prescription is used. Exact minimisation requires a new type of winner-take-all encoding prescription to be used; the nearest neighbour prescription must be replaced by the so-called minimum distortion prescription, in which the choice of winner is influenced by the reference vectors to which it is connected by the SOM neighbourhood function. Figure 1 shows graphically how nearest neighbour and minimum distortion encoding are related.

The minimum distortion prescription was used in [11] to study the equilibrium density of reference vectors in a 1-dimensional input space, where it was reported that the result was insensitive to the choice of neighbourhood function used in the SOM, provided that it was a monotonically decreasing symmetric function. The corresponding results using a standard SOM with the nearest neighbour prescription appeared in [14], where it was shown that a neighbourhood-sensitive density of reference vectors emerged.

### D. Self-supervised multi-layer SOMs

Hierarchical multi-layer SOMs were studied in [12, 13]. For instance, a 2-layer version of this type of network splits the input space into a number of lower dimensional subspaces, and trains a SOM in each subspace. Simultaneously, another SOM is trained on the outputs of the above SOMs to produce the final network output. This approach can be cascaded to more stages if required. This type of multi-layer SOM can be refined by introducing an error function which measures the average Euclidean error between the input vector and a

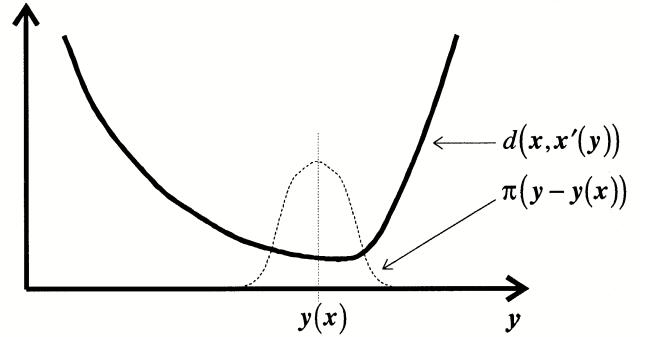


Figure 1: Assume that the input vector  $\mathbf{x}$  and the function  $x'(\mathbf{y})$  are such that the error function  $d[\mathbf{x}, \mathbf{x}'(\mathbf{y})]$  has the  $\mathbf{y}$  dependence shown above by the solid curve. There is a well-defined minimum, and the shape of the minimum is skewed so that  $d[\mathbf{x}, \mathbf{x}'(\mathbf{y})]$  increases more rapidly to the right than to the left of the minimum. Assume that the SOM neighbourhood function  $\pi[\mathbf{y} - \mathbf{y}(\mathbf{x})]$  is the symmetric function denoted by the dashed curve. The error function must be averaged over  $\mathbf{y}$  (weighted by the SOM neighbourhood function) to determine the expected error for the particular choice of  $\mathbf{x}$ ,  $x'(\mathbf{y})$  and  $y(\mathbf{x})$ . In effect,  $d[\mathbf{x}, \mathbf{x}'(\mathbf{y})]$  is convolved with  $\pi(\mathbf{y})$  to produce a smeared error function. The encoding prescription is one's choice of  $y(\mathbf{x})$ , and the optimum choice is not the minimum of the original error function with respect to  $\mathbf{y}$  (nearest neighbour encoding), but rather the minimum of the smeared error function with respect to  $\mathbf{y}$  (minimum distortion encoding). Because of the skewed  $d[\mathbf{x}, \mathbf{x}'(\mathbf{y})]$  the minimum distortion encoding of  $\mathbf{x}$  is thus displaced a little to the left of the nearest neighbour encoding.

reference vector that is constructed from the final network output [12, 13]. This forces the optimisation of the layers to be tied together in such a way that they have to be trained co-operatively. For instance, in a 2-layer network the outputs from the SOMs in the first layer have to be matched to the capabilities of the SOM in the second layer; this is achieved by sending back-propagation signals from the second layer to the first layer. However, these back-propagation signals are not derived from an external supervisor, so this type of training algorithm is called “self-supervised”.

### E. Purpose of this paper

The purpose of this paper is to embed the above variational formulation of SOMs in a more general framework in which the transformations  $\mathbf{y}(\mathbf{x})$  and  $x'(\mathbf{y})$  are replaced by probabilistic transformations  $P_1(\mathbf{y}|\mathbf{x})$  and  $P_2(\mathbf{x}'|\mathbf{y})$ . Equation 1.2 then becomes

$$D = \int d\mathbf{x} P(\mathbf{x}) \int d\mathbf{y} dx' P_2(\mathbf{x}'|\mathbf{y}) P_1(\mathbf{y}|\mathbf{x}) d(\mathbf{x}, \mathbf{x}') \quad (1.4)$$

The integration over  $\mathbf{y}$  and  $x'$  performs a weighted average of  $d\{\mathbf{x}, \mathbf{x}'[\mathbf{y}(\mathbf{x})]\}$  over a variety of alternative transformations  $\mathbf{y}(\mathbf{x})$  and  $x'(\mathbf{y})$ , rather than just a single pair

Table I: Variational Principle and Unsupervised Networks.

Term	Variational principle	Unsupervised network
$\mathbf{x}$	Input vector	Training/test vector
$\mathbf{y}(\mathbf{x})$	Input to output transformation	Encoding prescription
$\mathbf{x}'(\mathbf{y})$	Output to input transformation	Reference vector
$d[\mathbf{x}, \mathbf{y}(\mathbf{x}), \mathbf{x}'(\mathbf{y})]$	Function	Error function
$P(\mathbf{x})$	Integration measure	Probability density of training/test vectors
$D$	Functional	Average error over the training/test set

of transformations as would have been the case in the basic variational formulation. Similarly Equation 1.3 be-

comes

$$D = \int d\mathbf{x} P(\mathbf{x}) \int d\mathbf{n} \pi(\mathbf{n}) \int d\mathbf{y} d\mathbf{x}' P_2(\mathbf{x}'|\mathbf{y} + \mathbf{n}) P_1(\mathbf{y}|\mathbf{x}) d(\mathbf{x}, \mathbf{x}') \quad (1.5)$$

This “sum over alternatives” is useful for a number of reasons.

1. Theoretical manipulations are easier to perform on “soft” probabilities than on “hard” deterministic functions.
2. The probabilistic approach lends itself well to a simulated annealing approach in numerical simulations.
3. Contact with standard results can be made by making the replacement  $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2$  (i.e. a Euclidean distance) in Equation 1.5. This leads to a probabilistic generalisation of standard SOM theory.
4. If a single pair of transformations  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  was being used, but one was uncertain about which particular pair, then a probabilistic formulation would be necessary.

#### F. Structure of this paper

The principal new result in this paper is a Bayesian derivation of the properties of SOMs starting from a generalisation of Equation 1.4. Thus a Markov chain of probabilistic transformations of the input vector is inverted by sending the (probabilistic) output of the chain back through the inverse probabilistic transformations (derived from Bayes’ theorem) to eventually re-emerge as a (probabilistic) reconstructed version of the input vector. This type of structure will be called a folded Markov chain (FMC). This FMC is then optimised by minimising the average Euclidean error between the input vector

and its reconstruction. Various constraints can be placed on this optimisation. For instance, if a 2-stage FMC (i.e. 2 stages of probabilistic transformation) is considered, and only its first stage is optimised, then the theory of SOMs emerges (as in [6, 9, 10]). Alternatively, if the state space of each stage of a 2-stage FMC is split into two (or more) lower dimensional subspaces, then the theory of self-supervision emerges (as in [12, 13]).

The structure of this paper is as follows. In Section II the idea of an FMC will be introduced, where Bayes’ theorem is used to invert a Markov chain of probabilistic transformations. In Section III the relationship between FMCs and VQs and SOMs is derived, and it is shown how a 1-stage (or 2-stage) FMC contains a VQ (or SOM) as a special case. In Section IV these results are extended to the case of a pair of coupled FMCs, and it is shown how self-supervision emerges naturally. In the first Appendix the relationship between the continuum notation that is used in this paper and the more usual discrete notation is explained. In the second Appendix a more complete and technically rigorous derivation of the results of section 3 is presented. The main new results are contained in Section III and Section IV.

## II. BASIC THEORY

The notation that is used for probabilities is very carefully chosen so as to avoid ambiguities that could arise if the notation  $P(\dots)$  were used blindly to denote “the probability density of ...”. However, occasional use of the ambiguous  $P(\dots)$  notation is made where there is no possibility of an ambiguity arising. Also, a careful distinction is drawn between the notation that is used for inverse probabilities (obtained from Bayes’ theorem), and

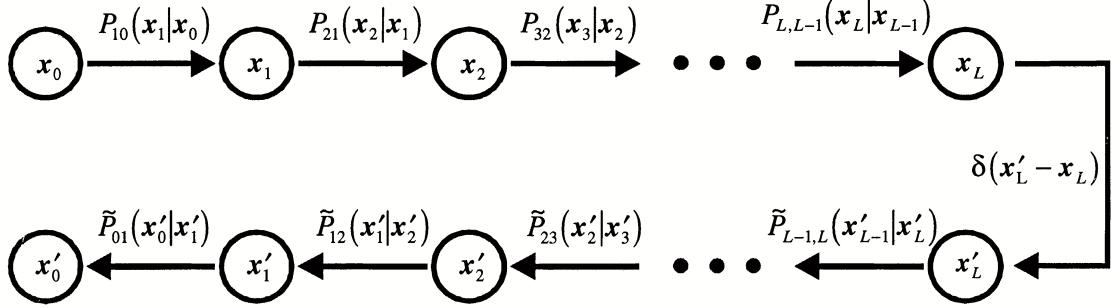


Figure 2: A folded Markov chain (i.e. both the forwards and backwards directions are represented). The top (bottom) half of the diagram represents the forward (backward) pass through the chain. The conditional probability  $P_{k+1,k}(\mathbf{x}_{k+1}|\mathbf{x}_k)$  is used as a probabilistic transformation that generates the probable states of layer  $k+1$  from the state of layer  $k$ , and the conditional probability  $\tilde{P}_{k,k+1}(\mathbf{x}'_k|\mathbf{x}'_{k+1})$  is similarly used to generate the probable states of layer  $k$  from the state of layer  $k+1$ ; this is also referred to generically as stage  $k$  of the FMC. These two conditional probabilities are related by Bayes' theorem.

the notation that is used for forward probabilities; the former always appear with a tilde over the  $P$ , which thus appears as  $\tilde{P}$ . Also the state space(s) will be assumed to be continuous, except where otherwise noted. This is because the corresponding discrete space results can be written down by inspection of the continuum results, and because the meaning of a continuum calculation is usually more transparent than its discrete counterpart.

The type of Markov chain that will be considered is shown in Figure 2.

This is a folded Markov chain (FMC) which performs an  $L$ -stage transformation of an input vector  $\mathbf{x}_0$  to an output vector  $\mathbf{x}_L$  (via  $L-1$  intermediate vectors

$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{L-1}$ ), and then performs the Bayes' inverse transformation to arrive eventually at a reconstructed input vector  $\mathbf{x}'_0$ . The delta function  $\delta(\mathbf{x}_L - \mathbf{x}'_L)$  is used to ensure that  $\mathbf{x}'_L = \mathbf{x}_L$ . The conditional probabilities are related by Bayes' theorem as follows

$$\tilde{P}_{k,k+1}(\mathbf{x}_k|\mathbf{x}_{k+1}) P_{k+1}(\mathbf{x}_{k+1}) = P_{k+1,k}(\mathbf{x}_{k+1}|\mathbf{x}_k) P_k(\mathbf{x}_k) \quad (2.1)$$

where  $P_k(\mathbf{x}_k)$  denotes the marginal probability (density) of  $\mathbf{x}_k$ . In order to construct any joint probability in the FMC system it is both necessary and sufficient to specify  $P_0(\mathbf{x}_0)$  and the  $L$  transformations  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0), P_{2,1}(\mathbf{x}_2|\mathbf{x}_1), \dots, P_{L,L-1}(\mathbf{x}_L|\mathbf{x}_{L-1})$ . Thus

---


$$\begin{aligned} P(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_L; \mathbf{x}'_L, \dots, \mathbf{x}'_1, \mathbf{x}'_0) &= P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) P_{2,1}(\mathbf{x}_2|\mathbf{x}_1) \dots \\ &\quad \dots P_{L,L-1}(\mathbf{x}_L|\mathbf{x}_{L-1}) \delta(\mathbf{x}'_L - \mathbf{x}_L) \tilde{P}_{L-1,L}(\mathbf{x}'_{L-1}|\mathbf{x}'_L) \dots \\ &\quad \dots \tilde{P}_{1,2}(\mathbf{x}'_1|\mathbf{x}'_2) \tilde{P}_{0,1}(\mathbf{x}'_0|\mathbf{x}'_1) \end{aligned} \quad (2.2)$$

The marginal probability  $P_k(\mathbf{x}_k)$  is obtained by integrating over all variables other than  $\mathbf{x}_k$ , which yields

$$P_k(\mathbf{x}_k) = \int d\mathbf{x}_0 d\mathbf{x}_1 \dots d\mathbf{x}_{k-1} P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) P_{2,1}(\mathbf{x}_2|\mathbf{x}_1) \dots P_{k,k-1}(\mathbf{x}_k|\mathbf{x}_{k-1}) \quad (2.3)$$


---

Note that Bayes' theorem guarantees that the marginal probabilities of  $\mathbf{x}_k$  and  $\mathbf{x}'_k$  are the same. In this paper our attention will be restricted to FMCs with 1 or 2 stages only (i.e.  $L = 1$  or  $2$ ). Furthermore, whenever an FMC is to be optimised, only its first stage will be optimised.

### III. 1- AND 2-STAGE FOLDED MARKOV CHAINS

FMCs are especially interesting in adaptive network design theory, because they turn out to contain some well-known systems as special cases: a VQ is a special case of a 1-stage FMC, and an SOM is a special case of a 2-stage FMC. The success of these derivations relies on the similarities that exist between the following two situations:

1. Direct/inverse probabilities occur in Bayes' theo-

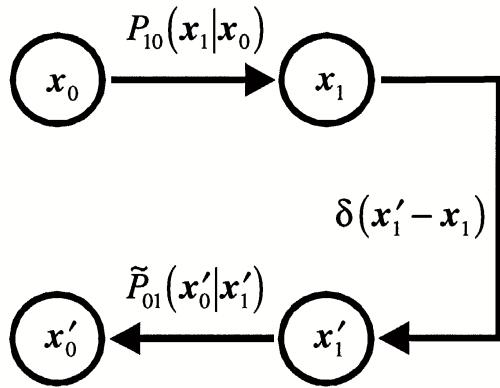


Figure 3: A 1-stage folded Markov chain. When  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) = \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$ , and  $D$  is minimised with respect to  $\mathbf{x}_1(\mathbf{x}_0)$ , this reduces to a vector quantiser with an infinite number of reference vectors (i.e. continuum limit). See the Appendix for a detailed discussion of the relationship between the continuum and discrete cases. Bayes' theorem ensures that  $\tilde{P}_{0,1}(\mathbf{x}'_0|\mathbf{x}'_1)$  cannot be varied independently of  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ .

rem as applied to a Markov chain, which might be used in the analysis of scattering problems in layered media, for instance. The sequence of processing operations is source  $\rightarrow$  scattering  $\rightarrow$  inverse scattering  $\rightarrow$  reconstruction.

2. Encoding/decoding operations occur in VQs and SOMs, which might be used in the analysis of information transmission down a noisy communication channel, for instance. The sequence of processing operations is source  $\rightarrow$  encode  $\rightarrow$  decode  $\rightarrow$  reconstruction.

#### A. 1-Stage folded Markov chain: vector quantiser

A 1-stage FMC is shown in Figure 3. The derivation starts from the definition of the average Euclidean error  $D$  between  $\mathbf{x}_0$  and  $\mathbf{x}'_0$  in a 1-stage FMC.

$$D = \int d\mathbf{x}_0 d\mathbf{x}_1 d\mathbf{x}'_0 d\mathbf{x}'_1 P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \delta(\mathbf{x}'_1 - \mathbf{x}_1) \tilde{P}_{0,1}(\mathbf{x}'_0|\mathbf{x}'_1) \|\mathbf{x}'_0 - \mathbf{x}_0\|^2 \quad (3.1)$$

After integrating out the (irrelevant)  $\mathbf{x}'_1$  using the delta function  $\delta(\mathbf{x}'_1 - \mathbf{x}_1)$ , using Bayes' theorem in the form  $\tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1) P_1(\mathbf{x}_1) = P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) P_0(\mathbf{x}_0)$  (i.e. Equation 2.1 with  $k = 0$ ), and rearranging the order of the integrations, this reduces to

$$D = \int d\mathbf{x}_1 P_1(\mathbf{x}_1) \int d\mathbf{x}_0 d\mathbf{x}'_0 \tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1) \tilde{P}_{0,1}(\mathbf{x}'_0|\mathbf{x}_1) \|\mathbf{x}'_0 - \mathbf{x}_0\|^2 \quad (3.2)$$

The next step is to expand the norm as  $\|\mathbf{x}_0\|^2 + \|\mathbf{x}'_0\|^2 - 2\mathbf{x}_0 \cdot \mathbf{x}'_0$  and to perform the integrations where possible to obtain

$$D = 2 \int d\mathbf{x}_1 P_1(\mathbf{x}_1) \left[ \int d\mathbf{x}_0 \tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1) \|\mathbf{x}_0\|^2 - \left\| \int d\mathbf{x}_0 \tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1) \mathbf{x}_0 \right\|^2 \right] \quad (3.3)$$

which may be rewritten as

$$D = 2 \int d\mathbf{x}_1 P_1(\mathbf{x}_1) \int d\mathbf{x}_0 \tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1) \left\| \mathbf{x}_0 - \int d\mathbf{u}_0 \tilde{P}_{0,1}(\mathbf{u}_0|\mathbf{x}_1) \mathbf{u}_0 \right\|^2 \quad (3.4)$$

The derivation of Equation 3.4 from Equation 3.2 is well-known; it says that the average Euclidean error between pairs of vectors drawn independently from  $\tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1)$  is twice the variance of vectors drawn from  $\tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1)$ . Finally, Bayes' theorem (i.e. Equation 2.1 with  $k = 0$ ) can be used to obtain the required result.

$$D = 2 \int d\mathbf{x}_0 P_0(\mathbf{x}_0) \int d\mathbf{x}_1 P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \left\| \mathbf{x}_0 - \int d\mathbf{u}_0 \tilde{P}_{0,1}(\mathbf{u}_0|\mathbf{x}_1) \mathbf{u}_0 \right\|^2 \quad (3.5)$$

Equation 3.5 has all of the right structure to relate FMCs to VQs. It has a source of input vectors  $P_0(\mathbf{x}_0)$ ,

a “soft” encoder  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ , and a reference vector  $\int d\mathbf{u}_0 \tilde{P}_{0,1}(\mathbf{u}_0|\mathbf{x}_1) \mathbf{u}_0$  attached to each  $\mathbf{x}_1$  with which to

compare the input vector  $\mathbf{x}_0$  to compute a Euclidean distortion. The only differences between this FMC and a standard VQ are:

1.  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  is not a winner-take-all encoder. Each input vector  $\mathbf{x}_0$  is transformed into each possible output vector  $\mathbf{x}_1$  with probability  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ . In the language of neural networks, it is as if each possible output vector had an “activity” specified by  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ . A winner-take-all would result if  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  were replaced by a probability whose mass was concentrated all at one point; this would be a delta function  $\delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$ .
2. The reference vector  $\int d\mathbf{u}_0 \tilde{P}_{0,1}(\mathbf{u}_0|\mathbf{x}_1) \mathbf{u}_0$  is dependent on the encoder  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ ; they are related by Bayes’ theorem (i.e. Equation 2.1 with  $k = 0$ ). In a VQ the reference vector and the encoder are also related because the encoder is usually a nearest neighbour prescription, which in turn depends on the location of the reference vectors. It is not at all obvious that these two pictures (the FMC and the VQ) are related in a simple way.

Now consider a modified form of Equation 3.5 in which  $D$  becomes

$$D = 2 \int d\mathbf{x}_0 P_0(\mathbf{x}_0) \int d\mathbf{x}_1 P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_1)\|^2 \quad (3.6)$$

where  $\int d\mathbf{u}_0 \tilde{P}_{0,1}(\mathbf{u}_0|\mathbf{x}_1) \mathbf{u}_0$  has been replaced by the function  $\mathbf{x}'_0(\mathbf{x}_1)$ . Functionally differentiate this expression for  $D$  with respect to  $\mathbf{x}'_0(\mathbf{x}_1)$  to obtain

$$\frac{\delta D}{\delta \mathbf{x}'_0(\mathbf{x}_1)} = -4 \int d\mathbf{x}_0 P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) [\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_1)] \quad (3.7)$$

The stationary point  $\frac{\delta D}{\delta \mathbf{x}'_0(\mathbf{x}_1)} = 0$  is obtained when  $\mathbf{x}'_0(\mathbf{x}_1)$  satisfies

$$\mathbf{x}'_0(\mathbf{x}_1) = \frac{\int d\mathbf{x}_0 P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \mathbf{x}_0}{\int d\mathbf{x}_0 P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)} \quad (3.8)$$

By using Bayes’ theorem (i.e. Equation 2.1 with  $k = 0$ ) this stationarity condition reduces to

$$\mathbf{x}'_0(\mathbf{x}_1) = \int d\mathbf{x}_0 \tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1) \mathbf{x}_0 \quad (3.9)$$

So the modified expression for  $D$  in Equation 3.6 reduces to the original expression for  $D$  in Equation 3.5 provided that  $\mathbf{x}'_0(\mathbf{x}_1)$  is chosen to minimise  $D$ . This is a major simplification, because the coupling (via Bayes’ theorem) between  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  and  $\int d\mathbf{u}_0 \tilde{P}_{0,1}(\mathbf{u}_0|\mathbf{x}_1) \mathbf{u}_0$  that appeared in Equation 3.5 can now safely be ignored by the

simple trick of using Equation 3.6 instead (with the proviso that  $\mathbf{x}'_0(\mathbf{x}_1)$  should always be optimised so as to minimise  $D$ ).

Make the following replacement in Equation 3.6

$$P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \longrightarrow \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)] \quad (3.10)$$

which converts  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  into a winner-take-all encoder. A winner-take-all encoder might appear intuitively to be the obvious solution to the problem of minimising  $D$  with respect to  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ . However, other solutions may also be possible in general. A detailed derivation is given in the Appendix to show how this result emerges in the case of a Euclidean error function. Also a couple of simple counterexamples are presented in the Appendix to show how non-winner-take-all encoders can also be valid solutions.

With these replacements  $D$  becomes

$$D = 2 \int d\mathbf{x}_0 P_0(\mathbf{x}_0) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_1(\mathbf{x}_0))\|^2 \quad (3.11)$$

which is exactly what would be written for the continuum version of a VQ (apart from the trivial overall factor of 2). The network representation of a VQ is shown in Figure 4 for a discrete-valued output, which should be compared with the FMC representation shown in Figure 3.

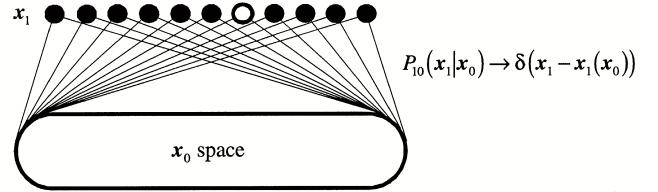


Figure 4: A discrete VQ represented as a network. The bottom layer is the input space (or  $\mathbf{x}_0$ ), the top layer is the output space (or  $\mathbf{x}_1$ ), and the connections between the two represent a soft encoding operation  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ , akin to that used in [15]. A winner-take-all VQ uses an encoder of the form  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) = \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$ . The input and output layers of this network are represented in different ways: the input layer is the vector  $\mathbf{x}_0$  represented as in Figure 3, whereas each node of the output layer corresponds to exactly one possible state of the vector  $\mathbf{x}_1$ . Note also that the connections are not to be interpreted as weights in the conventional sense, rather they merely indicate the functional interdependence of the various parts of the network. An example of a winner in the output layer is represented by the open circle.

The gradient of  $D$  w.r.t.  $\mathbf{x}'_0(\mathbf{x}_1)$  is given by the functional derivative

$$\frac{\delta D}{\delta \mathbf{x}'_0(\mathbf{x}_1)} = -4 \int d\mathbf{x}_0 P_0(\mathbf{x}_0) \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)] [\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_1)] \quad (3.12)$$

By inspecting the dependence of Equation 3.11 on  $\mathbf{x}_1(\mathbf{x}_0)$ , and by setting  $\frac{\delta D}{\delta \mathbf{x}'_0(\mathbf{x}_1)}$  in Equation 3.12 to zero, the following batch training prescription for minimising  $D$  can be obtained

$$\begin{aligned} \mathbf{x}_1(\mathbf{x}_0) &= \arg \min_{\mathbf{x}_1} \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_1)\|^2 \\ \mathbf{x}'_0(\mathbf{x}_1) &= \frac{\int d\mathbf{x}_0 P_0(\mathbf{x}_0) \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)] \mathbf{x}_0}{\int d\mathbf{x}_0 P_0(\mathbf{x}_0) \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]} \end{aligned} \quad (3.13)$$

These results merit the following remarks:

1. The function  $\mathbf{x}'_0(\mathbf{x}_1)$  can be interpreted as the continuum version of a VQ codebook, where  $\mathbf{x}_1$  is the (continuum) code index and  $\mathbf{x}'_0(\mathbf{x}_1)$  is the code vector (or reference vector) associated with that index.
2. The result for  $\mathbf{x}'_0(\mathbf{x}_1)$  corresponds to Equation 3.8 with  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \rightarrow \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$ , and it is the “centroiding” prescription for updating the code vectors after a batch of training data has been presented to a VQ, as used in the LBG algorithm [5].
3. The result for  $\mathbf{x}_1(\mathbf{x}_0)$  is the “nearest neighbour” encoding prescription for encoding the input of a VQ.

An on-line training prescription can also be obtained to implement updates to  $\mathbf{x}'_0(\mathbf{x}_1)$  after each input vector  $\mathbf{x}_0$  is selected at random from  $P_0(\mathbf{x}_0)$ . The on-line prescription is

$$\mathbf{x}'_0(\mathbf{x}_1) \longrightarrow \mathbf{x}'_0(\mathbf{x}_1) + \epsilon \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)] [\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_1)] \quad (3.14)$$

Note that the delta function permits non-zero updates only for  $\mathbf{x}_0 = \mathbf{x}'_0(\mathbf{x}_1)$ ; this is the continuum version of updating the nearest neighbour code vector towards the input vector. The relationship between this continuum result and the corresponding discrete result is discussed in the Appendix.

This completes the demonstration that an optimal 1-stage FMC is a VQ. Note that the use of a Euclidean error

is sufficient (but not necessary) for this result to emerge. There are other choices of error function for which this result does not emerge.

### B. 2-Stage folded Markov chain: self-organising map

Because the derivation is so similar to the case of a 1-stage FMC, only an abbreviated derivation will be given.

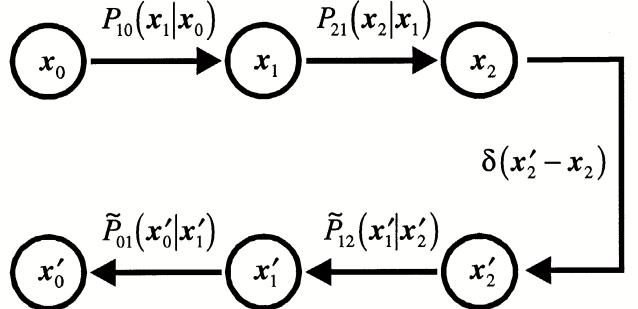


Figure 5: A 2-stage folded Markov chain. This is basically the same as Figure 3, except that  $P_{2,1}(\mathbf{x}_2|\mathbf{x}_1)$  causes the information that flows through the folded Markov chain to be further corrupted before it can begin its return journey.

For simplicity, the following notation will be used

$$\begin{aligned} P_{2,0}(\mathbf{x}_2|\mathbf{x}_0) &= \int d\mathbf{x}_1 P_{2,1}(\mathbf{x}_2|\mathbf{x}_1) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \\ \tilde{P}_{0,2}(\mathbf{x}_0|\mathbf{x}_2) P_2(\mathbf{x}_2) &= P_{2,0}(\mathbf{x}_2|\mathbf{x}_0) P_0(\mathbf{x}_0) \end{aligned} \quad (3.15)$$

$P_{2,0}(\mathbf{x}_2|\mathbf{x}_0)$  is the probability (density) that  $\mathbf{x}_2$  will be generated by  $\mathbf{x}_0$ , taking into account all of the possible values that the intermediate state  $\mathbf{x}_1$  might take.  $\tilde{P}_{0,2}(\mathbf{x}_0|\mathbf{x}_2)$  is the corresponding inverse probability that is obtained from Bayes’ theorem.

Using this notation the expression for  $D$  becomes (compare Equation 3.1)

$$\begin{aligned} D &= \int d\mathbf{x}_0 d\mathbf{x}_1 d\mathbf{x}_2 d\mathbf{x}'_0 d\mathbf{x}'_1 d\mathbf{x}'_2 P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) P_{2,1}(\mathbf{x}_2|\mathbf{x}_1) \delta(\mathbf{x}'_2 - \mathbf{x}_2) \\ &\quad \times \tilde{P}_{1,2}(\mathbf{x}'_1|\mathbf{x}'_2) \tilde{P}_{0,1}(\mathbf{x}'_0|\mathbf{x}'_1) \|\mathbf{x}'_0 - \mathbf{x}_0\|^2 \end{aligned} \quad (3.16)$$

which simplifies to (compare Equation 3.5)

$$D = 2 \int d\mathbf{x}_0 P_0(\mathbf{x}_0) \int d\mathbf{x}_2 P_{2,0}(\mathbf{x}_2|\mathbf{x}_0) \left\| \mathbf{x}_0 - \int d\mathbf{u}_0 \tilde{P}_{0,2}(\mathbf{u}_0|\mathbf{x}_2) \mathbf{u}_0 \right\|^2 \quad (3.17)$$


---

Consider a modified form of  $D$  (compare Equation 3.6)

$$D = 2 \int d\mathbf{x}_0 P_0(\mathbf{x}_0) \int d\mathbf{x}_2 P_{2,0}(\mathbf{x}_2|\mathbf{x}_0) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_2)\|^2 \quad (3.18)$$

Set  $\frac{\delta D}{\delta \mathbf{x}'_0(\mathbf{x}_2)} = 0$  to obtain (compare Equation 3.8 and Equation 3.9)

$$\begin{aligned} \mathbf{x}'_0(\mathbf{x}_2) &= \frac{\int d\mathbf{x}_0 P_0(\mathbf{x}_0) P_{2,0}(\mathbf{x}_2|\mathbf{x}_0) \mathbf{x}_0}{\int d\mathbf{x}_0 P_0(\mathbf{x}_0) P_{2,0}(\mathbf{x}_2|\mathbf{x}_0)} \\ &= \int d\mathbf{x}_0 \tilde{P}_{0,2}(\mathbf{x}_0|\mathbf{x}_2) \mathbf{x}_0 \end{aligned} \quad (3.19)$$

The modified expression for  $D$  in Equation 3.18 reduces to the original expression in Equation 3.17 provided that

---

$\mathbf{x}'_0(\mathbf{x}_2)$  is chosen to minimise  $D$ , so Equation 3.18 will be used in preference to Equation 3.17 (with the proviso that  $\mathbf{x}'_0(\mathbf{x}_2)$  should always be chosen to minimise  $D$ ).

Make the replacement  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \rightarrow \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$  which converts  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  into a winner-take-all encoder (see the discussion following Equation 3.10, and the Appendix for more details), and note that  $P_{2,0}(\mathbf{x}_2|\mathbf{x}_0) = \int d\mathbf{x}_1 P_{2,1}(\mathbf{x}_2|\mathbf{x}_1) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ , to obtain  $D$  as (compare Equation 3.11)

$$D = 2 \int d\mathbf{x}_0 P_0(\mathbf{x}_0) \int d\mathbf{x}_2 P_{2,1}[\mathbf{x}_2|\mathbf{x}_1(\mathbf{x}_0)] \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_2)\|^2 \quad (3.20)$$

The functional derivative  $\frac{\delta D}{\delta \mathbf{x}'_0(\mathbf{x}_2)}$  is thus (compare Equation 3.12)

$$\frac{\delta D}{\delta \mathbf{x}'_0(\mathbf{x}_2)} = -4 \int d\mathbf{x}_0 P_0(\mathbf{x}_0) P_{2,1}(\mathbf{x}_2|\mathbf{x}_1(\mathbf{x}_0)) (\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_2)) \quad (3.21)$$


---

which leads to the following batch training prescription for minimising  $D$  (compare Equation 3.13)

$$\begin{aligned} \mathbf{x}_1(\mathbf{x}_0) &= \arg \min_{\mathbf{x}_1} \int d\mathbf{x}_2 P_{2,1}(\mathbf{x}_2|\mathbf{x}_1) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_2)\|^2 \\ \mathbf{x}'_0(\mathbf{x}_2) &= \frac{\int d\mathbf{x}_0 P_0(\mathbf{x}_0) P_{2,1}[\mathbf{x}_2|\mathbf{x}_1(\mathbf{x}_0)] \mathbf{x}_0}{\int d\mathbf{x}_0 P_0(\mathbf{x}_0) P_{2,1}[\mathbf{x}_2|\mathbf{x}_1(\mathbf{x}_0)]} \end{aligned} \quad (3.22)$$

$$\begin{aligned} \mathbf{x}_1(\mathbf{x}_0) &= \arg \min_{\mathbf{x}_1} \int d\mathbf{x}_2 P_{2,1}(\mathbf{x}_2|\mathbf{x}_1) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_2)\|^2 \\ \mathbf{x}'_0(\mathbf{x}_2) &= \frac{\int d\mathbf{x}_0 P_0(\mathbf{x}_0) P_{2,1}[\mathbf{x}_2|\mathbf{x}_1(\mathbf{x}_0)] \mathbf{x}_0}{\int d\mathbf{x}_0 P_0(\mathbf{x}_0) P_{2,1}[\mathbf{x}_2|\mathbf{x}_1(\mathbf{x}_0)]} \end{aligned} \quad (3.23)$$

and the following on-line training prescription (compare Equation 3.14)

$$\mathbf{x}'_0(\mathbf{x}_2) \rightarrow \mathbf{x}'_0(\mathbf{x}_2) + \epsilon P_{2,1}[\mathbf{x}_2|\mathbf{x}_1(\mathbf{x}_0)] [\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_2)] \quad (3.24)$$

These results correspond to the results that were reported in [6, 9, 10]. They can be interpreted as the generalisation of the VQ results in the previous section to the case where the output of the VQ is corrupted by the action of  $P_{2,1}(\mathbf{x}_2|\mathbf{x}_1)$  before Bayes' theorem is then used in an attempt to reconstruct the input vector.

This winner-take-all version of a 2-stage FMC turns out to be an SOM whose network representation is shown in Figure 6 for a discrete-valued output, which should be

compared with the FMC representation that is shown in Figure 5

The SOM interpretation of these results for optimising a 2-stage FMC is as follows:

1. The function  $\mathbf{x}'_0(\mathbf{x}_2)$  can be interpreted as the continuum version of the SOM reference vectors, where  $\mathbf{x}_2$  is the (continuum) index and  $\mathbf{x}'_0(\mathbf{x}_2)$  is the reference vector associated with that index. The batch update prescription for  $\mathbf{x}'_0(\mathbf{x}_2)$  is a generalisation of the LBG “centroiding” prescription [5] which accounts for the effect of  $P_{2,1}(\mathbf{x}_2|\mathbf{x}_1)$ .
2. The result for  $\mathbf{x}_1(\mathbf{x}_0)$  is not a “nearest neighbour” encoding prescription. Rather, it says that  $\mathbf{x}_1(\mathbf{x}_0)$  is the value that  $\mathbf{x}_1$  must take in order to ensure that the distortion  $D$  is minimised after taking into account the effect of  $P_{2,1}(\mathbf{x}_2|\mathbf{x}_1)$ . Thus the nearest neighbour encoding prescription has become a minimum distortion encoding prescription. This reduces to the nearest neighbour encoding prescription when  $P_{2,1}(\mathbf{x}_2|\mathbf{x}_1) \rightarrow \delta(\mathbf{x}_2 - \mathbf{x}_1)$ , as expected.
3. The on-line training prescription is the continuum version of the standard SOM training prescription, where  $P_{2,1}(\mathbf{x}_2|\mathbf{x}_1)$  plays the role of the SOM neighbourhood function.  $P_{2,1}(\mathbf{x}_2|\mathbf{x}_1)$  also has this interpretation in the batch training prescription.

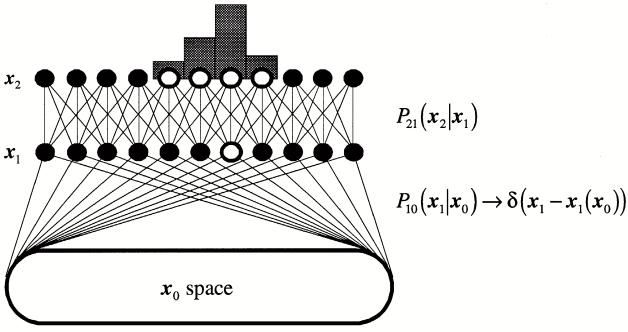


Figure 6: A discrete SOM represented as a network. This is the same as the VQ network in Figure 4 with an additional stage of processing applied to its output layer. Each node in the hidden and output layer of this SOM network corresponds to exactly one possible state of the vector  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , respectively.  $P_{2,1}(\mathbf{x}_2|\mathbf{x}_1)$  serves as the SOM neighbourhood function by connecting together states of  $\mathbf{x}_1$  so that they become ordered (across the page in this case). Note that the VQ network in Figure 4 does not have this ordering property, although the states of  $\mathbf{x}_1$  (or nodes) are still drawn in an ordered fashion, for convenience. An example of a winner in the hidden layer is drawn as an open circle, as are each of the corresponding soft winners in the output layer. An example of the degree to which each node in the output layer is activated is indicated by a histogram, which records  $P_{2,1}(\mathbf{x}_2|\mathbf{x}_1)$  for each possible state that  $\mathbf{x}_2$  might take.

This completes the demonstration that an optimal 2-stage FMC is an SOM. Note that minimum distortion encoding is used, rather than nearest neighbour encoding, so this type of SOM is only an approximation to the standard SOM that was discussed in [3].

#### IV. COUPLED 2-STAGE FOLDED MARKOV CHAINS

In [12, 13] some interesting results were reported where the behaviour of a multi-layer SOM could be interpreted as if the higher network layers were supervising the lower layers, and the term “self-supervision” was introduced to describe this effect. The purpose of this section is to show how these results can be derived from the theory of FMCs.

##### A. Splitting the Markov chain state spaces

The derivation starts by splitting each state of a 2-stage FMC into two lower dimensional pieces, as shown in Figure 7.

Figure 7 can be obtained by making the following

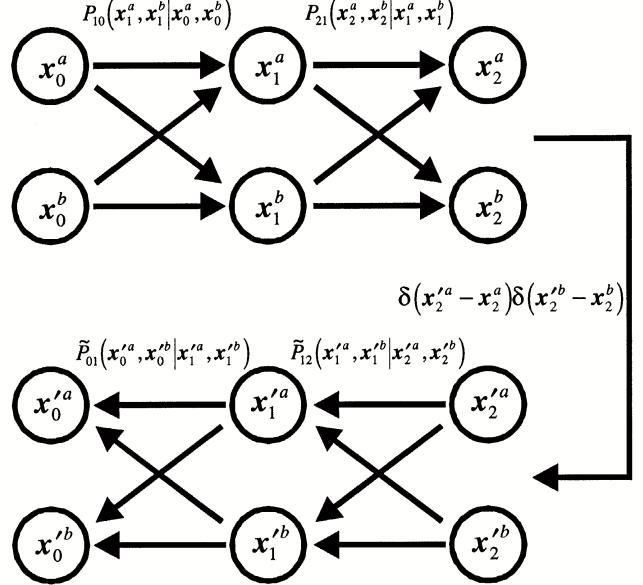


Figure 7: A 2-stage folded Markov chain with each state split into two lower dimensional pieces. This is basically the same as Figure 5, except that the states have been exploded to reveal their internal structure.

changes to the notation in Figure 5

$$\begin{aligned} x_0 &\rightarrow (x_0^a, x_0^b) & x'_0 &\rightarrow (x'_0^a, x'_0^b) \\ x_1 &\rightarrow (x_1^a, x_1^b) & x'_1 &\rightarrow (x'_1^a, x'_1^b) \\ x_2 &\rightarrow (x_2^a, x_2^b) & x'_2 &\rightarrow (x'_2^a, x'_2^b) \end{aligned} \quad (4.1)$$

so the FMC in Figure 7 is the same as the FMC in Figure 5, apart from the notation that is used to describe its state spaces. This tautology is motivated by the need to break up high dimensional spaces, such as might occur in image processing problems, into a number of coupled lower dimensional spaces. For instance, if the coupling between the FMCs is weak, then a series expansion about zero coupling strength could be attempted. The lowest order term in this expansion would correspond to a pair of uncoupled FMCs, and the higher order terms would correspond to interactions between the FMCs.

If the new notation in Equation 4.1 is used together with the generalisation of Equation 3.10

$$P_{1,0}(x_1^a, x_1^b | x_0^a, x_0^b) \rightarrow \delta[x_1^a - x_1^a(x_0^a, x_0^b)] \delta[x_1^b - x_1^b(x_0^a, x_0^b)] \quad (4.2)$$

in the expression for the distortion  $D$  in a 2-stage FMC (see Equation 3.17), and noting that  $P_{2,0}(\mathbf{x}_2|\mathbf{x}_0) = \int d\mathbf{x}_1 P_{2,1}(\mathbf{x}_2|\mathbf{x}_1) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ , then  $D$  reduces to

$$D = 2 \int d\mathbf{x}_0^a d\mathbf{x}_0^b P_0(\mathbf{x}_0^a, \mathbf{x}_0^b) \int d\mathbf{x}_2^a d\mathbf{x}_2^b P_{2,1}[\mathbf{x}_2^a, \mathbf{x}_2^b | \mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b), \mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)] \\ \times \left( \left\| \mathbf{x}_0^a - \int d\mathbf{u}_0^a \tilde{P}_{0,2}(\mathbf{u}_0^a | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^a \right\|^2 + \left\| \mathbf{x}_0^b - \int d\mathbf{u}_0^b \tilde{P}_{0,2}(\mathbf{u}_0^b | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^b \right\|^2 \right) \quad (4.3)$$


---

where the Euclidean error has been split into a sum of contributions from the  $a$  and the  $b$  subspaces of  $\mathbf{x}_0$ . The Euclidean error terms have an interesting structure. For instance, in FMC  $a$  the Euclidean distance between  $\mathbf{x}_0^a$  and  $\int d\mathbf{u}_0^a \tilde{P}_{0,2}(\mathbf{u}_0^a | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^a$  is computed, which explicitly depends via  $\tilde{P}_{0,2}(\mathbf{u}_0^a | \mathbf{x}_2^a, \mathbf{x}_2^b)$  on the outputs  $(\mathbf{x}_2^a, \mathbf{x}_2^b)$  of both FMC  $a$  and FMC  $b$ . In the previous section terms like  $\int d\mathbf{u}_0 \tilde{P}_{0,2}(\mathbf{u}_0 | \mathbf{x}_2) \mathbf{u}_0$  turned out to correspond to SOM reference vectors, so by analogy it is expected that  $\int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^k$  ( $k = a, b$ ) will also turn out to be the reference vectors for a pair of coupled SOMs corresponding to FMC  $a$  and FMC  $b$ .

Unfortunately, because of the coupling between FMC  $a$  and FMC  $b$ ,  $\int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^k$  ( $k = a, b$ ) depend on both  $\mathbf{x}_2^a$  and  $\mathbf{x}_2^b$ . The purpose of the following derivation is to find an approximate way of optimising  $D$  that does not involve these simultaneous dependencies on both  $\mathbf{x}_2^a$  and  $\mathbf{x}_2^b$ , but uses quantities like  $\int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^k) \mathbf{u}_0^k$  rather than  $\int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^k$  for  $k = a, b$ .

Thus each term can be split in the following manner for  $k = a, b$

$$\left\| \mathbf{x}_0^k - \int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^k \right\|^2 = \left\| \begin{array}{l} (\mathbf{x}_0^k - \int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^k) \mathbf{u}_0^k) \\ + \left( \int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^k) \mathbf{u}_0^k - \int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^k \right) \end{array} \right\|^2 \quad (4.4)$$

where the distinction between  $\int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^k$  and  $\int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^k) \mathbf{u}_0^k$  has been carefully used to separate the  $\mathbf{x}_0^k$  and  $\int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^k$  terms. The following two definitions are introduced for  $k = a, b$

$$D_0(k) \equiv 2 \int d\mathbf{x}_0^a d\mathbf{x}_0^b P_0(\mathbf{x}_0^a, \mathbf{x}_0^b) \int d\mathbf{x}_2^k P_{2,1}[\mathbf{x}_2^k | \mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b), \mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)] \\ \times \left\| \mathbf{x}_0^k - \int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^k) \mathbf{u}_0^k \right\|^2$$

$$D_1(k) \equiv 2 \int d\mathbf{x}_2^a d\mathbf{x}_2^b P_2(\mathbf{x}_2^a, \mathbf{x}_2^b) \left\| \int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^k) \mathbf{u}_0^k - \int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^k \right\|^2 \quad (4.5)$$

and used together with Bayes' theorem in the form

$$P_0(\mathbf{x}_0^a, \mathbf{x}_0^b) P_{2,1}[\mathbf{x}_2^a, \mathbf{x}_2^b | \mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b), \mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)] = \tilde{P}_{0,2}(\mathbf{x}_0^a, \mathbf{x}_0^b | \mathbf{x}_2^a, \mathbf{x}_2^b) P_2(\mathbf{x}_2^a, \mathbf{x}_2^b) \quad (4.6)$$


---

to obtain  $D$  in the form

$$D = D_0(a) + D_0(b) - D_1(a) - D_1(b) \quad (4.7)$$

The first two terms in Equation 4.7 have a structure that is simpler than in Equation 4.3. For instance  $D_0(a)$  depends on the Euclidean distance between  $\mathbf{x}_0^a$  and  $\int d\mathbf{u}_0^a \tilde{P}_{0,2}(\mathbf{u}_0^a | \mathbf{x}_2^a) \mathbf{u}_0^a$ , which depends only on the output of FMC  $a$  and not on the output of FMC  $b$ . An analogous remark applies to  $D_0(b)$ .

The last two terms in Equation 4.7 contain the undesirable terms such as  $\int d\mathbf{u}_0^a \tilde{P}_{0,2}(\mathbf{u}_0^a | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^a$ . In the

following derivation these terms will be discarded to obtain an approximate scheme for minimising  $D$ .

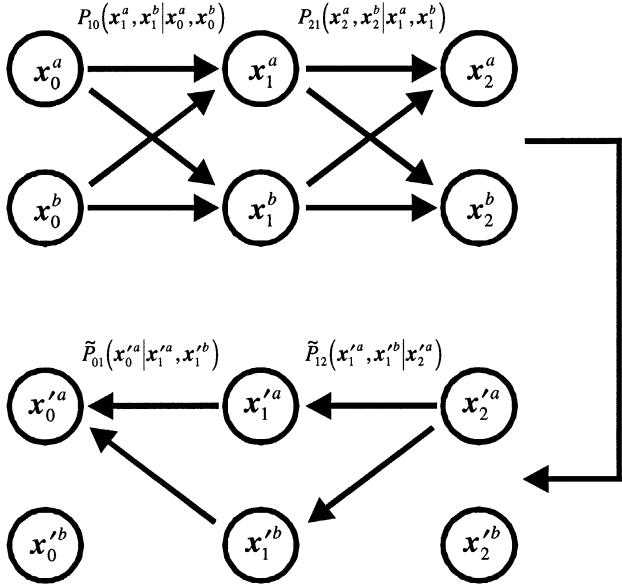


Figure 8: An approximation to a 2-stage folded Markov chain with each state split into two lower dimensional pieces. Only the  $\tilde{P}_{0,2}(\mathbf{x}_0^a | \mathbf{x}_2^a)$  part is indicated on the reverse part of the chain. The approximation arises because  $\mathbf{x}_2^a$ , rather than  $(\mathbf{x}_2^a, \mathbf{x}_2^b)$ , is used as the initial state on the reverse part of the chain. An analogous discussion holds for  $\tilde{P}_{0,2}(\mathbf{x}_0^b | \mathbf{x}_2^b)$ .

## B. Least upper bound optimisation scheme

Note that the following inequalities hold

$$\begin{aligned} D_0(a) &\geq 0 \\ D_0(b) &\geq 0 \\ D_1(a) &\geq 0 \\ D_1(b) &\geq 0 \\ D &\geq 0 \end{aligned} \quad (4.8)$$

These lead to the following constraint on  $D$

$$0 \leq D \leq D_0(a) + D_0(b) \quad (4.9)$$

Although ideally  $D$  itself should be minimised, it turns out to be much simpler to minimise its upper bound  $D_0(a) + D_0(b)$ . This least upper bound prescription achieves what is required, namely the elimination of the undesirable  $D_1(a)$  and  $D_1(b)$  terms which depend on  $\int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^a, \mathbf{x}_2^b) \mathbf{u}_0^k$  for  $k = a, b$ . This approximate prescription becomes exact in the limit where the coupling between the FMCs in Figure 7 tends to zero.

This approximate approach to optimising the FMC is shown in Figure 8. At this point it is appropriate to introduce a modified form of  $D_0(k)$  in which  $\int d\mathbf{u}_0^k \tilde{P}_{0,2}(\mathbf{u}_0^k | \mathbf{x}_2^k) \mathbf{u}_0^k$  in Equation 4.5 is replaced by the function  $\mathbf{x}_0'^k(\mathbf{x}_2^k)$  for  $k = a, b$

$$D_0(k) = 2 \int d\mathbf{x}_0^a d\mathbf{x}_0^b P_0(\mathbf{x}_0^a, \mathbf{x}_0^b) \int d\mathbf{x}_2^k P_{2,1}[\mathbf{x}_2^k | \mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b), \mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)] \|\mathbf{x}_0^k - \mathbf{x}_0'^k(\mathbf{x}_2^k)\|^2 \quad (4.10)$$

The functional derivative  $\frac{\delta D}{\delta \mathbf{x}_0'^k(\mathbf{x}_2^k)}$  then becomes for  $k = a, b$

$$\frac{\delta D_0(k)}{\delta \mathbf{x}_0'^k(\mathbf{x}_2^k)} = -4 \int d\mathbf{x}_0^a d\mathbf{x}_0^b P_0(\mathbf{x}_0^a, \mathbf{x}_0^b) \int d\mathbf{x}_2^k P_{2,1}[\mathbf{x}_2^k | \mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b), \mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)] [\mathbf{x}_0^k - \mathbf{x}_0'^k(\mathbf{x}_2^k)] \quad (4.11)$$

and the cross derivatives  $\frac{\delta D_0(a)}{\delta \mathbf{x}_0'^b(\mathbf{x}_2^b)}$  and  $\frac{\delta D_0(b)}{\delta \mathbf{x}_0'^a(\mathbf{x}_2^a)}$  are zero. The stationary point  $\frac{\delta D}{\delta \mathbf{x}_0'^k(\mathbf{x}_2^k)} = 0$  is obtained when  $\mathbf{x}_0'^k(\mathbf{x}_2^k)$  satisfies for  $k = a, b$

$$\mathbf{x}_0'^k(\mathbf{x}_2^k) = \frac{\int d\mathbf{x}_0^a d\mathbf{x}_0^b P_0(\mathbf{x}_0^a, \mathbf{x}_0^b) P_{2,1}[\mathbf{x}_2^k | \mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b), \mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)] \mathbf{x}_0^k}{\int d\mathbf{x}_0^a d\mathbf{x}_0^b P_0(\mathbf{x}_0^a, \mathbf{x}_0^b) P_{2,1}[\mathbf{x}_2^k | \mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b), \mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)]} \quad (4.12)$$

Note that this result can readily be generalised to an arbitrary choice of  $P_{1,0}(\mathbf{x}_1^a, \mathbf{x}_1^b | \mathbf{x}_0^a, \mathbf{x}_0^b)$  (i.e. not assuming Equation 4.2).

By using Bayes' theorem in the form shown in Equation 4.6 this reduces to

$$\mathbf{x}_0'^k(\mathbf{x}_2^k) = \int d\mathbf{x}_0^k \tilde{P}_{0,2}(\mathbf{x}_0^k | \mathbf{x}_2^k) \mathbf{x}_0^k \quad k = a, b \quad (4.13)$$

Thus the modified expression for  $D_0(k)$  in Equation 4.10

reduces to the original expression for  $D_0(k)$  in Equation 4.5 provided that  $\mathbf{x}_0'^k(\mathbf{x}_2^k)$  is chosen to minimise  $D_0(k)$ , so Equation 4.10 will be used in preference to Equation 4.5 (with the proviso that  $\mathbf{x}_0'^k(\mathbf{x}_2^k)$  should always be chosen to minimise  $D_0(k)$ ).

When the upper bound of  $D$  in Equation 4.9 is minimised with respect to  $\mathbf{x}_1^k(\mathbf{x}_0^a, \mathbf{x}_0^b)$  using Equation 4.10 it yields for  $k = a, b$

$$\mathbf{x}_1^k(\mathbf{x}_0^a, \mathbf{x}_0^b) = \arg \min_{\mathbf{x}_1^k} \left( \int d\mathbf{x}_2^a P_{2,1}(\mathbf{x}_2^a | \mathbf{x}_1^a, \mathbf{x}_1^b) \|\mathbf{x}_0^a - \mathbf{x}_0'^a(\mathbf{x}_2^a)\|^2 + \int d\mathbf{x}_2^b P_{2,1}(\mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b) \|\mathbf{x}_0^b - \mathbf{x}_0'^b(\mathbf{x}_2^b)\|^2 \right) \quad (4.14)$$

and the following on-line training prescription for  $k = a, b$

$$\mathbf{x}_0'^k(\mathbf{x}_2^k) \longrightarrow \mathbf{x}_0'^k(\mathbf{x}_2^k) + \epsilon P_{2,1}[\mathbf{x}_2^k | \mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b), \mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)] [\mathbf{x}_0^k - \mathbf{x}_0'^k(\mathbf{x}_2^k)] \quad (4.15)$$

These results correspond to the self-supervised training scheme that was proposed in [12, 13], which was the result of a detailed study made of the problem of designing an encoder/decoder for a pair of communication channels whose transmitted information was degraded by a noise process (both external noise and noisy coupling between the channels). The improvement in performance when channel coupling is taken into account was shown to be significant, so the least upper bound approximation is justified in hindsight for this type of system.

The following remarks can be made about these results:

1. The functions  $\mathbf{x}_0'^a(\mathbf{x}_2^a)$  and  $\mathbf{x}_0'^b(\mathbf{x}_2^b)$  are the continuum versions of the reference vectors of a pair of SOMs corresponding to FMC  $a$  and FMC  $b$ , respectively.
2. The results for  $\mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b)$  and  $\mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)$  are modified forms of the minimum distortion encoding prescription, in which the coupling between the FMC  $a$  and FMC  $b$  manifests itself through  $P_{2,1}(\mathbf{x}_2^a | \mathbf{x}_1^a, \mathbf{x}_1^b)$  and  $P_{2,1}(\mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$ .
3. In both the batch and the on-line training prescription  $P_{2,1}(\mathbf{x}_2^a | \mathbf{x}_1^a, \mathbf{x}_1^b)$  and  $P_{2,1}(\mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$  play the role of neighbourhood functions for SOM  $a$  and SOM  $b$ , respectively. However, the coupling between the SOMs causes these neighbourhood functions to be dependent on the input data, as will be discussed in more detail below.

### C. Network representation of self-supervision

In Figure 9 the network representation of the pair of coupled SOMs is shown for a discrete-valued output.

The detailed interpretation of Equation 4.14 and Equation 4.15 is shown in Figure 10.

The marginal probabilities  $P_{2,1}(\mathbf{x}_2^a | \mathbf{x}_1^a, \mathbf{x}_1^b)$  and  $P_{2,1}(\mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$  are obtained by projecting  $P_{2,1}(\mathbf{x}_2^a, \mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$  onto  $\mathbf{x}_2^a$  and  $\mathbf{x}_2^b$ , respectively. The data dependence of  $P_{2,1}(\mathbf{x}_2^a, \mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$  causes these marginal probabilities to be data dependent. In particular, they can be biased as shown in Figure 10, which causes the neighbourhood functions (in the SOM interpretation) in  $\mathbf{x}_2^a$  and  $\mathbf{x}_2^b$  space to be biased.

The data dependence has another subtle side effect in Figure 10. The input transformations  $\mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b)$

and  $\mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)$  depend on the marginal probabilities  $P_{2,1}(\mathbf{x}_2^a | \mathbf{x}_1^a, \mathbf{x}_1^b)$  and  $P_{2,1}(\mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$ , because the input transformations satisfy a minimum distortion criterion which depends on these marginal probabilities (see Equation 4.14). However, the marginal probabilities themselves are data dependent, because they depend on  $\mathbf{x}_1^a$  and  $\mathbf{x}_1^b$ , which in turn depend on the input transformations. Overall, the marginal probabilities and the input transformations are mutually dependent, which makes the minimum distortion encoding prescription quite subtle to implement in this case.

Further details on self-supervision can be found in [12, 13], where a detailed discussion and numerical simulation of the consequences of using a particular type of  $P_{2,1}(\mathbf{x}_2^a, \mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$  are presented, a comparison is made between nearest neighbour and minimum distortion encoding, and a comparison is made between using mutually dependent neighbourhood functions  $P_{2,1}(\mathbf{x}_2^k | \mathbf{x}_1^a, \mathbf{x}_1^b)$  ( $k = a, b$ ) and independent neighbourhood functions  $P_{2,1}(\mathbf{x}_2^a | \mathbf{x}_1^a)$  and  $P_{2,1}(\mathbf{x}_2^b | \mathbf{x}_1^b)$ .

## V. CONCLUSIONS

In this paper it has been demonstrated that VQ theory, SOM theory, and the theory of self-supervision all emerge naturally when an FMC is optimised so as to minimise the expected Euclidean error between an input vector and its attempted reconstruction (using Bayes' theorem).

FMC theory can be used to facilitate many computations that would otherwise be theoretically and/or numerically intractable. The “soft” probabilities that are used in the FMC are easier to compute with than the “hard” delta functions in the corresponding winner-take-all VQs and SOMs. The results contained in this paper guarantee that these “soft” computations reduce to the required “hard” computations when the first stage of the FMC is optimised.

## VI. ACKNOWLEDGEMENTS

The author is indebted to the following people for critically reading this paper: Eric Jakeman, David Lowe, Graeme Mitchison.

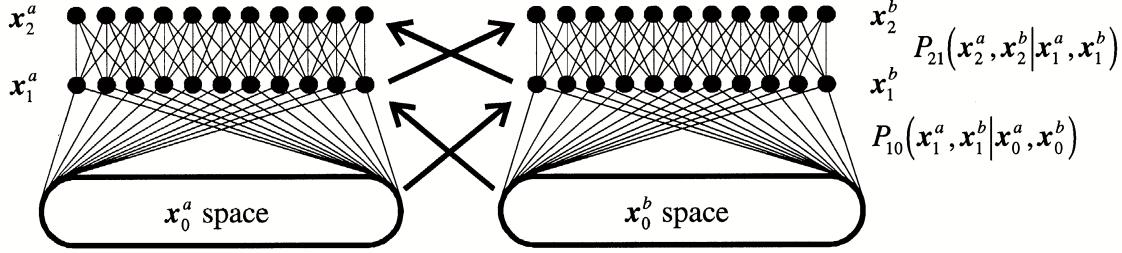


Figure 9: A pair of coupled discrete SOMs represented as a pair of coupled networks of the type shown in Figure 6. As derived above, the coupling  $P_{2,1}(\mathbf{x}_2^a, \mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$  between networks  $a$  and  $b$  expresses itself as a pair of data dependent neighbourhood functions  $P_{2,1}(\mathbf{x}_2^a | \mathbf{x}_1^a, \mathbf{x}_1^b)$  and  $P_{2,1}(\mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$ . When the first stage of each of these coupled SOMs is optimised, taking account of  $P_{2,1}(\mathbf{x}_2^a | \mathbf{x}_1^a, \mathbf{x}_1^b)$  and  $P_{2,1}(\mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$ , the encoding functions  $\mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b)$  and  $\mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)$  become mutually coupled.

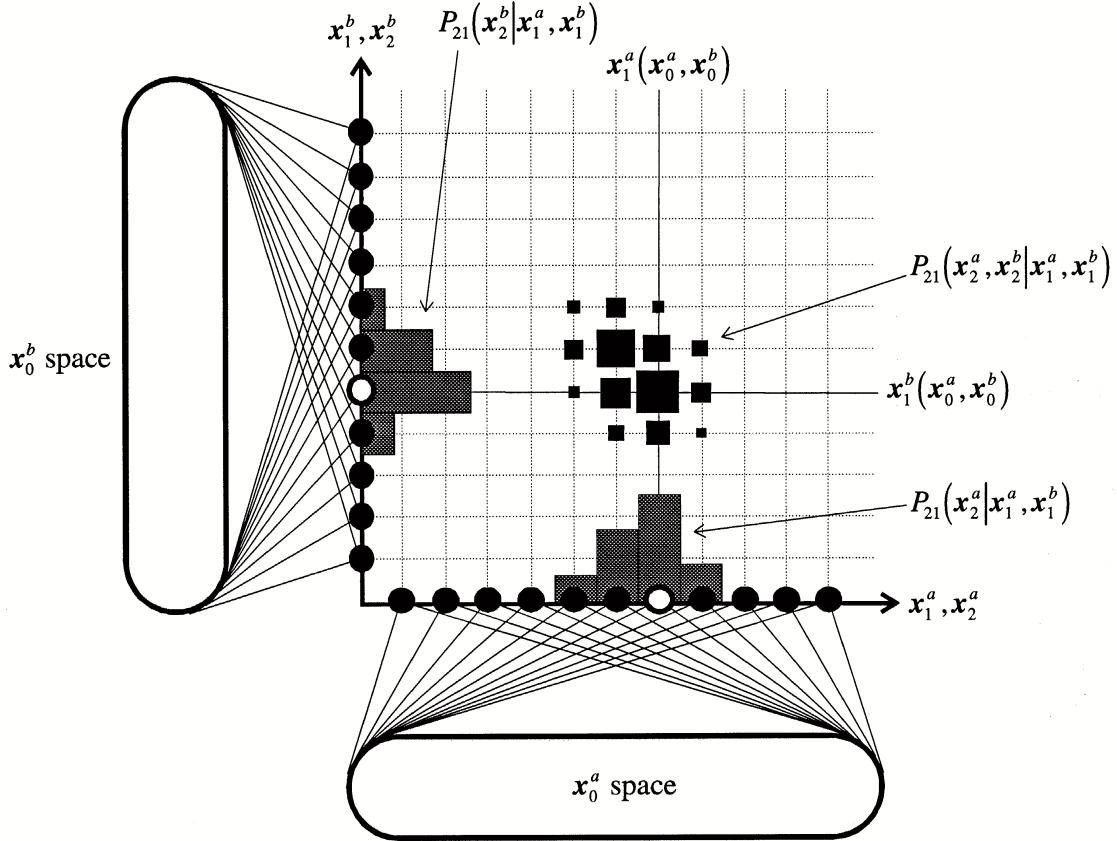


Figure 10: Diagram showing the detailed operation of a pair of coupled FMCs. For simplicity,  $(\mathbf{x}_1^a, \mathbf{x}_1^b)$  is assumed to sit in the same vector space as  $(\mathbf{x}_0^a, \mathbf{x}_0^b)$ . Examples of the winners  $\mathbf{x}_1^a(\mathbf{x}_0^a, \mathbf{x}_0^b)$  and  $\mathbf{x}_1^b(\mathbf{x}_0^a, \mathbf{x}_0^b)$  are indicated by the open circles. These are then jointly smeared into the distribution  $P_{2,1}(\mathbf{x}_2^a, \mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$ , which is drawn as a 2-dimensional histogram. As derived above, the optimisation depends on a pair of neighbourhood functions  $P_{2,1}(\mathbf{x}_2^a | \mathbf{x}_1^a, \mathbf{x}_1^b)$  and  $P_{2,1}(\mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$ , which are the marginal probabilities of  $P_{2,1}(\mathbf{x}_2^a, \mathbf{x}_2^b | \mathbf{x}_1^a, \mathbf{x}_1^b)$ , and which are drawn as 1-dimensional histograms in the diagram. These data-dependent neighbourhood functions not only influence the optimisation of the FMC, but also determine the winners that should have been used in the first place.

#### Appendix A: Relationship between continuum and discrete vector quantisers

Throughout this paper continuum notation is used. In the case of a VQ this has the effect that the index that is

used to select the winning reference vector is assumed to be a continuous-valued quantity, rather than a discrete-valued quantity. The purpose of this appendix is to relate the continuum case to the discrete case.

It is sufficient to discuss the meaning of the on-line

training prescription in Equation 3.13 and Equation 3.14, which is presented again here for convenience.

$$\begin{aligned} \mathbf{x}_1(\mathbf{x}_0) &= \arg \min_{\mathbf{x}_1} \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_1)\|^2 \\ \mathbf{x}'_0(\mathbf{x}_1) &\longrightarrow \mathbf{x}'_0(\mathbf{x}_1) + \epsilon \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)] [\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_1)] \end{aligned} \quad (\text{A1})$$

where  $\mathbf{x}_1$  is assumed to lie in the unit  $N$ -dimensional hypercube. The discrete counterpart of this on-line prescription is

$$\begin{aligned} \mathbf{k}_1(\mathbf{x}_0) &= \arg \min_{\mathbf{k}_1} \|\mathbf{x}_0 - \mathbf{x}'_{0,\mathbf{k}_1}\|^2 \\ \mathbf{x}'_{0,\mathbf{k}_1} &\longrightarrow \mathbf{x}'_{0,\mathbf{k}_1} + \epsilon \delta_{\mathbf{k}_1, \mathbf{k}_1(\mathbf{x}_0)} (\mathbf{x}_0 - \mathbf{x}'_{0,\mathbf{k}_1}) \end{aligned} \quad (\text{A2})$$

where  $\mathbf{k}_1$  is a point on an  $N$ -dimensional cubic lattice.

The relationship between the nearest neighbour encoding prescriptions  $\mathbf{x}_1(\mathbf{x}_0)$  and  $\mathbf{k}_1(\mathbf{x}_0)$  is simple to understand, so no further comment is required. On the other hand, the interpretation of the update prescription for  $\mathbf{x}'_0(\mathbf{x}_1)$  is quite subtle, so it will now be discussed in some detail.

In order to interpret the continuum prescription correctly it is necessary to integrate over a small region  $S$  with volume  $\delta V$  that encloses the point  $\mathbf{x}_1 = \mathbf{x}_1(\mathbf{x}_0)$ , and to assume that  $\mathbf{x}'_0(\mathbf{x}_1)$  is a smooth function of  $\mathbf{x}_1$  (for a justification of this assumption, see the last paragraph of this Appendix). This leads to the result

$$\mathbf{x}'_0(\mathbf{x}_1) \longrightarrow \mathbf{x}'_0(\mathbf{x}_1) + \frac{\epsilon}{\delta V} [\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{x}_1)] \quad \text{where } \mathbf{x}_1 \in S \text{ and } \mathbf{x}_1(\mathbf{x}_0) \in S \quad (\text{A3})$$

which can be compared directly with the update prescription for  $\mathbf{x}'_{0,\mathbf{k}_1}$ . Note that  $\frac{\epsilon}{\delta V}$ , rather than  $\epsilon$ , determines the size of the updates of the reference vectors that are attached to points  $\mathbf{x}_1$  inside  $S$ . The smaller the volume  $\delta V$  is, the smaller  $\epsilon$  has to be to keep the updates the same size. The most natural prescription is to impose a lower bound  $\delta L$  on the length scale of  $\mathbf{x}_1$  of interest, which then imposes a natural size on the volume of  $S$  such that  $\delta V = \delta L^N$ . Since  $\mathbf{x}_1$  is assumed to lie in the unit hypercube, then this prescription leads to a finite effective number of reference vectors given by  $\delta L^{-N}$ , which would correspond to an  $N$ -dimensional cubic lattice whose size was  $\frac{1}{\delta L}$  lattice spacings in each dimension.

An interesting side effect of introducing this inner length scale  $\delta L$  (which is effectively a regularisation constant) is that it automatically generates a SOM on-line training prescription. To see this, note that in Equation A3 a whole volume  $\delta V$  of reference vectors  $\mathbf{x}_1$  in the neighbourhood  $S$  of the point  $\mathbf{x}_1 = \mathbf{x}_1(\mathbf{x}_0)$  is updated. This is precisely the type of update prescription that creates a SOM. Furthermore, this justifies with hindsight the assumption made earlier that  $\mathbf{x}'_0(\mathbf{x}_1)$  is a smooth function of  $\mathbf{x}_1$ .

## Appendix B: Full optimisation of 2-stage folded Markov chains

In this appendix it will be shown that if a 2-stage FMC with a Euclidean error function is optimised with respect to  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ , then it reduces to  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) = \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$ . Because a 2-stage FMC contains a 1-stage FMC as a special case, it is necessary to present the derivation only for a 2-stage FMC. The result that is obtained depends on certain properties of the Euclidean error function; it is not a general property of FMCs.

There are two constraints that must be respected during the optimisation of  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ . Total probability must be conserved

$$\int d\mathbf{x}_1 P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) = 1 \quad \text{for all } \mathbf{x}_0 \quad (\text{B1})$$

and  $\tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1)$  must vary in response to variations of  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  in such a way that Bayes' theorem is respected (i.e Equation 2.1 with  $k = 0$ )

$$\tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1) \left[ \int d\mathbf{u}_0 P_{1,0}(\mathbf{x}_1|\mathbf{u}_0) P_0(\mathbf{u}_0) \right] = P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) P_0(\mathbf{x}_0) \quad \text{for all } \mathbf{x}_0, \mathbf{x}_1 \quad (\text{B2})$$

The first constraint will be imposed for all  $\mathbf{x}_0$  by using a Lagrange multiplier function  $\lambda(\mathbf{x}_0)$ , whereas the second constraint will be imposed by using Bayes' theorem to eliminate  $\tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1)$  from  $D$ .

The overall quantity  $\overline{D}$  to be minimised is therefore

$$\begin{aligned} \overline{D} = & \int d\mathbf{x}_0 d\mathbf{x}_1 d\mathbf{x}_2 d\mathbf{x}'_0 d\mathbf{x}'_1 d\mathbf{x}'_2 P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) P_{2,1}(\mathbf{x}_2|\mathbf{x}_1) \delta(\mathbf{x}'_2 - \mathbf{x}_2) \\ & \times \tilde{P}_{1,2}(\mathbf{x}'_1|\mathbf{x}'_2) \tilde{P}_{0,1}(\mathbf{x}'_0|\mathbf{x}'_1) \|\mathbf{x}'_0 - \mathbf{x}_0\|^2 \\ & - \int d\mathbf{x}_0 \lambda(\mathbf{x}_0) \int d\mathbf{x}_1 P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \end{aligned} \quad (\text{B3})$$

Integration over  $\mathbf{x}_2$  and use of Bayes' theorem on both  $\tilde{P}_{1,2}(\mathbf{x}'_1|\mathbf{x}'_2)$  and  $\tilde{P}_{0,1}(\mathbf{x}'_0|\mathbf{x}'_1)$  transforms  $\overline{D}$  into a form that is suitable for differentiation with respect to the sought after quantity  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$

$$\begin{aligned} \overline{D} = & \int d\mathbf{x}_0 d\mathbf{x}_1 d\mathbf{x}'_0 d\mathbf{x}'_1 d\mathbf{x}'_2 P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) P_{1,0}(\mathbf{x}'_1|\mathbf{x}'_0) P_0(\mathbf{x}'_0) \\ & \times \frac{P_{2,1}(\mathbf{x}'_2|\mathbf{x}_1) P_{2,1}(\mathbf{x}'_2|\mathbf{x}'_1)}{\int d\mathbf{u}_0 d\mathbf{u}_1 P_{2,1}(\mathbf{x}'_2|\mathbf{u}_1) P_{1,0}(\mathbf{u}_1|\mathbf{u}_0) P_0(\mathbf{u}_0)} \|\mathbf{x}'_0 - \mathbf{x}_0\|^2 \\ & - \int d\mathbf{x}_0 \lambda(\mathbf{x}_0) \int d\mathbf{x}_1 P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \end{aligned} \quad (\text{B4})$$

Functional differentiation of  $\overline{D}$  with respect to  $P_{1,0}(\mathbf{z}_1|\mathbf{z}_0)$ , and using the symmetry of  $\overline{D}$  under interchange of  $\mathbf{x}_0$  and  $\mathbf{x}'_0$ , then leads to

$$\begin{aligned} \frac{\delta \overline{D}}{\delta P_{1,0}(\mathbf{z}_1|\mathbf{z}_0)} = & 2 \int d\mathbf{x}_0 d\mathbf{x}_1 d\mathbf{x}'_0 d\mathbf{x}'_1 d\mathbf{x}'_2 \delta(\mathbf{x}_0 - \mathbf{z}_0) \delta(\mathbf{x}_1 - \mathbf{z}_1) P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}'_1|\mathbf{x}'_0) P_0(\mathbf{x}'_0) \\ & \times \frac{P_{2,1}(\mathbf{x}'_2|\mathbf{x}_1) P_{2,1}(\mathbf{x}'_2|\mathbf{x}'_1)}{\left( \int d\mathbf{u}_0 d\mathbf{u}_1 P_{2,1}(\mathbf{x}'_2|\mathbf{u}_1) P_{1,0}(\mathbf{u}_1|\mathbf{u}_0) P_0(\mathbf{u}_0) \right)} \|\mathbf{x}'_0 - \mathbf{x}_0\|^2 \\ & - \int d\mathbf{x}_0 d\mathbf{x}_1 d\mathbf{x}'_0 d\mathbf{x}'_1 d\mathbf{x}'_2 P_0(\mathbf{x}_0) P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) P_{1,0}(\mathbf{x}'_1|\mathbf{x}'_0) P_0(\mathbf{x}'_0) \\ & \times \frac{P_{2,1}(\mathbf{x}'_2|\mathbf{x}_1) P_{2,1}(\mathbf{x}'_2|\mathbf{x}'_1)}{\left[ \int d\mathbf{u}_0 d\mathbf{u}_1 P_{2,1}(\mathbf{x}'_2|\mathbf{u}_1) P_{1,0}(\mathbf{u}_1|\mathbf{u}_0) P_0(\mathbf{u}_0) \right]^2} \\ & \times \left[ \int d\mathbf{v}_0 d\mathbf{v}_1 \delta(\mathbf{v}_0 - \mathbf{z}_0) \delta(\mathbf{v}_1 - \mathbf{z}_1) P_{2,1}(\mathbf{x}'_2|\mathbf{v}_1) P_0(\mathbf{v}_0) \right] \|\mathbf{x}'_0 - \mathbf{x}_0\|^2 \\ & - \int d\mathbf{x}_0 \lambda(\mathbf{x}_0) \int d\mathbf{x}_1 \delta(\mathbf{x}_0 - \mathbf{z}_0) \delta(\mathbf{x}_1 - \mathbf{z}_1) \end{aligned} \quad (\text{B5})$$

Using Bayes' theorem this simplifies to

$$\begin{aligned} \frac{\delta \overline{D}}{\delta P_{1,0}(\mathbf{z}_1|\mathbf{z}_0)} = & 2P_0(\mathbf{z}_0) \int d\mathbf{x}'_2 P_{2,1}(\mathbf{x}'_2|\mathbf{z}_1) \int d\mathbf{x}'_1 \tilde{P}_{1,2}(\mathbf{x}'_1|\mathbf{x}'_2) \\ & \times \int d\mathbf{x}'_0 \tilde{P}_{0,1}(\mathbf{x}'_0|\mathbf{x}'_1) \|\mathbf{x}'_0 - \mathbf{z}_0\|^2 \\ & - P_0(\mathbf{z}_0) \int d\mathbf{x}'_2 P_{2,1}(\mathbf{x}'_2|\mathbf{z}_1) \int d\mathbf{x}_1 \tilde{P}_{1,2}(\mathbf{x}_1|\mathbf{x}'_2) \int d\mathbf{x}'_1 \tilde{P}_{1,2}(\mathbf{x}'_1|\mathbf{x}'_2) \\ & \times \int d\mathbf{x}_0 d\mathbf{x}'_0 \tilde{P}_{0,1}(\mathbf{x}_0|\mathbf{x}_1) \tilde{P}_{0,1}(\mathbf{x}'_0|\mathbf{x}'_1) \|\mathbf{x}'_0 - \mathbf{x}_0\|^2 \\ & - \lambda(\mathbf{z}_0) \end{aligned} \quad (\text{B6})$$

Expansion and factorisation eventually lead to the result

$$\frac{\delta \overline{D}}{\delta P_{1,0}(\mathbf{z}_1|\mathbf{z}_0)} = 2P_0(\mathbf{z}_0) \int d\mathbf{x}'_2 P_{2,1}(\mathbf{x}'_2|\mathbf{z}_1) \left\| \int d\mathbf{x}'_0 d\mathbf{x}'_1 \tilde{P}_{0,1}(\mathbf{x}'_0|\mathbf{x}'_1) \tilde{P}_{1,2}(\mathbf{x}'_1|\mathbf{x}'_2) \mathbf{x}'_0 - \mathbf{z}_0 \right\|^2 - \lambda(\mathbf{z}_0) \quad (\text{B7})$$

The stationarity condition  $\frac{\delta \overline{D}}{\delta P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)} = 0$  is therefore

$$2P_0(\mathbf{x}_0) \int d\mathbf{x}'_2 P_{2,1}(\mathbf{x}'_2|\mathbf{x}_1) \left\| \int d\mathbf{x}'_0 \tilde{P}_{0,2}(\mathbf{x}'_0|\mathbf{x}'_2) \mathbf{x}'_0 - \mathbf{x}_0 \right\|^2 = \lambda(\mathbf{x}_0) \quad (\text{B8})$$

---

The right hand side of Equation B8 is a function only of  $\mathbf{x}_0$ , whereas the left hand side is a function of both

$\mathbf{x}_0$  and  $\mathbf{x}_1$ , where  $\mathbf{x}_1$  appears only in the  $P_{2,1}(\mathbf{x}'_2|\mathbf{x}_1)$  factor. Because  $\mathbf{x}_1$  appears only on the left hand side of this equation, its effect must somehow vanish. There are two types of solution in which the dependence on  $\mathbf{x}_1$  vanishes:

1.  $P_0(\mathbf{x}_0) = 0$ . The dependence of the left hand side of Equation B8 is suppressed by the zero-valued  $P_0(\mathbf{x}_0)$  factor. Any legal probability  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  is thus permitted for values of  $\mathbf{x}_0$  that have  $P_0(\mathbf{x}_0) = 0$ . This solution may be eliminated because for those  $\mathbf{x}_0$  that have  $P_0(\mathbf{x}_0) = 0$  there is no need to define  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  anyway.
2.  $P_0(\mathbf{x}_0) > 0$ . Note that the variance-like factor  $\|\int d\mathbf{x}'_0 \tilde{P}_{0,2}(\mathbf{x}'_0|\mathbf{x}'_2) \mathbf{x}'_0 - \mathbf{x}_0\|^2 \geq 0$ , so there are two situations to consider:
  - (a)  $\|\cdots\|^2 > 0$ . In order to suppress the  $\mathbf{x}_1$  dependence of the left hand side of Equation B8  $\mathbf{x}_1$  must be uniquely determined by the value of  $\mathbf{x}_0$ . So  $\mathbf{x}_1 = \mathbf{x}_1(\mathbf{x}_0)$  and  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) = \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$ .
  - (b)  $\|\cdots\|^2 = 0$ . Apparently, any legal probability  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  is permitted. However, any  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \neq \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$  will guarantee that the variance-like factor  $\|\cdots\|^2 > 0$ , so the only  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  that can possibly survive are  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) = \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$ . Note that with  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) = \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$  it is still possible that  $\|\cdots\|^2 > 0$ , in which case this solution can be eliminated.

The only solution that remains is  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) = \delta[\mathbf{x}_1 -$

$\mathbf{x}_1(\mathbf{x}_0)]$ . This result establishes the fact that the replacement of  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  by  $\delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$  used in Equation 3.10 (and in Equation 4.2, in the case of coupled FMCs) emerges naturally from minimising  $D$  in the function space of probabilities  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ .

It is important to note that the choice of a Euclidean error function is sufficient (but not necessary) for  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) = \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$  to emerge. Also, note that it is not true in general that optimisation of  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  leads to  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) = \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$ . For instance, this is the case when  $\|\mathbf{x}_0 - \mathbf{x}'_0\|^2$  is replaced by either of the following two functional forms

$$\|\mathbf{x}_0 - \mathbf{x}'_0\|^2 \longrightarrow \begin{cases} A(\mathbf{x}_0) + B(\mathbf{x}'_0) & \text{counterexample 1} \\ A(\mathbf{x}_0) B(\mathbf{x}'_0) & \text{counterexample 2} \end{cases} \quad (B9)$$

Although these might not be considered to be sensible error functions, the fact that counterexamples exist is in itself important. These counterexamples may be described briefly as follows:

1. Counterexample 1 leads to a  $D$  which has no dependence on  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ . Optimisation of  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  will allow any legal probability, so  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \neq \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$  is permitted.
2. Counterexample 2 is more complicated to analyse. However, it is possible to show that  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$ , when viewed as a matrix, has a block diagonal structure. This type of  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0)$  does not imply a deterministic relationship between  $\mathbf{x}_0$  and  $\mathbf{x}_1$ , so  $P_{1,0}(\mathbf{x}_1|\mathbf{x}_0) \neq \delta[\mathbf{x}_1 - \mathbf{x}_1(\mathbf{x}_0)]$  is permitted.

- 
- [1] N Farvardin, *A study of vector quantisation for noisy channels*, IEEE Transactions on Information Theory **36** (1990), no. 4, 799–809.
- [2] N Farvardin and V Vaishampayan, *On the performance and complexity of channel-optimised VQs*, IEEE Transactions on Information Theory **37** (1991), no. 1, 155–160.
- [3] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [4] H Kumazawa, M Kasahara, and T Namekawa, *A construction of vector quantisers for noisy channels*, Electronics and Engineering in Japan **67** (1984), no. 4, 39–47.
- [5] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [6] S P Luttrell, *Hierarchical self-organising networks*, Proceedings of IEE Conference on Artificial Neural Networks (London), IEE, 1989, pp. 2–6.
- [7] ———, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
- [8] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
- [9] ———, *Self-organisation: a derivation from first principles of a class of learning algorithms*, Proceedings of IJCNN international joint conference on neural networks (Washington DC), IEEE, 1989, pp. 495–498.
- [10] ———, *Derivation of a class of training algorithms*, IEEE Transactions on Neural Networks **1** (1990), no. 2, 229–232.
- [11] ———, *Code vector density in topographic mappings: scalar case*, IEEE Transactions on Neural Networks **2** (1991), no. 4, 427–436.
- [12] ———, *Self-supervised training of hierarchical vector quantisers*, Proceedings of IEE conference on artificial neural networks (Bournemouth), IEE, 1991, pp. 5–9.
- [13] ———, *Self-supervised adaptive networks*, Proceedings of the IEE F **139** (1992), no. 6, 371–377.
- [14] H Ritter, *Asymptotic level density for a class of vector quantisation processes*, IEEE Transactions on Neural Networks **2** (1991), no. 1, 173–175.
- [15] E Yair, K Zeger, and A Gersho, *Competitive learning and soft competition for vector quantiser design*, IEEE Transactions on Signal Processing **40** (1992), no. 2, 294–309.

# Self-Organisation of Multiple Winner-take-all Neural Networks \*

STEPHEN P. LUTTRELL

*Defence Research Agency, St Andrews Road, Malvern, Worcs WR14 3PS, UK.*

In this paper, analysis of the information content of discretely firing neurons in unsupervised neural networks is presented, where information is measured according to the network's ability to reconstruct its input from its output with minimum mean square Euclidean error. It is shown how this type of network can self-organise into multiple winner-take-all subnetworks, each of which tackles only a low-dimensional subspace of the input vector. This is a rudimentary example of a neural network that effectively subdivides a task into manageable subtasks.

## I. INTRODUCTION

The purpose of this paper is to investigate the problem of subdividing a task into separate and easily computable pieces. This problem has been recently studied in [5] and [19], where an autoencoder (i.e. input vector  $\rightarrow$  coded version of input  $\rightarrow$  reconstruction of input vector) is optimised according to a minimum description length (MDL) optimisation principle [17]. The MDL approach requires that the total amount of information that needs to be transmitted from encoder to decoder is minimised, which leads automatically to optimal networks which factorise the encoder/decoder into a number of parts, each of which deals with a separate component of the input. The MDL objective function leads to a trade-off between two extreme approaches: a single neuron fires (i.e. winner-take-all) in response to the input which requires very little bandwidth to communicate to the decoder and the reconstruction of the input is inaccurate, or alternately many neurons fire (perhaps a whole pattern of activity is observed) in response to the input which requires a large bandwidth to communicate to the decoder and the reconstruction of the input is accurate. The MDL approach minimises an objective function that is a sum of three terms: a code cost, a reconstruction cost and a model cost. In [5] and [19], the model cost is ignored, so the optimal network minimises an objective function that is a sum of code and reconstruction costs. Principal components analysis (PCA) and vector quantisers (VQs) both emerge from the MDL approach if an appropriate upper bound is placed on the code cost.

The MDL approach is not unique in its ability to subdivide a task into separate and easily computable pieces. The author has studied a variety of problems of this type [9–12, 15], where the input space of an adaptive network is manually subdivided into a hierarchically nested set of subspaces; the purpose of this paper is to extend this approach so that the subdivision of the input spaces emerges spontaneously. This approach may be interpreted within the MDL framework as an autoencoder

in which the model cost is ignored, the code cost is held constant, and the reconstruction cost is a sum of squared differences. Although this approach is a special case of the MDL approach in which only the reconstruction cost may be varied, it nevertheless yields networks which are still able to factorise the encoder/decoder into a number of parts. The key to this ability is the coding scheme that is used, which is the set of locations of neurons that are observed to ‘fire’ in the coding layer of the network. Two extreme cases make contact with earlier results: a VQ emerges if one neuron fires (i.e. there is a small code cost), and PCA emerges if a large number of neurons fire (i.e. there is a large code cost). A more general scheme in which the number of observed firing neurons is variable (i.e. there is a variable code cost) would more properly be described as an MDL approach; this will not be studied in this paper.

In Section II, a probabilistic framework for modelling a neural network (NN) in which the output neurons fire discretely in response to the presentation of an input will be presented. In Section III, this theory will be used to derive optimal networks for a variety of simple training scenarios. These scenarios will be chosen to best illustrate the emergent network property that we seek: namely, the subdivision of a task into manageable pieces. To this end, a simple winner-take-all neuron firing model will be assumed, and the training set will be assumed to have a symmetric low-dimensional structure that lends itself well to analytic calculations. Finally, in Section IV, the theoretical results obtained in Section III are interpreted. This reveals many interesting properties, among which is the ability of a winner-take-all NN that is modelled as described in Section II to self-organise in such a way that it subdivides a task into manageable subtasks. From Section II D onwards, the material presented in this paper is new.

## II. THEORY

In Section II A, a summary will be given of the basic probabilistic framework which models how a layer of neurons fires in response to an input vector; this has been described in detail in [14, 16]. In Section II B, this will be applied to problems in which only a single neural firing event is observed, and in Section II C this will be ex-

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 11, 2019.

This paper appeared in Connection Science, 1997, vol. 9, no. 1, pp. 11-30. © 1997 Crown Copyright.

tended to the case of multiple neural firing events. This generalisation is necessary in order to obtain a network objective function that leads to the required emergent network properties when it is minimised. In Section II D the parameters of the network will be optimised, and the corresponding optimal network objective function will be derived.

### A. Probabilistic model of neural firing events

The model will consist of two layers of neurons. The input layer has a pattern of activity that represents the components of the input vector  $\mathbf{x}$ , and the output layer consists of  $M$  neurons labelled by an index  $y$  ( $y = 1, 2, \dots, M$ ) which has a pattern of activity that represents the average firing rate of each output neuron. If an input vector  $\mathbf{x}$  is presented to this network, then the output neurons fire discretely in response to the input. The pattern of activity is the time average of all of the individual neural firing rates, which does not preserve any information about correlations between the firing events. If it is assumed that the timing of the firing events is ignored, then after  $n$  neurons have fired the probabilistic description of the relationship between the input and output of the network is given by  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$ , where  $y_i$  is the location in the output layer of the  $i$ th neuron that fires. In this paper, it will also be assumed that the order in which the  $n$  neurons fire is not observed, in which case  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  is a sum of probabilities over all  $n!$  permutations of  $(y_1, y_2, \dots, y_n)$ , so it is a symmetric function of the  $y_i$ .

In the  $n = 1$  case, the probabilistic description  $\Pr(y|\mathbf{x})$  is proportional to the firing rate of neuron  $y$  in response to input  $\mathbf{x}$ , where the proportionality constant depends on the firing rates of all of the neurons. When  $n > 1$  there is an indirect relationship between the (symmetric) probabilistic description  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$ , and  $\Pr(y|\mathbf{x})$ , which is given by the marginal probability

$$\Pr(y|\mathbf{x}) = \sum_{y_2, \dots, y_n=1}^M \Pr(y_1, y_2, \dots, y_n | \mathbf{x}) \quad (2.1)$$

It is important to maintain this distinction between the events that are observed (i.e. how the firing events  $(y_1, y_2, \dots, y_n)$  arise from the input  $\mathbf{x}$ ) and the probabilistic description of the same thing (i.e.  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$ ). The detailed calculation of  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  depends on an explicit model of how the firing events are generated, but as will be shown in this paper many useful results can be derived without needing to specify the exact details of how the  $(y_1, y_2, \dots, y_n)$  firing events arise. In the  $n \rightarrow \infty$  limit,  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  has all of its probability concentrated in the vicinity of those  $(y_1, y_2, \dots, y_n)$  that are consistent with the observed long-term average firing rate of each neuron.

### B. Single firing event model

A theory of self-organising networks based on an analysis of a probabilistic encoder/decoder was presented in [13]; it dealt with the  $n = 1$  case referred to in Section II A. The objective function that needs to be minimised in order to optimise a network in this theory is the Euclidean distortion  $D$  defined as

$$D \equiv \sum_{y=1}^M \int d\mathbf{x} d\mathbf{x}' \Pr(\mathbf{x}) \Pr(y|\mathbf{x}) \Pr(\mathbf{x}'|y) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (2.2)$$

where  $\mathbf{x}$  is an input vector,  $y$  is a coded version of  $\mathbf{x}$ .  $\mathbf{x}'$  is an approximate version of  $\mathbf{x}$  reconstructed from  $y$ ,  $\Pr(\mathbf{x})$  is the probability density of input vectors,  $\Pr(y|\mathbf{x})$  is a probabilistic encoder and  $\Pr(\mathbf{x}'|y)$  is a probabilistic decoder which is specified by Bayes' theorem as

$$\Pr(\mathbf{x}'|y) = \frac{\Pr(y|\mathbf{x}) \Pr(\mathbf{x})}{\int d\mathbf{x}' \Pr(y|\mathbf{x}') \Pr(\mathbf{x}')} \quad (2.3)$$

$D$  can be rearranged into the form [13]

$$D = 2 \sum_{y=1}^M \int d\mathbf{x} \Pr(\mathbf{x}) \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \quad (2.4)$$

where the reference vectors  $\mathbf{x}'(y)$  are defined as

$$\mathbf{x}'(y) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|y) \mathbf{x} \quad (2.5)$$

Although Equation 2.2 is symmetric with respect to interchanging the encoder and decoder, Equation 2.4 is not. This is because Bayes' theorem has made explicit the dependence of  $\Pr(\mathbf{x}|y)$  on  $\Pr(y|\mathbf{x})$  in Equation 2.4. From a NN viewpoint,  $\Pr(y|\mathbf{x})$  describes the feedforward transformation from the input layer to the output layer, and  $\mathbf{x}'(y)$  describes the feedback transformation that is implied from the output layer to the input layer. The feedback transformation is necessary to implement the objective function that has been chosen here. This use of separate feedforward and feedback transformations is analogous to the recognition and generative models used in [1] and [4]; however, in this paper the recognition (i.e. feedforward) model is the basic building block, whereas in [1] and [4] the generative model is the basic building block. Another difference between these two approaches is that this paper describes an auto-encoding method, whereas [1] and [4] describe a probability density approximation method.

Minimisation of  $D$  with respect to all of its free parameters leads to an optimal encoder/decoder. In Equation 2.4, the  $\Pr(y|\mathbf{x})$  are the only free parameters, because  $\mathbf{x}'(y)$  is determined by equation (5). However, in practice, both  $\Pr(y|\mathbf{x})$  and  $\mathbf{x}'(y)$  may be treated as free parameters [13], because the  $\mathbf{x}'(y)$  satisfy Equation 2.5 at stationary points of  $D$  with respect to variation of  $\mathbf{x}'(y)$ . In the case where  $\Pr(y|\mathbf{x})$  describes winner-take-all

neurons, this reduces to the standard Linde-Buzo-Gray (LBG) VQ optimisation algorithm [8].

An extension of this model to the case where the location of the single firing event is corrupted by a noise process is equivalent to the well-known topographic mapping model [6]. This may also be interpreted as a way of optimizing the transmission of information along a noisy communication channel [2, 7, 16].

### C. Multiple firing event model

The model may be extended to the case where  $n$  output neurons are observed to fire.  $\Pr(y|\mathbf{x})$  is then re-

placed by  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$ , which is the probability that  $(y_1, y_2, \dots, y_n)$  are the first  $n$  neurons to fire (in that order). With this modification,  $D$  becomes (compare Equation 2.4)

$$D \equiv \sum_{y_1, y_2, \dots, y_n=1}^M \int d\mathbf{x} \Pr(\mathbf{x}) \Pr(y_1, y_2, \dots, y_n | \mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y_1, y_2, \dots, y_n)\|^2 \quad (2.6)$$

where the reference vectors  $\mathbf{x}'(y_1, y_2, \dots, y_n)$  are defined as (compare Equation 2.5)

$$\mathbf{x}'(y_1, y_2, \dots, y_n) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|y_1, y_2, \dots, y_n) \mathbf{x} \quad (2.7)$$

The dependence of  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  and  $\mathbf{x}'(y_1, y_2, \dots, y_n)$  on  $n$  output neuron locations complicates this result. Assume that  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  is a symmetric function of its  $(y_1, y_2, \dots, y_n)$  arguments, which corresponds to ignoring the order in which the first  $n$  firing events occur (i.e.  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  is a sum over all permutations of  $(y_1, y_2, \dots, y_n)$ ). Assume that the neurons fire independently so that  $\Pr(y_1, y_2 | \mathbf{x}) = \Pr(y_1 | \mathbf{x}) \Pr(y_2 | \mathbf{x})$ .  $D$  may then be shown to satisfy the inequality  $D \leq D_1 + D_2$  (see Appendix A), where

$$D_1 \equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y | \mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \quad (2.8)$$

$$D_2 \equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y | \mathbf{x}) \mathbf{x}'(y) \right\|^2$$

In this case  $\mathbf{x}'(y)$  is not given by Equation 2.5.  $D_1$  and  $D_2$  are both non-negative.  $D_1 \rightarrow 0$  as  $n \rightarrow \infty$ , and  $D_2 = 0$  when  $n = 1$ , so the  $D_1$  term is the sole contribution when  $n = 1$ , and the  $D_2$  term provides the dominant contribution as  $n \rightarrow \infty$ . The difference between the  $D_1$  and the  $D_2$  terms is the location of the  $\sum_{y=1}^M \Pr(y | \mathbf{x}) (\dots)$  average: in the  $D_2$  term it averages a vector quantity  $\mathbf{x}'(y)$ , whereas in the  $D_1$  term it averages a Euclidean

distance  $\|\mathbf{x} - \mathbf{x}'(y)\|^2$ . The  $D_2$  term will therefore exhibit interference effects, whereas the  $D_1$  term will not. Thus far the only assumptions about the objective function that have been made are as follows:

1. The required number  $n$  of observed neural firing events is specified in advance. This could be relaxed by averaging over different possible values of  $n$ , but this would complicate the interpretation of the results that will be presented in Section III.
2. The order in which the neural firing events occur is not observed, and hence their exact timing is not observed. There is plenty of scope for generalising the model by observing the timing of the firing events, but such extensions are not presented in this paper.
3. The neural firing events occur independently of each other. In Appendix A, an expression for  $D_2$  is derived which does not assume statistically independent firing events.
4. The objective function seeks to minimise an average Euclidean reconstruction error. This choice is made for two main reasons. First, it makes the theoretical manipulations easier to do. Second, it reduces to some standard theories in the  $n = 1$  case, such as VQ theory and topographic mapping networks [13].

Ideally  $D_1 + D_2$  should be minimised with respect to both  $\Pr(y | \mathbf{x})$  and  $\mathbf{x}'(y)$ . However, in this paper  $\Pr(y | \mathbf{x})$  will be restricted to modelling one or more winner-take-all NNs, but  $\mathbf{x}'(y)$  will be fully optimised.

#### D. Stationarity condition

The optimal  $\mathbf{x}'(y)$  must satisfy  $\frac{\partial(D_1+D_2)}{\partial \mathbf{x}'(y)} = 0$ . From Equation 2.8,  $\frac{\partial(D_1+D_2)}{\partial \mathbf{x}'(y)}$  can be written as

$$\frac{\partial(D_1+D_2)}{\partial \mathbf{x}'(y)} = -\frac{4}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \Pr(y|\mathbf{x}) \left( \mathbf{x} - \mathbf{x}'(y) + (n-1) \sum_{y'=1}^M \Pr(y'|\mathbf{x}) (\mathbf{x} - \mathbf{x}'(y')) \right) \quad (2.9)$$

and using Bayes' theorem in the form  $\Pr(\mathbf{x}|y) \Pr(y) = \Pr(y|\mathbf{x}) \Pr(\mathbf{x})$ , the stationarity condition a  $\frac{\partial(D_1+D_2)}{\partial \mathbf{x}'(y)}$  yields a matrix equation for the  $\mathbf{x}'(y)$  (assuming that  $\Pr(y) > 0$ )

$$n \int d\mathbf{x} \Pr(\mathbf{x}|y) \mathbf{x} = (n-1) \sum_{y'=1}^M \left( \int d\mathbf{x} \Pr(\mathbf{x}|y) \Pr(y'|\mathbf{x}) \right) \mathbf{x}'(y') + \mathbf{x}'(y) \quad (2.10)$$

Now write  $D_1 + D_2$  in Equation 2.8 in the form

$$\begin{aligned} D_1 + D_2 &= 2 \int d\mathbf{x} \Pr(\mathbf{x}) \|\mathbf{x}\|^2 - 4 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x} \cdot \mathbf{x}'(y) \\ &\quad + \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x}'(y)\|^2 + \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 \end{aligned} \quad (2.11)$$

and manipulate Equation 2.10 by taking the inner product with  $\mathbf{x}'(y)$ , multiplying by  $\Pr(y)$ , using Bayes' theorem in the form  $\Pr(\mathbf{x}|y) \Pr(y) = \Pr(y|\mathbf{x}) \Pr(\mathbf{x})$ , and summing over  $y$  using  $\sum_{y=1}^M (\dots)$  to obtain

$$n \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x} \cdot \mathbf{x}'(y) = (n-1) \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 + \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x}'(y)\|^2 \quad (2.12)$$

Now combine Equation 2.11 and Equation 2.12 to obtain the result

$$D_1 + D_2 = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \|\mathbf{x}\|^2 - \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x}'(y)\|^2 - \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 \quad (2.13)$$

The results in Equation 2.10 and Equation 2.13 may be used to solve for the optimum reference vectors and to evaluate the corresponding stationary value of  $D_1 + D_2$ .

### III. SOLVABLE ANALYTIC MODEL

The purpose of this Section is to work through a case study in order to demonstrate some interesting properties that emerge when  $D_1 + D_2$  is minimised. In Section III A, specific forms for  $\Pr(y|\mathbf{x})$  and  $\Pr(\mathbf{x})$  are given, in which  $\Pr(y|\mathbf{x})$  models one or more winner-take-all NNs and  $\Pr(\mathbf{x})$  models a training set that lives on the surface of a high-dimensional torus. In Section III C, the optimum  $\mathbf{x}'(y)$  are derived, and in Section III B, the optimum  $\mathbf{x}'(y)$  are substituted back into the expression for  $D_1 + D_2$  to obtain the minimum value of the objective function.

#### A. Simplified $\Pr(y|\mathbf{x})$ and $\Pr(\mathbf{x})$

It is convenient to constrain the probabilistic description of the neuron firing properties  $\Pr(y|\mathbf{x})$ , so that it models a winner-take-all NN

$$\Pr(y|\mathbf{x}) = \delta_{y,y(\mathbf{x})} \quad (3.1)$$

This may readily be generalised to model more than one winner-take-all NN, in which the set of  $M$  neurons is partitioned into  $N$  subsets. The  $p$ th subset will be denoted as  $Y_p$  which contains  $M_p$  neurons ( $\sum_{p=1}^N M_p = M$ ), whose properties are modelled by the following mixture distribution:

$$\Pr(y|\mathbf{x}) = \frac{1}{N} \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)} \quad (3.2)$$

where the input vector has been partitioned as  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ , and the  $p$ th subset of neurons is associated with the  $p$ th subspace of the input via the function  $y_p(\mathbf{x}_p)$ , where  $y_p(\mathbf{x}_p) \in Y_p$  ensures that the neuron located at  $y_p(\mathbf{x}_p)$  is indeed in the  $p$ th subset. Each subset is assumed to have equal weight in the mixture distribution in Equation 3.2.

It is also convenient to assume that there is a hypothetical infinite-sized training set which may be described in a frequentist way by the probability density  $\text{Pr}(\mathbf{x})$ , and to make the following statistical independence assumption:

$$\text{Pr}(\mathbf{x}) = \text{Pr}(\mathbf{x}_1) \text{Pr}(\mathbf{x}_2) \cdots \text{Pr}(\mathbf{x}_N) \quad (3.3)$$

which ensures that the  $N$  subsets of neurons are each associated with a statistically independent subspace of the input vector. Furthermore assume that each of these subspaces can be broken down further as  $\mathbf{x}_p = (\mathbf{x}_{p,1}, \mathbf{x}_{p,2}, \dots, \mathbf{x}_{p,K_p})$  and that  $\text{Pr}(\mathbf{x}_p)$  factorises thus

$$\text{Pr}(\mathbf{x}_p) = \text{Pr}(\mathbf{x}_{p,1}) \text{Pr}(\mathbf{x}_{p,2}) \cdots \text{Pr}(\mathbf{x}_{p,K_p}) \quad (3.4)$$

to model  $K_p$  statistically independent subspaces within  $\mathbf{x}_p$ . Overall, this model allows each subset of neurons to be associated with one or more statistically independent subspaces in the input vector, and prevents different subsets of neurons from having any such subspaces in common.

For simplicity, assume that the  $\text{Pr}(\mathbf{x}_{p,i})$  all have the same properties given by

$$\text{Pr}(\mathbf{x}_{p,i}) = \frac{1}{2\pi} \int_0^{2\pi} d\theta \delta(\mathbf{x}_{p,i} - \mathbf{x}_{p,i}(\theta)) \quad (3.5)$$

i.e.  $\text{Pr}(\mathbf{x}_{p,i})$  describes a probability density of vectors that live on a loop in  $\mathbf{x}_{p,i}$ -space parameterised by a phase angle  $\theta$ . In order to make it easy to deduce the optimum reference vectors, choose  $\mathbf{x}_{p,i}(\theta)$  so that the following two conditions are satisfied for  $i = 1, 2$

$$\begin{aligned} \|\mathbf{x}_{p,i}\|^2 &= \text{constant} \\ \left\| \frac{\partial \mathbf{x}_{p,i}}{\partial \theta} \right\|^2 &= \text{constant} \end{aligned} \quad (3.6)$$

This type of training set can be visualised topologically. Each  $\mathbf{x}_{p,i}$  is parameterised by its own phase angle, and therefore lives in a subspace that has the topology of a circle, which is denoted as  $S^1$ . The Cartesian product of  $K$  circles is a  $K$ -torus, so because of the independence assumptions in Equation 3.3 and Equation 3.4,  $\mathbf{x}$  lives on the surface of a  $K$ -torus, where  $K = \sum_{p=1}^N K_p$ . The minimisation of  $D_1 + D_2$  thus reduces to finding the optimum way of optimising a set of winner-take-all NNs for input vectors that live on a  $K$ -torus, with the proviso that the probability density of input vectors is uniform on the  $K$ -torus (this follows from Equation 3.5 and Equation 3.6). Note that this type of toroidal training set is used only in order to be able to derive explicit analytic results, and an explicit example is shown in Figure 1.

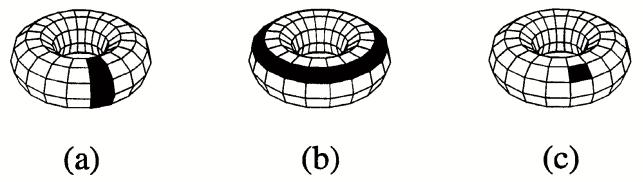


Figure 1: Explicit representation of  $S^1 \times S^1$  topology as a two-torus with the effect of three different types of winner-take-all neuron shown. A typical region of the two-torus that causes a neuron to fire is highlighted in each case.

## B. Optimum $\mathbf{x}'(y)$

In Appendix B, the stationarity condition is solved to obtain the following result for  $y \in Y_q$

$$x_q'(y) = \frac{Nn}{N+n-1} \int d\mathbf{x}_q \text{Pr}(\mathbf{x}_q|y) \mathbf{x}_q \quad (3.7)$$

whence

$$\begin{aligned} \|x_q'(y)\|^2 &= \frac{N^2 n^2}{(N+n-1)^2} \left\| \int d\mathbf{x}_q \text{Pr}(\mathbf{x}_q|y) \mathbf{x}_q \right\|^2 \\ &= \frac{N^2 n^2}{(N+n-1)^2} K_q R(M_q^{\frac{1}{K_q}}) \end{aligned} \quad (3.8)$$

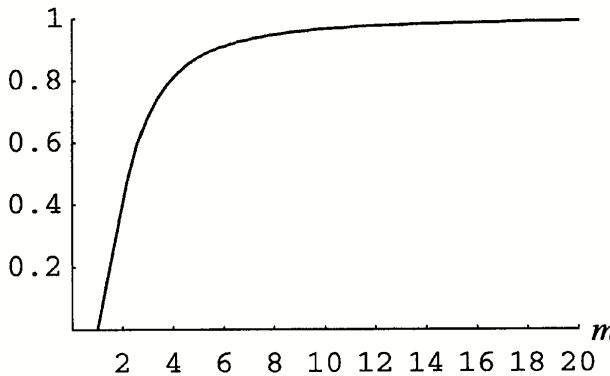
where the function  $R(m)$  is defined below, and the simplification makes use of the fact that  $\mathbf{x}_q$  lies on a  $K_q$ -torus, and that the  $M_q$  neurons partition the  $K_q$ -torus symmetrically into  $M_q$  equivalent pieces. Thus, each of the  $K_q$  circles that defines the  $K_q$ -torus is partitioned into  $M_q^{\frac{1}{K_q}}$  equivalent pieces, so the  $K_q$ -torus (which is the Cartesian product of  $K_q$  circles, whence the overall factor of  $K_q$  which appears in Equation 3.8) requires  $(M_q^{\frac{1}{K_q}})^{K_q} = M_q$  winner-take-all neurons to partition it. The function  $R(m)$  is defined as

$$\begin{aligned} R(m) &\equiv \left( \frac{\int_{-\frac{\pi}{M}}^{+\frac{\pi}{M}} d\theta \cos \theta}{\int_{-\frac{\pi}{M}}^{+\frac{\pi}{M}} d\theta} \right)^2 \\ &= \left( \frac{m}{\pi} \right)^2 \sin^2 \frac{\pi}{M} \end{aligned} \quad (3.9)$$

which is plotted in Figure 2 to show its expected monotonic dependence on  $m$ .

Geometrically,  $R(m)$  is the squared radius of the centroid of a  $\frac{2\pi}{M}$  arc of a unit circle centred on the origin. One limiting case is  $R(m) \rightarrow 0$  as  $m \rightarrow 1$  because the centroid of a circle centred on the origin is at the origin. Another limiting case is  $R(m) \rightarrow 1$  as  $m \rightarrow \infty$  because the centroid of an infinitesimal arc of a unit circle lies at unit distance from the origin.

In order to develop an intuitive feeling for the partitioning of circles by winner-take-all neurons, it is useful to consider a low-dimensional special case. Thus, consider the following case, where  $K = 2$  (i.e. the input

Figure 2: Plot of  $R(m)$ .

vectors live on a two-torus). The various ways in which neurons can partition a 2-torus are shown in Figure 1. In Figure 1(a) the neurons partition one of the  $S^1$  subspaces into a number of pieces, but not the other  $S^1$  subspace. In Figure 1(b) the situation is that the subspaces are interchanged with respect to Figure 1(a). In Figure 1(c) the neurons partition both subspaces. If there is the same number  $M$  of neurons in each case then in Figure 1(a) one subspace is partitioned into  $M$  equivalent pieces and the other into one piece, and in Figure 1(b) the roles of the subspaces are reversed, whereas in Figure 1(c) both subspaces are partitioned into  $\sqrt{M}$  equivalent pieces (to guarantee that the total number of pieces is  $M = (\sqrt{M})^2$ ).

### C. Optimum $D_1 + D_2$

The various terms in Equation 2.13 may now be simplified by using the results in Equation 3.7 and Equation 3.8, as detailed in Appendix C. The results for the three terms are

$$\begin{aligned} \text{term 1} &= 2K \\ \text{term 2} &= -\frac{2Nn}{(N+n-1)^2} \sum_{q=1}^N K_q R(M_q^{\frac{1}{K_q}}) \\ \text{term 3} &= -\frac{2n(n-1)}{(N+n-1)^2} \sum_{q=1}^N K_q R(M_q^{\frac{1}{K_q}}) \end{aligned} \quad (3.10)$$

which may be gathered together to obtain  $D_1 + D_2$  in the form

$$D_1 + D_2 = 2K - \frac{2n}{N+n-1} \sum_{q=1}^N K_q R(M_q^{\frac{1}{K_q}}) \quad (3.11)$$

where the various pieces of notation have the following meaning:  $N$  is the number of subsets into which the  $M$  neurons are split,  $M_q$  is the number of neurons in the  $q$ th subset (so  $\sum_{q=1}^N M_q = M$ ),  $K$  is the total number

of  $S^1$  subspaces in the input vector,  $K_q$  is the number of these subspaces that are associated with the  $q$ th subset (so  $\sum_{q=1}^N K_q = K$ ),  $n$  is the number of neural firing events, and  $R(m)$  is the squared distance from the centre of a unit circle to the centroid of an arc of that circle of angular size  $\frac{2\pi}{M}$ .

## IV. INTERPRETATION OF THE RESULTS

The expression for  $D_1 + D_2$  in Equation 3.11 has many interesting properties. In Section IV A, the special case where the neurons are not partitioned into subsets is considered, and in Section IV B this is generalised to the case where they are partitioned into more than one subset. In Section IV C, the case of a variable number of subsets is briefly discussed, and in Section IV D a simple worked example of the comparison between different types of potentially optimal solution is given.

### A. $N = 1$ Case

Assume that the neurons operate as a single winner-take-all NN so that  $N = 1$ , and assume that these neurons are associated with all of the circular components of the input vector so that  $K_1 = K$ , in which case Equation 3.11 reduces to

$$D_1 + D_2 = 2K \left( 1 - R(M^{\frac{1}{K}}) \right) \quad (4.1)$$

which is independent of  $n$  because after the first neuron has fired there is no more information to be gained by observing more firing events (there is only one winning neuron when  $N = 1$ ). In Equation 4.1,  $D_1 + D_2$  monotonically decreases from  $2K$  to 0 as  $M$  is increased from 1 to  $\infty$  (see Figure 2 for the behaviour of  $R(M)$ ). In the limit  $M = 1$ , no information about the input vector is preserved by observing the single neuron firing, so  $D_1 + D_2$  reduces to  $2K$  which corresponds to a state of complete ignorance about the input vector, as expected. In the limit  $M \rightarrow \infty$  the input can be reconstructed exactly because so much information is available in the firing events that  $D_1 + D_2$  reduces to 0, as expected. In the geometrical picture, the  $M$  neurons partition the input space into  $M$  pieces, and the  $M = 1$  and  $M \rightarrow \infty$  limits behave as expected.

Now assume that  $K_1 = K$  (where  $0 \leq a \leq 1$ ) so that the neurons are associated with only  $aK$  circular components of the input vector, and not associated with the other  $(1-a)K$  circular components. Equation 4.1 then becomes

$$D_1 + D_2 = 2(1-a)K + 2aK \left( 1 - R(M^{\frac{1}{aK}}) \right) \quad (4.2)$$

which is exactly as expected, because the  $2(1-a)K$  is the contribution to  $D_1 + D_2$  from the circular components which are not associated with the neurons (i.e. there is

no information about these components), and  $2aK(1 - R(M^{\frac{1}{aK}}))$  is what Equation 4.1 would have given for  $aK$  circular components in the first place. Apart from the overall factor of  $K$  in Equation 4.2 the expression for  $D_1 + D_2$  is a function of  $a$  and  $b = M^{\frac{1}{K}} > 1$ , thus

$$D_1 + D_2 = 2K \left( 1 - aR(b^{\frac{1}{a}}) \right) \quad (4.3)$$

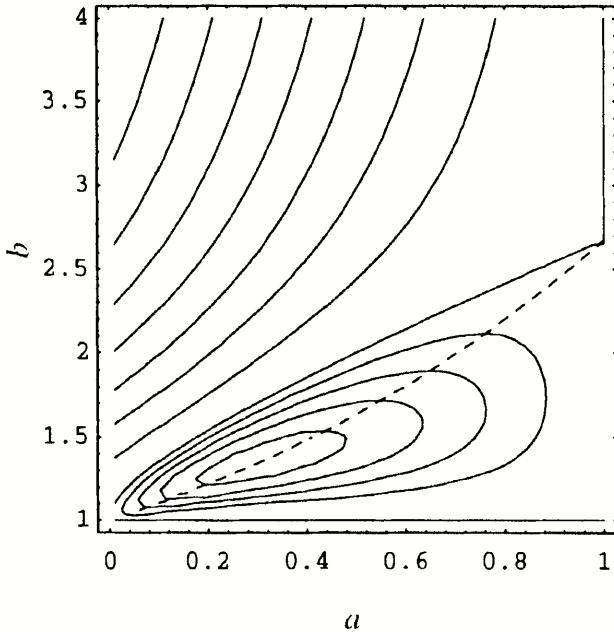


Figure 3: Contour plot of  $R(b) - aR(b^{\frac{1}{a}})$ , i.e.  $D_1 + D_2$  without the overall factor of  $2K$  and with its value at  $a = 1$  subtracted out. This plot may thus be used only when the value of  $b$  is held constant. There are two basic regions: (i) there is a minimum at  $(a, b) = (0.307, 1.352)$  (approximately), and (ii) for  $b > 2.676$  (approximately)  $D_1 + D_2$  increases as  $a$  is decreased from 1 to 0. Around the minimum the contours are spaced at intervals of 0.02, and elsewhere they are spaced at intervals of 0.1. The dashed line traces the locus of the minimum of  $D_1 + D_2$  with respect to  $a$ .

$R(b) - aR(b^{1/a})$  (i.e.  $D_1 + D_2$  in Equation 4.3 without the overall factor of  $2K$ , and with its value at  $a = 1$  subtracted off) is plotted in Figure 3, which shows that for certain values of  $a$  and  $b$  it is possible to achieve a  $D_1 + D_2$  that is smaller than the value attained when  $a = 1$  (for the same value of  $b$ ). This means that it is sometimes advantageous to associate the neurons with fewer than all of the input components (i.e. to set  $a < 1$ ). The contours in Figure 3 reveal that this situation prevails for  $1 < b < 2.676$  (approximately), which occurs for a fixed number of neurons  $M$  if the total number  $K$  of circular components in the input is large enough. Intuitively, if the input dimensionality is too high for a fixed number of neurons to be associated with all of the input components, then the neurons associate themselves with only a subspace of the input.

## B. $N > 1$ Case

When the neurons behave as  $N$  winner-take-all NNs (where  $N > 1$ ),  $D_1 + D_2$  in Equation 3.11 is dependent on  $n$ . This behaviour occurs because there is an advantage in observing more than one firing event when  $N$  neurons can potentially fire for a given input vector. Once each of the  $N$  winner-take-all networks have had their winner fire once, then there is no further information to be gained. However, the probability that each of the  $N$  winners has fired at least once increases monotonically towards 1 as  $n$  increases, so  $D_1 + D_2$  monotonically decreases with  $n$ .

Write Equation 3.11 in an analogous fashion to Equation 4.3 to obtain

$$D_1 + D_2 = 2K \left( 1 - \frac{n}{N+n-1} \sum_{q=1}^N a_q R(b_q^{\frac{1}{a_q}}) \right) \quad (4.4)$$

where  $K_q = a_q K$  (where  $0 \leq a_q \leq 1$ ) and  $b_q = M_q^{\frac{1}{K}} > 1$ . The effect of each of the  $N$  terms in  $\sum_{q=1}^N (\dots)$  can be separately plotted as in Figure 3, but now there are two additional constraints: (i)  $\sum_{q=1}^N a_q \leq 1$  which ensures that  $\sum_{q=1}^N K_q \leq K$  so that the input components are not over-used, and (ii)  $\sum_{q=1}^N (b_q)^K = M$  which ensures that  $\sum_{q=1}^N M_q = M$  so that all of the neurons are used.

For simplicity, take the limit  $n \rightarrow \infty$  in order to guarantee that each of the  $N$  winners has fired at least once, then the  $\frac{n}{N+n-1}$  factor in Equation 4.4 reduces to 1. Now suppose that the neurons are partitioned into  $N$  equal-sized subsets so that all of the  $b_q$  are the same ( $b_q = b = (\frac{M}{N})^{\frac{1}{K}}$ ), then it remains to decide how the input space should be partitioned by selecting appropriate values for the  $a_q$  which satisfy  $\sum_{q=1}^N a_q \leq 1$  (the case of variable  $b_q$  will be considered as part of the case of variable  $N$  in the next Section). By inspecting Figure 3, there are two distinct cases to consider.

First consider the  $b \geq 2.676$  case.  $D_1 + D_2$  monotonically decreases as  $a_q$  is increased, and also the  $a_q$ -dependence of  $D_1 + D_2$  is convex  $\cup$  as shown in Figure 4. Together with the constraint  $\sum_{q=1}^N a_q \leq 1$ , these properties lead to the optimum choice  $a_q = \frac{1}{N}$ . Thus, the optimal network is the one that partitions the input components into  $N$  equal-sized pieces. The convexity property vanishes as  $b \rightarrow \infty$ , and is replaced by a linear dependence of  $D_1 + D_2$  on  $a_q$ , in which case any set of  $a_q$  that satisfy  $\sum_{q=1}^N a_q = 1$  will be an optimal solution. In this degenerate case,  $b$  is large enough that the number of neurons per partition  $b^K$  is sufficiently large to admit many equally good ways of partitioning the input components.

Second consider the  $b < 2.676$  case. In this case, Figure 4 is replaced by Figure 5. The appearance of a minimum at  $a = a(b)$  can lead to a variety of types of optimum network. As  $b$  decreases the minimum moves towards smaller  $a_q$  values, and attains its maximum depth at

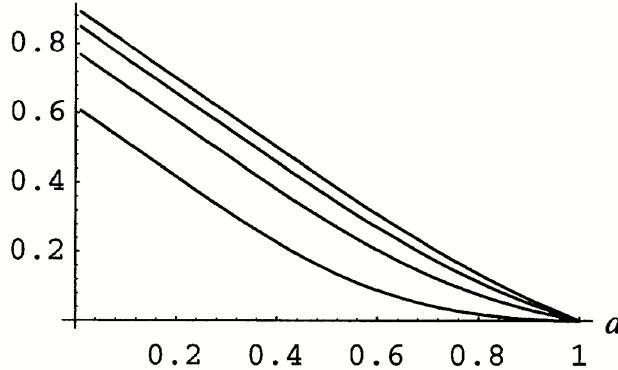
**D1+D2**

Figure 4: Several cuts from left to right through Figure 3 for  $b \geq 2.676$  (the values of  $b$  are separated by 1.0 starting at  $b = 2.676$ ). The behaviour is convex  $\cup$  and gradually degenerates to a linear behaviour as  $b \rightarrow \infty$ .

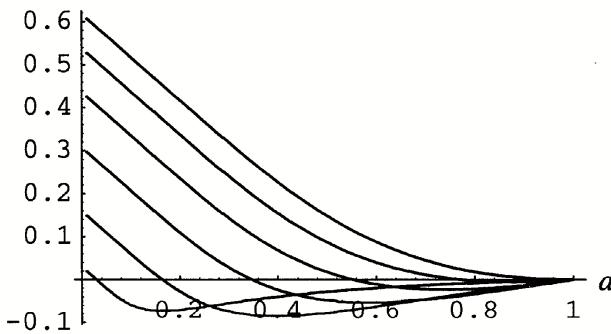
**DI+D2**

Figure 5: Several cuts from left to right through Figure 3 for  $b \leq 2.676$  (the values of  $b$  are separated by  $-0.3$  starting at  $b = 2.676$ ). There is a minimum that progressively moves towards smaller  $a_q$  values as  $b$  decreases.

$(a_q, b) = (0.307, 1.352)$  as noted in Figure 3. The optimum choice is  $a_q = a(b)$  provided that the constraint  $Na(b) \leq 1$  is not violated. Thus, the optimal network is the one that partitions the input components into  $N$  equal-sized pieces, plus a piece that is unused because  $Na(b) < 1$  in general. However, if  $a(b) > \frac{1}{N}$ , then the  $a_q$  cannot all lie at the minimum. In this case the solution reduces to the same  $a_q = \frac{1}{N}$  that was obtained in the  $b \geq 2.676$  case. In summary, for a given  $b$  the optimal choice is  $a_q = \frac{1}{N}$  (all of the input components are associated with neurons), unless  $N$  is small enough that the constraint  $Na(b) \leq 1$  is satisfied by choosing  $a_q = a(b)$ , so that some of the input components are not associated with any neurons.

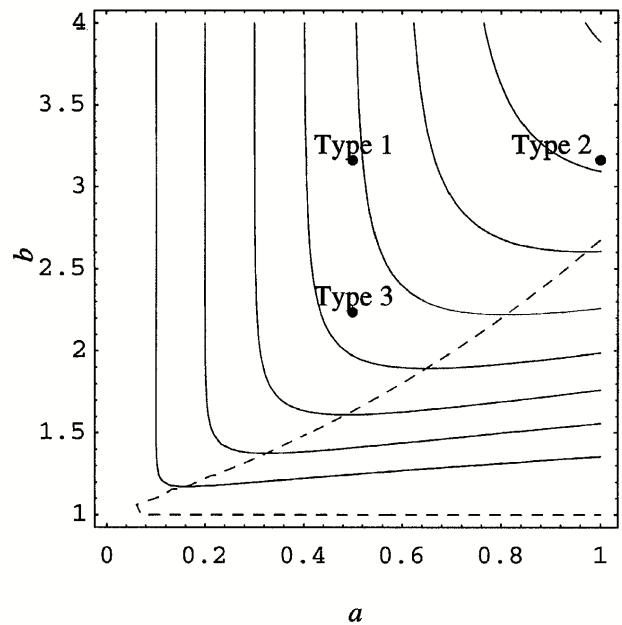
**C. Variable  $N$  Case**

Figure 6: Contour plot of  $-aR(b^{\frac{1}{a}})$ , i.e. the  $a$  and  $b$  dependent part of  $D_1 + D_2$  without the overall factor of  $2K$ . The contours are separated by a height interval 0.1 and  $D_1 + D_2$  decreases towards the top right-hand corner of the plot. The dashed line traces the locus of the minimum of  $D_1 + D_2$  with respect to  $a$ .

The most interesting situation is where the neurons have to decide for themselves how to split up into winner-take-all networks (i.e. what value of  $N$  to choose), because if  $N > 1$  turns out to be optimal then the NN splits into multiple winner-take-all networks by a spontaneous self-organisation process which is driven by the urge to minimise  $D_1 + D_2$ . Figure 6 shows a full contour plot of  $-aR(b^{\frac{1}{a}})$ , which may be used to examine the dependence of a single term in the sum which appears in the general expression for  $D_1 + D_2$  in Equation 4.4. Note that in Figure 6 the value at  $a = 1$  is not subtracted off, unlike in Figure 3; this allows parts of Figure 6 with different values of  $b$  to be compared with each other, as is necessary when  $b$  is being optimised as well as  $a$ . Many different types of optimal solution which minimise  $D_1 + D_2$  in Equation 4.4 can be deduced from this plot. The three special cases plotted in Figure 6 are discussed in the next section.

**D. Low dimensional example**

Equation 3.11 (and, equivalently, Equation 4.4) is a very general result for winner-take-all networks with input vectors that lie on high-dimensional toruses. However, it is useful to consider  $D_1 + D_2$  in a simple low-

dimensional case where  $K = 2$  (i.e. the input vectors have two circular components), and various choices for the other parameters. Three possible choices for the parameter values will be considered:

1. Type 1:  $N = 1, M_1 = M, K_1 = 1$  (i.e.  $a_1 = \frac{1}{2}, b_1 = \sqrt{M}$ ). This corresponds to all of the neurons behaving as in Figure 1(a), or all as in Figure 1(b).
2. Type 2:  $N = 1, M_1 = M, K_1 = 2$  (i.e.  $a_1 = 1, b_1 = \sqrt{M}$ ). This corresponds to all of the neurons behaving as in Figure 1(c).
3. Type 3:  $N = 2, M_1 = M_2 = \frac{M}{2}, K_1 = K_2 = 1$  (i.e.  $a_1 = a_2 = \frac{1}{2}, b_1 = b_2 = \sqrt{\frac{M}{2}}$ ). This corresponds to half of the neurons behaving as in Figure 1(a), and the other half as in Figure 1(b).

$$D_1 + D_2 = 2K \left( 1 - \frac{n}{N+n-1} \sum_{q=1}^N a_q R(b_q^{\frac{1}{a_q}}) \right) \quad (4.5)$$

When these values are substituted into Equation 3.11 (or, equivalently, Equation 4.4) the following results emerge:

$$D_1 + D_2 = \begin{cases} 4 - 2R(M) & \text{type 1} \\ 4 - 4R(\sqrt{M}) & \text{type 2} \\ 4 - \frac{4n}{n+1} R\left(\frac{M}{2}\right) & \text{type 3} \end{cases} \quad (4.6)$$

### **D1+D2**

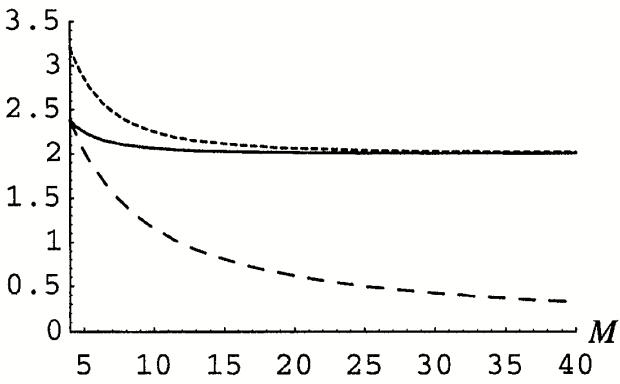


Figure 7: Plots of  $D_1 + D_2$  for  $n = 1$  for each of the three types of optimum. Type 1 is denoted by the solid line, type 2 by the dashed line and type 3 by the dotted line.

In Figure 7, these three expressions for  $D_1 + D_2$  are plotted for the case  $n = 1$  and  $M \geq 4$ . The type 2 solution is always optimal, so it is always favourable to cover the 2-torus as shown in Figure 1(c).

In Figure 8, the three expressions for  $D_1 + D_2$  are plotted for the case  $n = 4$  and  $M \geq 4$ . The type 2 solution

### **D1+D2**

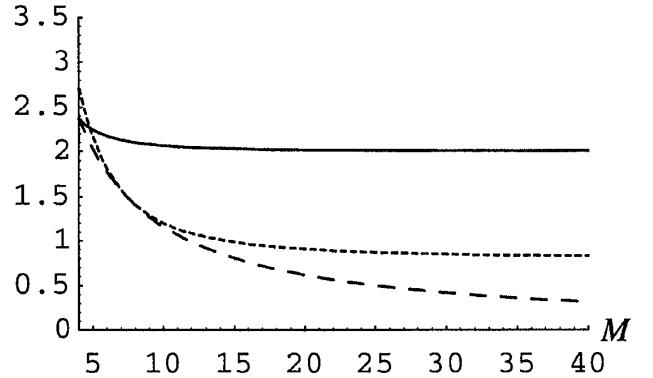


Figure 8: Plots of  $D_1 + D_2$  for  $n = 4$  for each of the three types of optimum.

is optimal over most of the range of values of  $M$ . However, the type 2 and type 3 solutions are equally favoured around  $M = 7$  to  $M = 8$ , where the type 3 solution is the one in which it is favourable to cover the 2-torus by using half the neurons as shown in Figure 1(a) and the other half as shown in Figure 1(b). Compared with the  $n = 1$  results in Figure 7, the type 3 solution has become more favoured because the three additional neural firing events that are allowed by  $n = 4$  make it more likely that both winner-take-all networks acquire useful information about the input vector, and hence help to reduce the value of  $D_1 + D_2$ .

### **D1+D2**

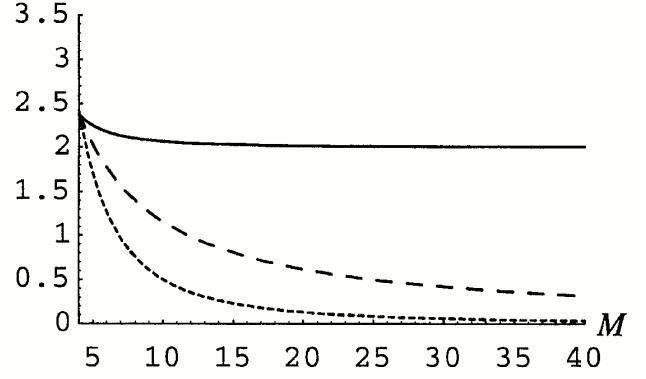


Figure 9: Plots of  $D_1 + D_2$  for  $n \rightarrow \infty$  for each of the three types of optimum.

In Figure 9, the three expressions for  $D_1 + D_2$  are plotted for the case  $n \rightarrow \infty$  and  $M \geq 4$ . In this case the type 3 solution is always optimal, so it is always favourable to cover the 2-torus by using half the neurons as shown in Figure 1(a) and the other half as shown in Figure 1(b).

These results are plotted for the case  $M = 10$  in Figure 6. Note that the contour value for the type 3 result must be multiplied by a factor of two to account for the fact that there are two contributing subsets of neurons (i.e.

$N = 2$  leads to two equal contributions  $-a_q R(b_q^{\frac{1}{q}})$  for  $q = 1, 2$  in Equation 4.4). When this factor of two is accounted for, the type 3 solution has a lower value of  $D_1 + D_2$  than either of the type 1 or type 2 solutions.

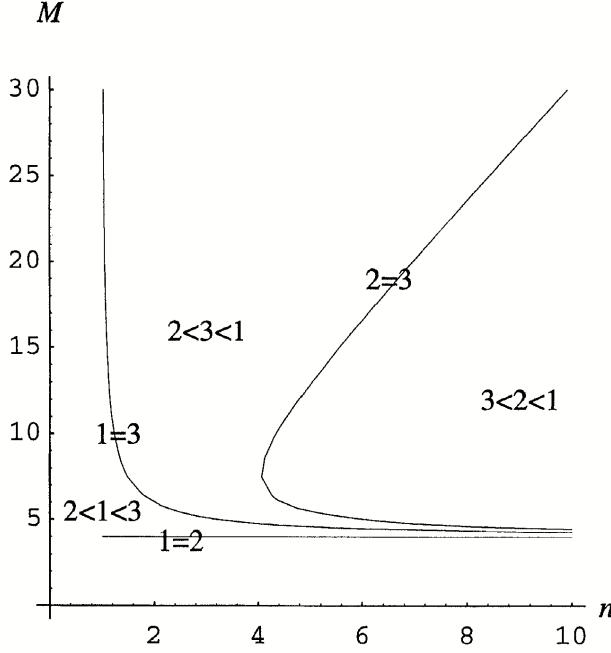


Figure 10: Stability diagram of the regions in which the three types of solution are optimal. The regions and boundaries between regions are labelled to indicate the relative values of  $D_1 + D_2$  for each type of solution.

In Figure 10, a stability diagram is given which shows the relative stability of the three types of solution for different  $M$  and  $n$ . For fixed  $M$  (i.e. fixed network size), as  $n$  is increased the type 3 solution is eventually favoured.

For fixed  $n$  (i.e. fixed observed number of firings), as  $M$  is increased the type 2 solution is eventually favoured. The constraint that holds in practical situations is a fixed network size  $M$ , so the type 3 solution (in which it is favourable to cover the 2-torus by using half the neurons as shown in Figure 1(a) and the other half as shown in Figure 1(b)) is always favoured provided that enough firing events are observed.

## V. CONCLUSIONS

The main result in this paper is the expression for the optimal objective function in Equation 3.11, which is exemplified by the stability diagram in Figure 10. The result in Figure 10 tells us that if a fixed number  $M$  of neurons in a winner-take-all network is trained on input vectors  $\mathbf{x}$  that split into two statistically independent subspaces  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ , then it is always possible to observe a large enough number of neural firing events  $n$  that the optimal network is one in which half the neurons become attached to one subspace and the other half to the other subspace. In effect, the type 3 solution is a factorial (or componential) network which models its input in terms of multiple causes [18].

The type of behaviour in which the neurons in a network partition themselves into two (or more) distinct subsets could serve as a model of the ocular dominance effect observed in the visual cortex (see [3] for a review of this subject), where the left and right eyes correspond to the two subspaces in  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ , and the subsets of neurons with left and right eye dominance could be modelled by the two subsets of neurons in the type 3 solution presented in this paper.

The neurons can be encouraged to develop optimal behaviour that ensures an orderly progression of neural properties across an array of neurons. This requires that cross-talk between the neighbouring neurons be included in the probabilistic firing model. This is the method that was used in [13] to model topographic mapping NNs.

## Appendix A: Upper bound for multiple firing model

It is possible to simplify Equation 2.6 by using the following identity:

$$\mathbf{x} - \mathbf{x}'(y_1, y_2, \dots, y_n) \equiv \frac{1}{n} \sum_{i=1}^n (\mathbf{x} - \mathbf{x}'(y_i)) + \frac{1}{n} \sum_{i=1}^n (\mathbf{x}'(y_i) - \mathbf{x}'(y_1, y_2, \dots, y_n)) \quad (\text{A1})$$

Note that this holds for all choices of  $\mathbf{x}'(y_i)$ . This allows the Euclidean distance to be expanded thus

$$\begin{aligned} \|\mathbf{x} - \mathbf{x}'(y_1, y_2, \dots, y_n)\|^2 &= \frac{1}{n^2} \sum_{i=1}^n \|\mathbf{x} - \mathbf{x}'(y_i)\|^2 + \frac{1}{n^2} \sum_{\substack{i,j=1 \\ i \neq j}}^n (\mathbf{x} - \mathbf{x}'(y_i)) \cdot (\mathbf{x} - \mathbf{x}'(y_j)) \\ &\quad + \frac{2}{n^2} \sum_{i,j=1}^n (\mathbf{x} - \mathbf{x}'(y_i)) \cdot (\mathbf{x}'(y_j) - \mathbf{x}'(y_1, y_2, \dots, y_n)) \\ &\quad + \frac{1}{n^2} \left\| \sum_{i=1}^n (\mathbf{x}'(y_i) - \mathbf{x}'(y_1, y_2, \dots, y_n)) \right\|^2 \end{aligned} \quad (\text{A2})$$

Define  $D_1$ ,  $D_2$  and  $D_3$  as

$$\begin{aligned} D_1 &\equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \\ D_2 &\equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1, y_2=1}^M \Pr(y_1, y_2|\mathbf{x}) (\mathbf{x} - \mathbf{x}'(y_1)) \cdot (\mathbf{x} - \mathbf{x}'(y_2)) \\ D_3 &\equiv 2 \sum_{y_1, y_2, \dots, y_n=1}^M \Pr(y_1, y_2, \dots, y_n) \left\| \mathbf{x}'(y_1, y_2, \dots, y_n) - \frac{1}{n} \sum_{i=1}^n \mathbf{x}'(y_i) \right\|^2 \end{aligned} \quad (\text{A3})$$


---

so that when the four terms in the expansion in Equation A2 are inserted into Equation 2.6 they yield

$$\begin{aligned} \text{term 1} &= D_1 \\ \text{term 2} &= D_2 \\ \text{term 3} &= -2 \times \text{term 4} \\ \text{term 4} &= D_3 \end{aligned} \quad (\text{A4})$$

where  $\Pr(y_1, y_2, \dots, y_n|\mathbf{x})$  has been assumed to be a symmetric function of  $(y_1, y_2, \dots, y_n)$  in the first two results, and the definition of  $\mathbf{x}'(y_1, y_2, \dots, y_n)$  in Equation 2.7 has been used to obtain the third result. These results allow  $D$  in Equation 2.6 to be expanded as  $D = D_1 + D_2 - D_3$ . By noting that  $D_3 \geq 0$ , an up-

per bound for  $D$  in the form  $D \leq D_1 + D_2$  follows immediately from these results. Note that  $D_1 \geq 0$ , whereas  $D_2$  can have either sign. In the special case where  $\Pr(y_1, y_2|\mathbf{x}) = \Pr(y_1|\mathbf{x})\Pr(y_2|\mathbf{x})$  (i.e.  $y_1$  and  $y_2$  are independent of each other given that  $\mathbf{x}$  is known),  $D_2$  reduces to

$$D_2 = \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 \quad (\text{A5})$$

which is manifestly positive. This is the form of  $D_2$  that is used throughout this paper.

## Appendix B: Optimum $\mathbf{x}'(y)$

The stationarity condition in Equation 2.10 is

$$n \int d\mathbf{x} \Pr(\mathbf{x}|y) \mathbf{x} = (n-1) \sum_{y'=1}^M \left( \int d\mathbf{x} \Pr(\mathbf{x}|y) \Pr(y'|\mathbf{x}) \right) \mathbf{x}'(y') + \mathbf{x}'(y) \quad (\text{B1})$$

In order to solve this, it is necessary to separate the  $y' = y$  and  $y' \neq y$  contributions thus

$$n \int d\mathbf{x} \Pr(\mathbf{x}|y) \mathbf{x} = (n-1) \left( \left( \int d\mathbf{x} \Pr(\mathbf{x}|y) \Pr(y|\mathbf{x}) \right) \mathbf{x}'(y) + \sum_{\substack{y'=1 \\ y' \neq y}}^M \left( \int d\mathbf{x} \Pr(\mathbf{x}|y) \Pr(y'|\mathbf{x}) \right) \mathbf{x}'(y') \right) + \mathbf{x}'(y) \quad (\text{B2})$$

Use Bayes' theorem in the form  $\Pr(\mathbf{x}) \Pr(y|\mathbf{x}) = \Pr(\mathbf{x}|y) \Pr(y)$  to simplify the  $y' = y$  contribution to Equation B2

$$\begin{aligned} \int d\mathbf{x} \Pr(\mathbf{x}|y) \Pr(y|\mathbf{x}) &= \frac{\int d\mathbf{x} \Pr(\mathbf{x}) \Pr(y|\mathbf{x})^2}{\int d\mathbf{x} \Pr(\mathbf{x}) \Pr(y|\mathbf{x})} \\ &= \frac{1}{N} \frac{\int d\mathbf{x} \Pr(\mathbf{x}) \left( \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)} \right)^2}{\int d\mathbf{x} \Pr(\mathbf{x}) \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)}} \\ &= \frac{1}{N} \end{aligned} \quad (\text{B3})$$

where, for a given value of  $y$ , only one term in  $\sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)}$  is non-zero because the ranges of the  $y_p(\mathbf{x}_p)$  functions do not overlap, so  $(\sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)})^2 = \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)}$ . Now simplify the  $y' \neq y$  contribution to Equation B2 (using Bayes' theorem as in the  $y' = y$  case)

$$\begin{aligned} \int d\mathbf{x} \Pr(\mathbf{x}|y) \Pr(y'|\mathbf{x}) &= \frac{\int d\mathbf{x} \Pr(\mathbf{x}) \Pr(y|\mathbf{x}) \Pr(y'|\mathbf{x})}{\int d\mathbf{x} \Pr(\mathbf{x}) \Pr(y|\mathbf{x})} \\ &= \frac{1}{N} \frac{\int d\mathbf{x} \Pr(\mathbf{x}) \left( \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)} \right) \left( \sum_{p'=1}^N \delta_{y',y_{p'}(\mathbf{x}_{p'})} \right)}{\int d\mathbf{x} \Pr(\mathbf{x}) \left( \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)} \right)} \end{aligned} \quad (\text{B4})$$

In order for  $\sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)}$  and  $\sum_{p'=1}^N \delta_{y',y_{p'}(\mathbf{x}_{p'})}$  to be simultaneously non-zero, the  $y' \neq y$  constraint must be strengthened to  $y$  and  $y'$  not lying in the same partition, so if  $y \in Y_r$ , then

$$\sum_{\substack{y'=1 \\ y' \neq y}}^M (\dots) \rightarrow \sum_{\substack{q=1 \\ q \neq r}}^N \sum_{y' \in Y_q} (\dots) \quad (\text{B5})$$

whence

$$\begin{aligned} \sum_{\substack{y'=1 \\ y' \neq y}}^M \int d\mathbf{x} \Pr(\mathbf{x}|y) \Pr(y'|\mathbf{x}) \mathbf{x}'(y') &= \frac{1}{N} \sum_{\substack{q=1 \\ q \neq r}}^N \sum_{y' \in Y_q} \frac{\int d\mathbf{x} \Pr(\mathbf{x}) \left( \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)} \right) \left( \sum_{p'=1}^N \delta_{y',y_{p'}(\mathbf{x}_{p'})} \right)}{\int d\mathbf{x} \Pr(\mathbf{x}) \left( \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)} \right)} \mathbf{x}'(y') \\ &= \frac{1}{N} \sum_{\substack{q=1 \\ q \neq r}}^N \sum_{y' \in Y_q} \frac{\int d\mathbf{x} \Pr(\mathbf{x}) \left( \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)} \right) \delta_{y',y_q(\mathbf{x}_q)}}{\int d\mathbf{x} \Pr(\mathbf{x}) \left( \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)} \right)} \mathbf{x}'(y') \\ &= \frac{1}{N} \sum_{\substack{q=1 \\ q \neq r}}^N \frac{1}{M_q} \sum_{y' \in Y_q} \mathbf{x}'(y') \end{aligned} \quad (\text{B6})$$

where the following simplification has been used  $\sum_{y' \in Y_q} \sum_{p'=1}^N \delta_{y',y_{p'}(\mathbf{x}_{p'})} (\dots) = \sum_{y' \in Y_q} \delta_{y',y_q(\mathbf{x}_q)} (\dots)$ , and a  $\delta_{y',y_q(\mathbf{x}_q)}$  factor appearing inside the  $\int d\mathbf{x} \Pr(\mathbf{x}) (\sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)})$  integral reduces it to  $\frac{1}{M_q}$  of the value it would otherwise have had (this follows from the form of the stationary solution which is yet to be derived, and can be checked with hindsight). Finally, combine the results for the  $y' = y$  Equation B3) and  $y' \neq y$  (Equation B6) contributions to obtain the stationary condition in Equation B2 in the form

$$n \int d\mathbf{x}_q \Pr(\mathbf{x}_q|y) \mathbf{x}_q = \frac{N+n-1}{N} \mathbf{x}'_q(y) + \frac{n-1}{N} \sum_{\substack{p=1 \\ p \neq r}}^N \frac{1}{M_p} \sum_{y' \in Y_p} \mathbf{x}'_q(y') \quad (\text{B7})$$

where  $y \in Y_r$  is assumed, and the input vector  $\mathbf{x}$  has been split into its separate components  $\mathbf{x}_q$ .

If  $r \neq q$  (i.e.  $y \notin Y_q$ ), then some further simplifications are possible.  $\int d\mathbf{x}_q \Pr(\mathbf{x}_q|y) \mathbf{x}_q$  must vanish, because  $\Pr(\mathbf{x}_q|y)$  does not then depend on  $\mathbf{x}_q$ , so  $\int d\mathbf{x}_q \Pr(\mathbf{x}_q|y) \mathbf{x}_q \propto \int d\mathbf{x}_q \mathbf{x}_q = \mathbf{0}$  (by symmetry). Equation B7 then yields

$$\frac{N+n-1}{N} \mathbf{x}'_q(y) + \frac{n-1}{N} \sum_{\substack{p=1 \\ p \neq r}}^N \frac{1}{M_p} \sum_{y' \in Y_p} \mathbf{x}'_q(y') = \mathbf{0} \quad (\text{B8})$$

However,  $y$  appears only in the first term, so both terms must separately be zero. The first and second terms then yield (assuming  $N+n \neq 1$  for the first term and  $n \neq 1$  for the second term)

$$\begin{aligned} \mathbf{x}'_q(y) &= \mathbf{0} \quad \text{for } y \notin Y_q \\ \sum_{y' \in Y_q} \mathbf{x}'_q(y') &= \mathbf{0} \end{aligned} \quad (\text{B9})$$

where the first result has been used to simplify the second result. The first of these results means that a subset's reference vector has zero components in all subspaces of the input that are not associated with that subset. The second of these results is guaranteed by symmetry, because the  $\mathbf{x}'_q(y)$  (for  $y \in Y_q$ ) form a regular  $M_p$ -gon centred on the origin in  $\mathbf{x}_q$ -space.

If  $r = q$  (i.e.  $y \in Y_q$ ), then Equation B7 yields

$$\mathbf{x}'_q(y) = \frac{Nn}{N+n-1} \int d\mathbf{x}_q \Pr(\mathbf{x}_q|y) \mathbf{x}_q \quad \text{for } y \in Y_q \quad (\text{B10})$$

### Appendix C: Optimum $D_1 + D_2$

The various terms in Equation 2.13 may be simplified by using the results in Equation 3.7 and Equation 3.8.

The first term is easy to evaluate because it depends only on  $\Pr(\mathbf{x})$

$$\begin{aligned} \text{term 2} &= 2 \int d\mathbf{x} \Pr(\mathbf{x}) \|\mathbf{x}\|^2 \\ &= 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{p=1}^N \sum_{i=1}^{K_p} \|\mathbf{x}_{p,i}\|^2 \\ &= 2K \end{aligned} \quad (\text{C1})$$

where  $\|\mathbf{x}_{p,i}\|^2 = 1$  and  $K = \sum_{p=1}^N K_p$  have been used in the simplification.

The second term may be evaluated as follows:

---


$$\begin{aligned} \text{term 2} &= -\frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x}'(y)\|^2 \\ &= -\frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{p=1}^N \sum_{y \in Y_p} \Pr(y|\mathbf{x}) \sum_{q=1}^N \|\mathbf{x}'_q(y)\|^2 \\ &= -\frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{q=1}^N \sum_{y \in Y_q} \Pr(y|\mathbf{x}) \|\mathbf{x}'_q(y)\|^2 \\ &= -\frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{q=1}^N \sum_{y \in Y_q} \Pr(y|\mathbf{x}) \frac{N^2 n^2}{(N+n-1)^2} K_q R(M_q^{\frac{1}{K_q}}) \\ &= -\frac{2Nn}{(N+n-1)^2} \sum_{q=1}^N K_q R(M_q^{\frac{1}{K_q}}) \end{aligned} \quad (\text{C2})$$

where the results  $\sum_{y=1}^M (\dots) = \sum_{p=1}^N \sum_{y \in Y_p} (\dots)$ ,  $\|\mathbf{x}'(y)\|^2 = \sum_{q=1}^N \|\mathbf{x}'_q(y)\|^2$ ,  $\mathbf{x}'_q(y) = \mathbf{0}$  for  $y \notin Y_q$ ,  $\sum_{y \in Y_q} \Pr(y|\mathbf{x}) = \frac{1}{N}$  and  $\int d\mathbf{x} \Pr(\mathbf{x}) = 1$  have been used in the simplification.

The third term may be evaluated as follows:

$$\begin{aligned}
\text{term 3} &= -\frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 \\
&= -\frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \sum_{p=1}^N \sum_{y \in Y_p} \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 \\
&= -\frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \sum_{p=1}^N \frac{1}{N} \mathbf{x}'(y_p(\mathbf{x}_p)) \right\|^2 \\
&= -\frac{2(n-1)}{nN^2} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{p=1}^N \|\mathbf{x}'(y_p(\mathbf{x}_p))\|^2 \\
&= -\frac{2(n-1)}{nN^2} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{q=1}^N \frac{N^2 n^2}{(N+n-1)^2} K_q R(M_q^{\frac{1}{K_q}}) \\
&= -\frac{2n(n-1)}{(N+n-1)^2} \sum_{q=1}^N K_q R(M_q^{\frac{1}{K_q}})
\end{aligned} \tag{C3}$$

where the results  $\sum_{y=1}^M (\dots) = \sum_{p=1}^N \sum_{y \in Y_p} (\dots)$ ,  $\Pr(y|\mathbf{x}) = \frac{1}{N} \sum_{p=1}^N \delta_{y,y_p(\mathbf{x}_p)}$ ,  $\mathbf{x}'_q(y) = \mathbf{0}$  for  $y \notin Y_q$ , and  $\int d\mathbf{x} \Pr(\mathbf{x}) = 1$  have been used in the simplification.

---

- [1] P Dayan, G E Hinton, R M Neal, and R S Zemel, *The Helmholtz machine*, Neural Computation **7** (1995), no. 5, 889–904.
- [2] N Farvardin, *A study of vector quantisation for noisy channels*, IEEE Transactions on Information Theory **36** (1990), no. 4, 799–809.
- [3] G Goodhill, *Correlations, competition and optimality: modelling the development of topography and ocular dominance*, Technical Report CSRP 226, Sussex University, 1992.
- [4] G E Hinton, P Dayan, B J Frey, and R M Neal, *The ‘wake-sleep’ algorithm for unsupervised neural networks*, Science **268** (1995), no. 5214, 1158–1161.
- [5] G E Hinton and R S Zemel, *Advances in neural information processing systems*, vol. 6, ch. Autoencoders, minimum description length, and Helmholtz free energy, pp. 3–10, Morgan Kaufmann, San Mateo, 1994.
- [6] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [7] H Kumazawa, M Kasahara, and T Namekawa, *A construction of vector quantisers for noisy channels*, Electronics and Engineering in Japan **67** (1984), no. 4, 39–47.
- [8] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [9] S P Luttrell, *Hierarchical self-organising networks*, Proceedings of IEE Conference on Artificial Neural Networks (London), IEE, 1989, pp. 2–6.
- [10] ———, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
- [11] ———, *A trainable texture anomaly detector using the adaptive cluster expansion (ACE) method*, Memorandum 4437, Royal Signals and Radar Establishment, Malvern, 1990.
- [12] ———, *A hierarchical network for clutter and texture modelling*, Proceedings of SPIE conference on adaptive signal processing (San Diego), SPIE, 1991, pp. 518–528.
- [13] ———, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
- [14] ———, *Handbook of neural computation*, ch. Designing analysable networks, pp. B5.3:1–B5.3:8, IOP and OUP, Oxford, 1995.
- [15] ———, *Maximum entropy and Bayesian methods*, ch. The cluster expansion: a hierarchical density model, pp. 269–278, Kluwer, Dordrecht, 1995.
- [16] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
- [17] J Rissanen, *Modelling by shortest data description*, Automatica **14** (1978), no. 5, 465–471.
- [18] E Saund, *A multiple cause mixture model for unsupervised learning*, Neural Computation **7** (1995), no. 1, 51–71.
- [19] R S Zemel and G E Hinton, *Advances in neural information processing systems*, vol. 6, ch. Developing population codes by minimising description length, pp. 11–18, Morgan Kaufman, San Francisco, 1994.

# A Discrete Firing Event Analysis of the Adaptive Cluster Expansion Network \*

S P Luttrell

Defence Research Agency, St Andrews Road, Malvern, Worcs WR14 3PS, UK

This paper describes how a hierarchical network for encoding sensor data (the adaptive cluster expansion network) can be constructed by linking together a number of elementary modules, each of which is a simple two-layer encoder/decoder network. To achieve this goal, a Bayesian analysis is applied to the discrete neural firing events that occur within each layer of the network.

## I. INTRODUCTION

The overall goal of this paper is to relate two pre-existing ideas: (i) the discrete neural firing event analysis of a two-layer self-organising network [12, 14], and (ii) the adaptive cluster expansion (ACE) approach to stepwise encoding of high-dimensional vectors [5, 6].

Figure 1(a) shows the basic structure of a two-layer self-organising network, in which there is both a feedforward recognition model and a feedback generative model (the terminology is borrowed from [1, 2]). The pattern of activity in the input layer is processed by the recognition model to cause neurons in the output layer to fire. For instance, if the recognition model were implemented using weighted connections feeding into sigmoid nonlinearities, then a two-layer perceptron network would result. However, unlike in the perceptron, the pattern of activity in the output layer is then processed by the generative model to try to reconstruct the pattern of activity in the input layer as accurately as possible [11]. The overall goal of optimising this type of network is to adjust the feedforward and feedback processing so that on average the reconstruction error is as small as possible.

The optimisation of such two-layer networks will be described in Section II, and the extension to coupled two-layer networks will be presented in Section III. Finally, a description of how these techniques may be applied to the ACE approach will be given in Section IV.

## II. OPTIMISATION OF A TWO-LAYER NETWORK

In order to make contact with existing results it is necessary to assume that only the location of a single firing event in the output layer, rather than the full pattern of activity, is used by the generative model to reconstruct the pattern of activity in the input layer. Thus define the following notation:  $\mathbf{x}$  = pattern of activity in the

input layer (which may be derived from the output firing events occurring in another network),  $\mathbf{y}$  = location of the firing event in the output layer (which is therefore a lattice vector),  $\Pr(\mathbf{x})$  = probability that input activity  $\mathbf{x}$  occurs,  $\Pr(\mathbf{y}|\mathbf{x})$  = probability that the firing event occurs at location  $\mathbf{y}$  given that the input activity  $\mathbf{x}$  is known (recognition model),  $\Pr(\mathbf{x}|\mathbf{y})$  = probability that the input activity is  $\mathbf{x}$  given that the location  $\mathbf{y}$  of the firing event is known (generative model). Bayes' theorem relates these probabilities thus:

$$\Pr(\mathbf{x}|\mathbf{y}) = \frac{\Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x})}{\int d\mathbf{x}' \Pr(\mathbf{y}|\mathbf{x}') \Pr(\mathbf{x}')} \quad (2.1)$$

A suitable optimisation principle is to minimise the average Euclidean reconstruction error  $D$  defined as [11, 12, 14]

$$D \equiv \int d\mathbf{x} d\mathbf{x}' \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x}'|\mathbf{y}) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (2.2)$$

where the average is performed over all  $\mathbf{x}$ ,  $\mathbf{x}'$  and  $\mathbf{y}$  under the joint probability  $\Pr(\mathbf{x}, \mathbf{x}', \mathbf{y}) = \Pr(\mathbf{x}) \Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x}'|\mathbf{y})$ . The Euclidean distance  $\|\mathbf{x} - \mathbf{x}'\|^2$  can be expanded out as  $\mathbf{x} \cdot \mathbf{x} + \mathbf{x}' \cdot \mathbf{x}' - 2\mathbf{x} \cdot \mathbf{x}'$ , and the symmetry of the expression for  $D$  can be used to integrate out  $\mathbf{x}'$  to yield the result

$$D = \int d\mathbf{x} \Pr(\mathbf{x}) D(\mathbf{x}) \quad (2.3)$$

where  $D(\mathbf{x})$  is defined as  $D(\mathbf{x}) \equiv 2 \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2$ , and  $\mathbf{x}'(\mathbf{y})$  is defined as  $\mathbf{x}'(\mathbf{y}) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ . Conveniently, the stationary point of  $D$  with respect to variation of  $\mathbf{x}'(\mathbf{y})$  is located at  $\mathbf{x}'(\mathbf{y}) = \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ , so the  $\mathbf{x}'(\mathbf{y})$  can be treated as a set of independent parameters whose values must be chosen by minimising  $D$ . In Equation 2.3 the network is described primarily in terms of the recognition model, unlike in [1, 2] where the generative model is primary. If  $\Pr(\mathbf{y}|\mathbf{x})$  is modelled as

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{N(\mathbf{m}(\mathbf{y}), \Sigma(\mathbf{y}))}{\sum_{\mathbf{y}'} N(\mathbf{m}(y'), \Sigma(y'))} \quad (2.4)$$

where  $N(\mathbf{m}, \Sigma)$  is a Gaussian probability distribution in  $\mathbf{x}$ -space with mean  $\mathbf{m}$  and covariance  $\Sigma$ , then minimising  $D$  in Equation 2.3 with respect to the  $\mathbf{m}(\mathbf{y})$  and  $\Sigma(\mathbf{y})$

\*Typeset in LATEX on May 22, 2019.

This paper appeared in Network: Computation in Neural Systems, 1997, vol. 7, pp. 285-290. Received 31 January 1996. This paper was presented at the Workshop on Information Theory and the Brain, held at the University of Stirling, UK, on 4-5 September 1995. © 1996 British Crown Copyright/DRA.

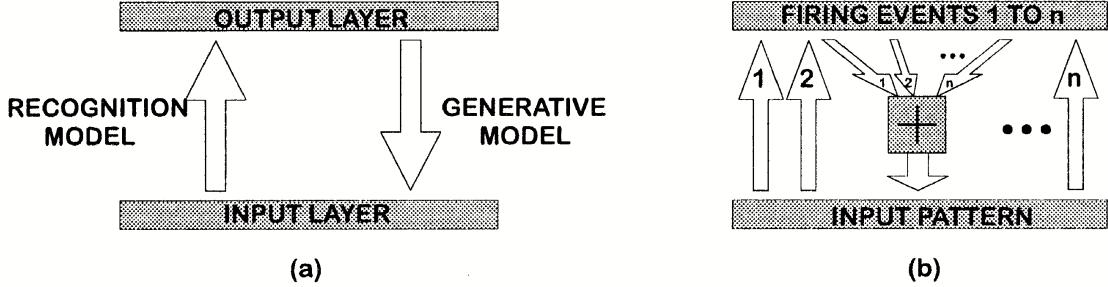


Figure 1: (a) The basic two-layer network with a feedforward recognition model and a feedback generative model. (b) How the recognition model is approximated in practice as a superposition of contributions from the firing events in the output layer.

yields  $D = \int d\mathbf{x} \Pr(\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y}(\mathbf{x}))\|^2$  where  $\mathbf{y}(\mathbf{x}) \equiv \arg \min_{\mathbf{y}} \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2$  [11], which is the standard method of optimising the code vectors  $\mathbf{x}'(\mathbf{y})$  in a vector quantiser using a nearest-neighbour encoding rule  $\mathbf{y}(\mathbf{x})$  [4].

This result can be generalised somewhat by making the replacement

$$\Pr(\mathbf{y}|\mathbf{x}) \longrightarrow \Pr(\mathbf{y}'|\mathbf{x}) = \sum_{\mathbf{y}} \Pr(\mathbf{y}'|\mathbf{y}) \Pr(\mathbf{y}|\mathbf{x}) \quad (2.5)$$

where the  $\Pr(\mathbf{y}|\mathbf{x})$  are now filtered through a matrix of conditional probabilities  $\Pr(\mathbf{y}'|\mathbf{y})$ . The effect of  $\Pr(\mathbf{y}'|\mathbf{y})$  is to cause the neuron firing probabilities to ‘leak’ onto

other neurons, which dilutes the information about precisely which neurons were supposed to fire in the first place. When  $D$  is minimised it tries to limit the damage that is caused by this probability leakage. It achieves this by ensuring that neurons that are connected by the matrix  $\Pr(\mathbf{y}'|\mathbf{y})$  have similar properties, so their parameters become topographically ordered. This type of self-organising map is similar to the well known Kohonen map [3] with neighbourhood function  $\Pr(\mathbf{y}'|\mathbf{y})$  [5, 7].

The result can readily be generalised to  $n$  firing events that are statistically independent given that the input  $\mathbf{x}$  is known [12, 14].  $D(\mathbf{x})$  in Equation 2.3 then becomes

$$D(\mathbf{x}) = 2 \sum_{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n} \Pr(\mathbf{y}_1|\mathbf{x}) \Pr(\mathbf{y}_2|\mathbf{x}) \dots \Pr(\mathbf{y}_n|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)\|^2 \quad (2.6)$$

If  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  is approximated as  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \approx \frac{1}{n} \sum_{i=1}^n \mathbf{x}'(\mathbf{y}_i)$ , as shown in Figure 1(b), then  $D$  has an easily computable upper bound  $D_{ub}$  [12, 14], where  $D \leq D_{ub} = D_1 + D_2$ , with  $D_1$  and  $D_2$  defined as

$$D_1 \equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (2.7)$$

$$D_2 \equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \mathbf{x}'(\mathbf{y}) \right\|^2$$

This result for  $n$  firing events should be compared with the result for one firing event given in Equation 2.3. In effect, Equation 2.3 has split into two terms: the  $D_1$  term is  $\frac{1}{n}$  the standard Euclidean reconstruction error from one firing event, and the  $D_2$  term has the neurons acting cooperatively to form a reconstruction  $\sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \mathbf{x}'(\mathbf{y})$ . When  $n = 1$  the  $D_2$  contribution vanishes, because if there is only one firing event then the neurons cannot act cooperatively. When  $n \rightarrow \infty$  the  $D_1$  contribution van-

ishes, because the reconstruction is dominated by neurons cooperating with each other.

### III. OPTIMISATION OF COUPLED TWO-LAYER NETWORKS

In Figure 2 two two-layer networks are combined to produce a three-layer network. For simplicity, the optimisation principle for multiple two-layer networks will simply be to minimise the sum of their separate Euclidean distortions. The following notation will be used below to describe the operation of the  $l$ th layer in a multi-layer network:  $\mathbf{x}^{(l)}$  = input pattern,  $\mathbf{s}^{(l)}$  = layer parameters,  $(\mathbf{y}_1^{(l)}, \mathbf{y}_2^{(l)}, \dots, \mathbf{y}_{n_l}^{(l)})$  =  $n_l$  output firing events. Thus for

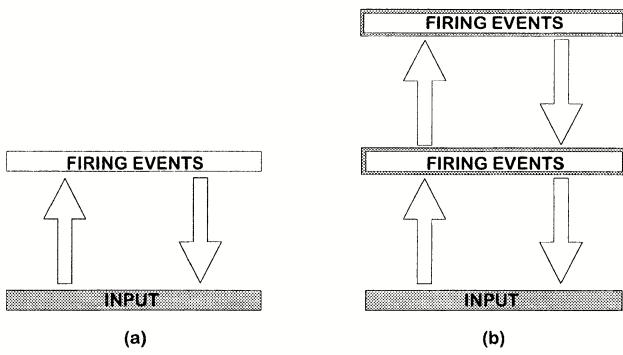


Figure 2: (a) The basic two-layer network. (b) Two two-layer networks coupled together. The input to the upper two-layer network (represented by the shaded rectangle in the middle layer) is derived from the firing events that occur in the output layer of the lower two-layer network (represented by the white rectangle in the middle layer).

an  $(L+1)$ -layer network with layers numbered  $0, 1, \dots, L$

$$\begin{aligned} D &= \sum_{l=1}^L D^{(l)} \\ &\leq \sum_{l=1}^L D_{ub}^{(l)} \\ &= \sum_{l=1}^L \left( D_1^{(l)} + D_2^{(l)} \right) \end{aligned} \quad (3.1)$$

where  $D^{(l)} = \sum_{\mathbf{x}^{(l)}} \Pr(\mathbf{x}^{(l)}) D^{(l)}(\mathbf{x}^{(l)})$  (by analogy with Equation 2.3) is the average Euclidean reconstruction distortion caused by the  $l$ th two-layer network. A summation  $\sum_{\mathbf{x}^{(l)}}$  rather than an integration  $\int d\mathbf{x}^{(l)}$  is used because it is now assumed that the input activity pattern  $\mathbf{x}^{(l)}$  is a vector of  $n_{l-1}$  lattice vectors describing the output firing events in layer  $l-1$

$$\mathbf{x}^{(l)} = (\mathbf{y}_1^{(l-1)}, \mathbf{y}_2^{(l-1)}, \dots, \mathbf{y}_{n_{l-1}}^{(l-1)}) \quad (3.2)$$

The upper bound in Equation 3.1 may be differentiated with respect to the parameters in layer  $l_0 \leq L$  to yield

$$\frac{\partial}{\partial \mathbf{s}^{(l_0)}} \sum_{l=1}^L D_{ub}^{(l)} = \frac{\partial D_{ub}^{(l_0)}}{\partial \mathbf{s}^{(l_0)}} + \sum_{l=l_0+1}^L \sum_{\mathbf{x}^{(l)}} \frac{\partial \Pr(\mathbf{x}^{(l)})}{\partial \mathbf{s}^{(l_0)}} D_{ub}^{(l)}(\mathbf{x}^{(l)}) \quad (3.3)$$

where the  $\sum_{l=l_0+1}^L (\dots)$  term describes the side effects in layers  $l > l_0$  of varying the parameters  $\mathbf{s}^{(l_0)}$  in layer  $l_0$ .

In the case shown in Figure 2(b) this reduces to

$$\begin{aligned} \frac{\partial}{\partial \mathbf{s}^{(1)}} \left( D_{ub}^{(1)} + D_{ub}^{(2)} \right) &= \frac{\partial D_{ub}^{(1)}}{\partial \mathbf{s}^{(1)}} + \sum_{\mathbf{x}^{(2)}} \frac{\partial \Pr(\mathbf{x}^{(2)})}{\partial \mathbf{s}^{(1)}} D_{ub}^{(2)}(\mathbf{x}^{(2)}) \\ \frac{\partial}{\partial \mathbf{s}^{(2)}} \left( D_{ub}^{(1)} + D_{ub}^{(2)} \right) &= \frac{\partial D_{ub}^{(2)}}{\partial \mathbf{s}^{(2)}} \end{aligned} \quad (3.4)$$

Differentiation with respect to the parameters in the upper network gives rise to a straightforward term  $\frac{\partial D_{ub}^{(2)}}{\partial \mathbf{s}^{(2)}}$ , whereas differentiation with respect to the parameters in the lower network gives rise to a sum of two contributions

$$\frac{\partial D_{ub}^{(1)}}{\partial \mathbf{s}^{(1)}} + \sum_{\mathbf{x}^{(2)}} \frac{\partial \Pr(\mathbf{x}^{(2)})}{\partial \mathbf{s}^{(1)}} D_{ub}^{(2)}(\mathbf{x}^{(2)}) \quad (3.5)$$

the first term of which is analogous to that produced by the upper network, and the second term of which describes a back-propagating derivative created by the upper network to inform the lower network about the effect that varying  $\mathbf{s}^{(1)}$  has on its input probability distribution  $\Pr(\mathbf{x}^{(2)})$  [9, 10]; this is self-supervision.

In [10] it was shown how this self-supervision mechanism leads to the automatic generation of neighbourhood functions of the type specified in Equation 2.5. In this model, topographic ordering of the neuron properties occurs as a result of the implicit neighbourhood function generated by the interaction between different layers of neurons, rather than as a result of an explicit neighbourhood function.

#### IV. ADAPTIVE CLUSTER EXPANSION (ACE)

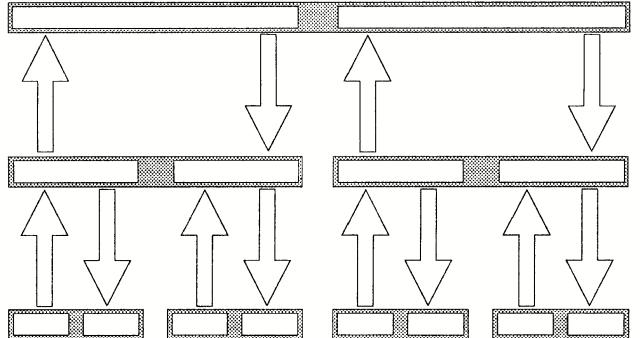


Figure 3: Adaptive cluster expansion (ACE) network formed from two-layer networks of the type shown in Figure 2(a). The input to each two-layer network is derived from the firing events generated by a pair of two-layer networks, as represented by the two white rectangles embedded inside each shaded rectangle.

The three-layer network shown in Figure 2(b) can have its layer-to-layer connections pruned in various ways. A particularly useful scheme is shown in Figure 3, where a number of two-layer networks of the type shown in Figure 2(a) are combined together hierarchically. The coupling between these two-layer networks leads to self-supervision effects. This network is called the adaptive cluster expansion (ACE) because it is an adaptive network that seeks to encode high-dimensional input patterns in a stepwise fashion, starting with low-dimensional pieces

(called clusters) of the input, and progressively (layer-by-layer) combining these to eventually yield a final output that encodes the whole high-dimensional input pattern.

In [6] an application of ACE to image compression is described; it uses only a single firing event in the output of each embedded two-layer network. There are many variations on the connectivity shown in Figure 3. For instance, an ACE network can be used for data fusion applications, where the clusters in the input layer correspond to the separate sensors receiving information, and the output layer (several layers further up the ACE hierarchy) holds a unified representation of the information supplied by the sensors. The output may need to have certain properties that do not emerge spontaneously by self-organisation as the ACE network is trained, in which case an external teacher must add some suitable external-supervision terms to  $D$ , which then modify the self-supervision terms that already appear in Equation 3.3 and Equation 3.4.

The ACE network architecture can be applied to the problem of producing an estimate of the joint PDF of patterns in a high-dimensional input space [8, 13]. If instead of minimising the reconstruction error, the feedforward and feedback processing are optimised specifically so that the best possible PDF estimate (in the relative entropy sense) is produced, then the overall network optimisation criterion is a sum of mutual information within each layer of the network, which is a generalisation of the mutual information maximisation principle used by Becker and Hinton [1].

## V. CONCLUSIONS

In a two-layer network the feedforward and feedback processing between the input and output layers are mutually coupled via Bayes' theorem. This processing can be optimised so as to minimise the average Euclidean reconstruction error that occurs when a forward pass followed by a backward pass is made through the network. If several such two-layer networks are connected together such that the input of each is derived from the output of one or more other two-layer networks, then a multi-layer network of connected two-layer networks can be built, and supervision of the lower networks by the higher networks occurs (i.e. self-supervision).

A particular example of this way of combining two-layer networks is the adaptive cluster expansion (ACE) network, in which each embedded two-layer network produces one or more firing events in its output layer, and two or more such output layers are concatenated to produce the input to another two-layer network; the overall structure is a hierarchically connected set of two-layer networks.

Because its layer-to-layer connectivity is strictly hierarchical, the version of ACE described in this paper is not as flexible as it potentially could be, but it has a large number of simple theoretical properties. It should therefore be used as a theoretical laboratory in which ideas can be tested out, before trying out more realistic connectivities.

- 
- [1] S Becker and G E Hinton, *Self-organising neural network that discovers surfaces in random-dot stereograms*, Nature **355** (1992), no. 6356, 161–163.
  - [2] P Dayan, G E Hinton, R M Neal, and R S Zemel, *The Helmholtz machine*, Neural Computation **7** (1995), no. 5, 889–904.
  - [3] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
  - [4] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
  - [5] S P Luttrell, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
  - [6] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
  - [7] ———, *Derivation of a class of training algorithms*, IEEE Transactions on Neural Networks **1** (1990), no. 2, 229–232.
  - [8] ———, *A hierarchical network for clutter and texture modelling*, Proceedings of SPIE conference on adaptive signal processing (San Diego), SPIE, 1991, pp. 518–528.
  - [9] ———, *Adaptive Bayesian networks*, Proceedings of SPIE conference on adaptive signal processing (Orlando), SPIE, 1992, pp. 140–151.
  - [10] ———, *Self-supervised adaptive networks*, Proceedings of the IEE F **139** (1992), no. 6, 371–377.
  - [11] ———, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
  - [12] ———, *Handbook of neural computation*, ch. Designing analysable networks, pp. B5.3:1–B5.3:8, IOP and OUP, Oxford, 1995.
  - [13] ———, *Maximum entropy and Bayesian methods*, ch. The cluster expansion: a hierarchical density model, pp. 269–278, Kluwer, Dordrecht, 1995.
  - [14] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.

# Conference Papers and Book Chapters



# PRIOR KNOWLEDGE IN SYNTHETIC APERTURE RADAR (SAR) PROCESSING \*

S P Luttrell and C J Oliver

## I. INTRODUCTION

All that we know of the world about us has been deduced from data collected by our senses (eyes, ears, etc), and those of our ancestors. Usually we employ sophisticated transducers (microscopes, telescopes, radars, etc) to transform signals into a form which our senses can "see", therefore we require an accurate model of the transducer(s) if we are to know how our sensory data is acquired. Furthermore we require a model of the source of the signals being transduced in order to "make sense of" this data. We shall call this model "prior knowledge", because it exists in advance of acquiring the data. Such prior knowledge manifests itself with various degrees of complication. In some cases the model may leave very little undetermined, and it remains to determine the values of a finite number of parameters from the data. This type of situation arises when we take measurements of a well known and controlled phenomenon. In other cases the model may provide only a weak constraint on the source of the signal. This may be for two basic reasons: we are ignorant of what we are observing, or we have a detailed model but it contains a large number of parameters. It could be argued that these are the same reason! In all cases however the model is essential if the data is to be interpreted at all.

## II. INFORMATION THEORY AND PRIOR KNOWLEDGE

The basic theory of information was formulated by Shannon. His interest lay in the transmission of information over noisy communication channels. Fortunately the theory is expressed in very general terms, and so we may use it to analyse the information which is acquired by sensors.

In the previous section we pointed out that data may be interpreted only in the light of a model. In other words the information content of sensor data is measured with respect to a model, and so it is not an absolute quantity! The only truly meaningful information measure is thus "relative information". In this spirit it is meaningless to speak of the information content of data ALONE; it is

always necessary to refer to the model being used to interpret the data. Our intuitive notion that an image (a radar picture, say) alone conveys information to us, holds true only because we have a prior notion of what we expect to give rise to such an image - there is an implicit model.

This view of models and data interpretation may seem unduly philosophical! However if we wish to perform a disciplined analysis of what information is, where it comes from, and where it goes to, it is absolutely necessary to identify ALL the information sources (and sinks!): prior knowledge is the most important implicit information source.

## III. DATA ACQUISITION AND PRIOR KNOWLEDGE

A striking way in which prior knowledge may be used is to optimise the way in which data is acquired. In order to understand how this is achieved, the meaning of data redundancy must be understood. Data is redundant if the signal is oversampled: there will not be much potential signal information present (which is undesirable), but what information is present is robust with respect to the effects of data noise (which is desirable). As the redundancy is decreased the amount of potential signal information is increased at the expense of reducing the data's robustness to noise. Clearly we cannot maximise potential signal information AND robustness with respect to noise simultaneously - a tradeoff must be sought. Information theory provides a framework for solving this problem. Thus the net level of signal information in the data may be calculated as the potential information content LESS the amount of information lost in data noise; both of these quantities have rigorous definitions within the framework of information theory. However the POTENTIAL information content is the signal information measured relative to the prior knowledge of the signal source (the model), and so the NET information also depends on the prior knowledge. Thus in seeking the optimum way in which to acquire data we must bear in mind that the optimisation is prior knowledge dependent. We must therefore "match" our transducer/sensor system to the available prior knowledge. This statement is intuitively obvious, but its expression in information theoretic terms is subtle.

The simplest type of data acquisition schemes are well represented as point samples of a continuous "field". Optimisation of the net information (for a given prior knowledge) is then achieved by finding those positions at which it is best to place the samples. In practice there is a cost

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 22, 2019.

This paper appeared in RSRE Research Review, 1985, pp. 73-77.  
Copyright © Controller, HMSO, London, 1985.

S P Luttrell's contribution to this paper was the work on super-resolution.

associated with each sample, and so we should place constraints on the sampling possibilities which are to be considered in the optimisation. The great advantage of this information theoretic approach is that it rigorously combines prior knowledge and data into a composite pool of information. Thus all sources of information are treated on the same footing as required in a unified approach to the problem of sensory data acquisition and interpretation.

#### **IV. APPLICATION TO SYNTHETIC-APERTURE RADAR**

Let us next discuss how the concepts of prior knowledge may be applied to the interpretation of synthetic-aperture radar (SAR) surveillance imagery. A SAR is a coherent radar which synthesises a large effective aperture by combining the signals which are received at well separated points. The large synthetic aperture gives a radar image with very high resolution when compared with conventional radar imagery. For simplicity we shall assume that such surveillance data is acquired by a conventional sampling scheme (eg Nyquist sampling). These images comprise an enormous quantity of data, e.g. 300,000 samples per second, of which the principal component is map-like terrain imagery, which we shall call CLUTTER, within which may be set a few significant objects, termed TARGETS. The clutter requires interpretation in the form of segmentation and classification into a comparatively restricted set of features, e.g. woodland, fields, rivers, roads, built-up areas, etc. Targets within such a scene would then be viewed against a background represented by the different categories of the model.

The observed SAR image could arise from an enormous number of different scatterer configurations all of which would be indistinguishable. If we introduce additional information in the form of prior knowledge about the scatterer properness then the number of feasible scatterer configurations is reduced to a manageable size. It is necessary to devise a suitable means of representing this prior information. One approach is to formulate models which represent the scatterer configuration and then interpret the image in terms of the parameters of these models. The models contain any prior knowledge we possess about the scattering process and the microscopic or macroscopic structure of the objects which can aid our interpretation of clutter and targets within SAR images.

#### **V. CLUTTER MODELS FROM PRIOR KNOWLEDGE**

A striking feature of SAR images is speckle which occurs for the same reasons as laser speckle. Speckle arises because coherent illumination preserves phase detail in the scattered field, which in turn permits con-

structive and destructive interference to occur between the various scattered waves. Such interference gives rise to images with a characteristic granular, or speckled, appearance. A suitable model for speckle production is one in which a large number of scatterers are randomly positioned within each resolution cell. For a constant scatterer cross-section the received field has complex Gaussian statistics with Rayleigh-distributed envelope and negative-exponential intensity (the so-called Gaussian speckle). The image of uniform grass typically demonstrates this type of statistics. More generally we may use a model in which the underlying cross-section fluctuates from resolution cell to resolution cell. The total field would then result from the convolution of the underlying fluctuation with the Gaussian speckle. The underlying fluctuations would then be obtained by deconvolving the speckle. Unfortunately the number of feasible solutions is very large, and so despeckling algorithms cannot provide unique solutions. This is because the effective, signal to noise ratio in a speckled image is of order of unity. We shall discuss later (super-resolution) how small regions of the image "speckle" contain valuable information on targets.

In order to progress beyond the simple scattering model above we need to introduce knowledge about the terrain structures themselves. Indeed, it is the correlations within the image, containing the structure of the scene, which convey the most information about the terrain. Our brains are capable of using this information for segmentation and classification. In principle, measurement of all the correlations between pixel intensities could be used in automatic classification. However the computational task for such a multi-dimensional approach is not feasible, so we are forced to consider simpler models which use only, for example, the spatial autocorrelation function and the single-point moments to represent the different classes of terrain.

The observed structures in the image may be either microscopic or macroscopic in scale. For example, microscopic terrain statistics could be used to represent the structural properties of a single class of terrain. In a wooded area one might represent the local structure in terms of the autocorrelation function of the image which would in turn be related to the underlying spatial scales and fluctuations. The spatial scale would be related to quantities such as tree size and separation, whereas the depth of fluctuation would depend on the extent of shadowing between trees and their variability of cross-section. In principle, the exact positions of individual trees and their sizes could be used in such a model but it seems more sensible to treat the cross-section within the feature as a random statistical variable with appropriate statistics and correlation properties. Thus a model which allows us to include arbitrary spatial spectra and a parameter which describes the spikiness of the fluctuations is called for. one such model is that in which the surface cross-section is gamma-distributed, arising from regarding the local scatterer density as being the result

of birth-death-migration process. In addition to defining the single-point statistics, correlations within the surface cross-section are also introduced. Following coherent imaging the resultant clutter then consists of correlated K-distributions.

The microscopic structure model treats the surface cross-section as a statistical random variable. It makes no use, therefore, of deterministic information on the positions of the boundaries of the different features which make up the image. In many cases such information is readily available from maps or other types of image. The microscopic model implies that the total cross-section is made up of the incoherent summation of individual cross-section contributions. Each feature may thus be treated separately. The total image intensity may be calculated by simply summing the intensities generated by each type of terrain within the resolution cell when imaged with the appropriate imaging response function. Therefore combining these three types of prior knowledge enables us to predict expected clutter properties for arbitrary distributions of terrain features and system resolution.

## VI. SUPER-RESOLUTION FROM PRIOR KNOWLEDGE

Having separated the SAR into target and clutter components by using a suitable threshold derived from the clutter model explained in the previous section, we may introduce further target-specific prior knowledge in order to analyse the target data. An effective way to proceed is given by SUPER-RESOLUTION theory. This technique is used to process the target data so that detail which was not originally apparent becomes clear. This is not a trivial filtering of the data to remove the effects of noise, rather the technique goes further and performs an inversion (or deconvolution) of the data acquisition process to provide us with an estimate of the distribution of scatterers within the target. The extent to which this inversion can be achieved depends on the "strength" of the target prior knowledge. In particular a useful degree of inversion is achieved if this prior knowledge states that most of the received signal derives from a source signal that has a restricted spatial (and/or temporal) extent, which is usually the case for targets. The use of such prior knowledge is intuitively clear since we can then discount the possibility that the inverse contains any pieces which lie outside the a priori admissible region; the inversion process may then concentrate on the admissible region ALONE. In practice the source signal is never completely constrained to lie in such a restricted region because the target data will contain contributions from the surrounding clutter, and so it is false to assert that all the energy comes from the target alone. However we have generalised the basic super-resolution technique to allow for this effect. Furthermore we have formulated a more general version of the technique which permits prior knowledge of internal target structure to be systematically incorporated into

the superresolution algorithm. In practice this technique may be applied with increased effect as the signal to clutter ratio of the SAR data rises beyond the minimum level required for target detection. The output of the super-resolution algorithm can be used as the input to a more general target recognition algorithm, which in turn could feed its conclusions back to control the super-resolution algorithm as necessary. Thus our technique should be viewed as one step in an integrated approach to data interpretation.

The super-resolution algorithm thus performs a simultaneous noise filtering and inversion of SAR target data. It exhibits the desirable property of "graceful degradation"; weakening the prior knowledge merely reduces the extent to which the inversion is taken. In the trivial case when no prior knowledge is supplied the output is merely an interpolation of the data. More generally the output is the same as would be produced by a suitable Wiener filtering of the data. The particular choice of Wiener filter is subtle, and it is made according to what prior knowledge, data acquisition system and data noise are present. Such a filter uses the detailed form of the "speckle" in the vicinity of a target to deduce the target structure. Some typical super-resolution results are shown in the figures. A composite object made up of four components, each with the same cross section but random phase, is combined with a clutter background in Figure 1. The corresponding data set (image) is shown in Figure 2; note that the target detail has been blurred into a single broad peak. Figure 3 shows the output of the super-resolution algorithm without feedback control, and Figure 4 shows the output after a small amount of feedback control has been introduced. Both of these outputs exhibit an enhancement of resolution when compared with the original data set in Figure 2, and feedback control is seen to provide the better output as expected. The types of feedback which we can introduce are varied and depend in detail on the particular application, so Figure 4 should not be taken as an exhaustive example of the types of result which we can obtain.

## VII. CONCLUSIONS

We have shown how both data acquisition and data interpretation must be conducted in the context of whatever prior knowledge is at hand. Thus the choice of how we should "deploy" a set of sensors in order to acquire the maximum amount of useful information must depend on what we already know. Information theory provides a means of making this choice in a systematic fashion. For an image which has been acquired by a SAR sensor the problem of interpretation is subtle. This can only be conducted if a model (prior knowledge) of what caused the image is available. A model with correlated gamma-distributed cross section fluctuations fits the clutter which is observed in SAR data. Such a model can be augmented by the inclusion of collateral map data

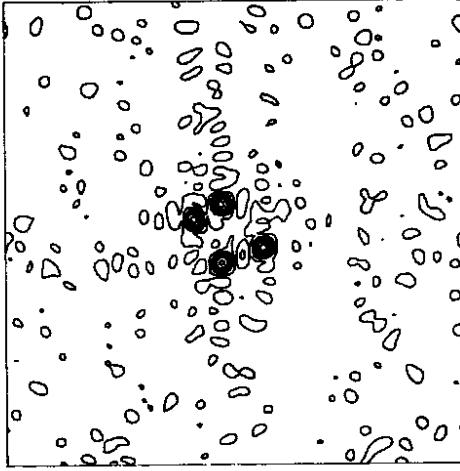


Figure 1: Composite object in clutter.

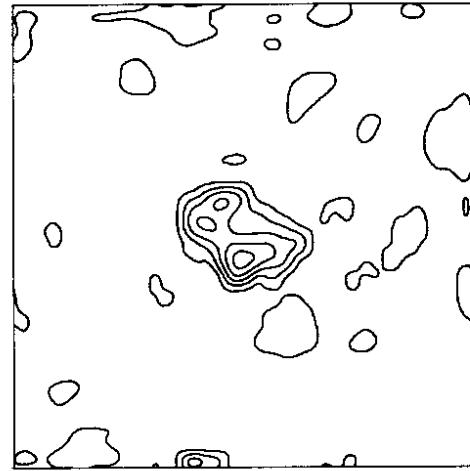


Figure 3: Super-resolved data (no feedback control).

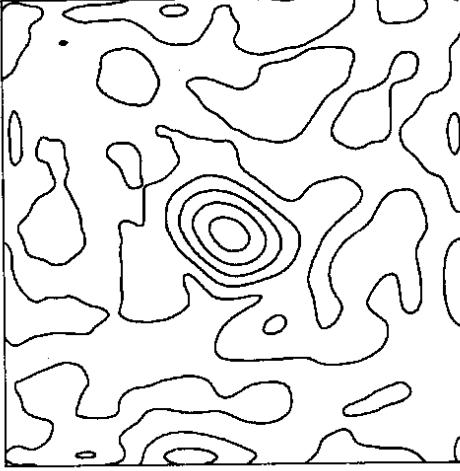


Figure 2: Data set.

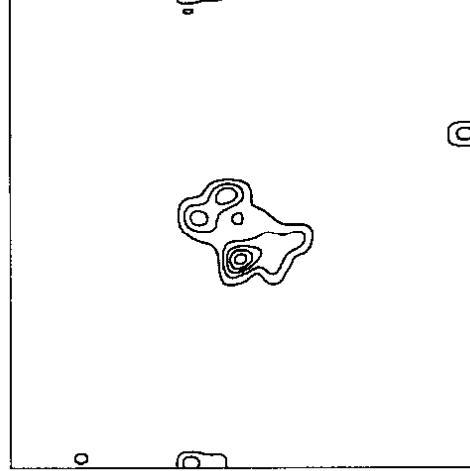


Figure 4: Super-resolved data (no feedback control).

of the scene which the SAR is surveying. Thus the map may be used to determine the clutter model parameters in advance of receiving the SAR image data. Regions of the SAR image which are inconsistent with this clutter model (typically small bright regions) are called targets; a different model is needed for these regions. A general method of interpreting such target data is provided by super-resolution. Thus prior knowledge of the general properties of targets is introduced in order to extract the maximum amount of target structural detail from the image. The effect of such processing is to enhance the resolution of the target image.

All our SAR data processing operations are conducted within an information theoretic framework. This is crucial because there are many ways of processing data to produce APPARENT detail out of nothing! Thus we explicitly identify all prior knowledge sources (ie models) and their implications for image processing. Such

a unified approach to image processing is necessary for systematic information extraction.

## VIII. THE AUTHORS

Dr Luttrell joined RSRE in 1982 on completing his PhD work in the field of quantum chromodynamics. Since then his role has been to conduct basic theoretical research into various aspects of data acquisition and interpretation. His fields of interest include information theory, sampling, physical model building, resolution enhancement, and learning.

Dr Oliver joined RSRE in 1967 to conduct research on photon statistics. He shared the MacRobert Award for this work in 1977. Since then he has investigated aspects of image coding and coded laser rangefinding be-

fore becoming involved in the interpretation of coherent synthetic-aperture radar images. Here his particular in-

terests are in clutter modelling and resolution enhancement.



# The Use of Markov Random Field Models in Sampling Scheme Design \*

S. P. Luttrell

Royal Signals and Radar Establishment, St Andrews Road, Malvern, Worcestershire, WR14 3PS, UK

I use information theoretic techniques to derive schemes for the Bayesian analysis of images with spatially homogeneous statistical properties. In any particular case the scheme is equivalent to deducing the structure of the Markov random field which models the data. This scheme may also be viewed as a generalised sampling technique where the data is reduced by a set of sampling functions to a more compact set of data, which nevertheless retains all the information content of the original data.

## I. INTRODUCTION

A problem in data analysis is to find a suitable means of data processing for solving the inverse problem of drawing inferences from the data. There are many approaches to this problem, each of which has its own advantages and disadvantages [2, 3]. Because there is a variety of methods it is not always clear what the common underlying principles are, so I shall derive ab initio general constraints which all data processing schemes ideally should satisfy.

I take as axiomatic the Bayesian approach to inference [7] which I briefly review in Section II. Thus I assume that all the information which is available about an experimental set-up can be encoded in the form of probability density functions (PDF), and that all inferences should be based upon the form of an *a posteriori* PDF.

A corollary of the Bayesian approach is that the first stage of data processing, namely the data collection process itself, influences the ability to solve the inverse problem. This statement says merely that data which is pertinent to solving the inverse problem must be collected! There is an optimisation problem to be solved in order to discover the best sampling scheme to use in data space, and the optimum solution will be dependent on what information is required from the data [11, 12]; this is explained in Section III.

Once the sampling scheme has been determined the next stage of data processing, namely solving the inverse problem, must have a structure which reflects the type of information which must be recovered from the data. This point is obvious, but it is not at all obvious how to construct an optimal data processing algorithm. I regard the construction of such an algorithm as analogous to the selection of the data sampling scheme: both lead to forms of information processing.

In Section IV I give a brief introduction to clutter models with the purpose of posing an inverse problem in the analysis of images of clutter with spatially homogeneous statistical properties. The particular inverse

problem which I shall consider is the identification of a particular clutter type from amongst a class of types, and I devote Section V to a discussion of the principles which underlie the reduction of clutter data.

Throughout the design of the data processing scheme, which I present in detail in Section VI, I shall be driven by the single requirement that at all stages of processing I must preserve as much information as possible which is pertinent to solving the inverse problem. In Section VII I describe how to recover a Markov random field (MRF) model from the processing scheme, and I show how to deal with multiple clutter types. Finally in Section VIII I describe some applications of my results to practical image processing problems.

## II. BAYESIAN ANALYSIS

I shall be concerned with constructing a particular framework within which inverse problems may be solved by using Bayesian and information theoretic techniques [7]. Throughout this paper I shall denote the state of the system under study as  $f$ , the state of the data as  $g$ , the transformation from  $f$  to  $g$  as  $T[f]$  which may be nonlinear and is always noisy. I shall call the triplet  $\{f, g, T[f]\}$  an experiment, so that

$$g = T(f) \quad (2.1)$$

defines the interrelationship amongst the state variables during a single experiment.

The basic tenet of Bayesian analysis is that information about a system can be encoded in a probability density function. There are two fundamental PDFs associated with a particular experiment: these are the *a priori* PDF  $P[f]$ , and a conditional PDF  $P[g|f]$  which describes the properties of  $T[f]$ . Bayes' theorem

$$P[f|g] \propto P[f] P[g|f] \quad (2.2)$$

is then used to construct the *a posteriori* PDF  $P[f|g]$ . I shall call a reconstruction of  $f$  which is based upon  $P[f|g]$  alone a Bayesian reconstruction.

There are major difficulties in the practical application of Bayesian analysis because it is sometimes difficult to prescribe  $P[f]$  explicitly, and even then it is usually impossible to use Bayes' theorem to obtain  $P[f|g]$  in a

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This appeared in Proceedings of the SPIE Conference on Inverse Problems in Optics, The Hague, 182-188, 1987.

simple form when  $T[f]$  produces correlations amongst the components of  $g$ . These difficulties negate the usefulness of conventional Bayesian analysis of complicated experiments.

The underlying cause of these difficulties is the explicit analytic approach which is being used: this leads to unwieldy formulae for the PDFs in all but the simplest cases. In practice a strictly Bayesian analysis is usually sacrificed in the interests of mathematical simplicity, and instead an *ad hoc* reconstruction method is substituted in order to solve the inverse problem. However I shall retain the Bayesian approach by using Monte Carlo techniques to construct the PDFs.

### III. SAMPLE POSITION OPTIMISATION

In order to put my method into perspective I shall give a summary of some work which I have done on optimisation of the positions of sample points [11, 12]. A common problem which faces an experimenter is to choose in some optimal fashion where to position the sample points in data space. This optimisation problem usually has many conflicting constraints, but an important overall aim is to retain as much useful information as possible in the sample values.

Let  $g$  be the (hypothetical) data that would be obtained if there were sample points at every possible position in data space, and  $g'$  be the data corresponding to a particular sampling scheme chosen by the experimenter. The useful information content of  $g'$  can be gauged by comparing the *a posteriori* PDFs  $P[f|g]$  and  $P[f|g']$ :  $P[f|g]$  represents the ideal to which all  $P[f|g']$  must aspire. Only if  $P[f|g] = P[f|g']$  is all information preserved by the sampling scheme, and invariably this condition is not met in practice. The sample position optimisation problem is complicated by the fact that different  $g$  require different sampling schemes to preserve their information content and to counteract the effect of noise, and so a compromise must be sought when selecting a particular scheme to use for all  $g$ .

The objective function which I maximise is mutual information (or transinformation) which measures the useful information content of  $g'$  in the Shannon sense [17, 18]. Mutual information  $I[f, g]$  is defined as

$$\begin{aligned} I[f, g] &\equiv \int [df] [dg] P[f, g] \log \left( \frac{P[f|g]}{P[f]} \right) \\ &\geq 0 \end{aligned} \quad (3.1)$$

which for an ensemble of experiments measures the average of (the logarithm of) the change in the number of Bayesian reconstructions of  $f$  which may be made before and after the data  $g$  are collected:  $I[f, g]$  is non-negative because  $g$  can only *assist* the Bayesian reconstruction process. The difference between  $I[f, g]$  for the (hypothetical) data  $g$  and the data  $g'$  which are actually used

is given by

$$\begin{aligned} \Delta I(g, g') &\equiv I[f, g'] - I[f, g] \\ &\leq 0 \end{aligned} \quad (3.2)$$

This quantity is never positive because  $g'$  can never contain more information than  $g$ , and it is zero only when  $P[f|g] = P[f|g']$ . I have obtained simple results for  $I[f, g]$  in the special case where both  $P[f]$  and  $P[g|f]$  are Gaussian PDFs and  $T[f]$  is linear [17, 18]. A satisfying corollary of these results is that Shannon sampling of bandlimited functions maximises  $I[f, g]$ .

Such simple results are not obtainable when using  $I[f, g]$  as the objective function if the PDFs are not Gaussian or  $T[f]$  is not linear. Indeed in the general case there are *ad hoc* constraints which are difficult to encode into the PDFs, so there is no easy information theoretic solution. However the general principle that ideally  $P[f|g'] = P[f|g]$  should be used for heuristic guidance on how to select good sampling schemes in complicated situations.

### IV. CLUTTER MODELS

An interesting class of problems for which a systematic approach to the ideal  $P[f|g'] = P[f|g]$  may be attempted is found in the analysis of clutter. Clutter is a generic term which describes that part of an image which is of no interest to the viewer; it literally clutters the field of view. Clutter is usually intrinsically stochastic in nature, and furthermore if the imaging system is coherent then image speckle (which is a form of clutter) compounds its own stochastic properties with those of clutter [14]. The properties of an image  $g$  of a particular type of clutter  $f$  are therefore best described probabilistically by a PDF  $P[g|f]$ . Note that I now use  $f$  not to describe a particular detailed object (i.e. placement of scatterers) which is being imaged by  $T[f]$ , but rather to describe a class of objects (i.e. a Darticular clutter type):  $f$  might for instance specify the moments and autocorrelation function of the distribution of scatterers.  $T[f]$  does not operate directly on  $f$  in this picture: an intermediate scattered field is produced from the scatterers, which is then imaged.

In many cases clutter has spatially homogeneous stochastic properties and also  $T[f]$  is displacement invariant so that

$$P[g|f] = P[L g|f] \quad (4.1)$$

where  $L$  is any translation operator. This reduction is only possible when the data  $g$  is sampled on a lattice with the same invariance under  $L$  (as is usually the case). There has been much work on constructing homogeneous clutter models, and  $K$ -distributions have been found to describe a wide variety of types of clutter [6]. However it is difficult to apply Bayesian methods to these results because the *a posteriori* PDFs are intractable.

I shall therefore adopt a pragmatic approach by deducing what I can about  $P[g|f]$  directly from an image  $g$  of a clutter type  $f$ . Because  $f$  is not explicitly used in my analysis (it is assumed to be fixed), I shall henceforth use the notation  $P[g]$  to mean  $P[g|f]$  except where otherwise stated.

## V. DATA REDUCTION

The first task which faces an experimenter is data reduction without loss of useful information. The information which I must retain in the case of clutter analysis is the spatially averaged statistical properties of  $g$ , and I shall construct a data reduction scheme which fulfils this requirement.

Data reduction is normally obtained by calculating a restricted number of carefully chosen functions of the data, and using the values of these functions for all subsequent analysis. These functions are known as sampling functions, and they are the natural generalisation of the basic sampling scheme (the data itself). A well known example of this type of strategy is the singular value decomposition (SVD) method where it is only necessary to use those singular vectors which have singular values which are large enough to defeat the data noise, thus achieving a reduction in the dimensionality required to represent the data [16]. Thus the SVD sampling functions are singular vectors which project linear combinations of the data sample values. However it is not necessary and usually inefficient to restrict oneself to linear sampling functions when analysing clutter.

I shall obtain the required reduction by defining a transformation

$$g' \equiv S[g] \quad (5.1)$$

where  $S[g]$  is a vector of possibly nonlinear sampling functions of  $g$ , and the dimensionality of  $g'$  is ideally much less than that of  $g$ .  $S[g]$  is an information preserving transformation if

$$P[g|g'] = P[g] \quad (5.2)$$

so that the  $g'$  and  $g$  are equivalent within the framework of Bayesian analysis and information theory.

In the language of statistical analysis a sampling function is a “statistic” [4] of  $g$ , and I shall use these terms interchangeably. Commonly used statistics are the mean, variance, covariance, etc. Furthermore a “sufficient set of statistics” [4] retains all the information that is present in  $g$ , and therefore corresponds to an information preserving transformation  $S[g]$ .

When  $P[g]$  is a Markov random field (MRF) the MRF potentials themselves form a sufficient set of statistics [13], and when there are symmetries present in the structure of the MRF (such as translation invariance) the number of independent potentials can be very small thus

giving a large reduction in dimensionality [13] (see Appendix A for a brief resumé on MRFs). Also any sufficient set of statistics may be used as a basis for the potentials of the corresponding MRF. I have shown [13] how a basis of MRF potentials may be constructed which has the minimum dimensionality which is necessary to guarantee sufficiency: this basis achieves the best data reduction without loss of information that is possible using MRF potentials.

Whilst it is interesting that the MRF potentials form a sufficient set of statistics, it is not practically useful if the form of the MRF is not known in advance. I need to construct an MRF model from an image  $g$  before I can use the model to perform data reduction. Because of the equivalence of MRF potentials and sufficient sets of statistics, the problem therefore reduces to one of constructing a sufficient set of statistics to describe the statistical properties of  $g$ .

## VI. CONSTRUCTING A SET OF STATISTICS

I shall assume throughout this section that the data has homogeneous statistical properties, and so I shall construct all sampling functions  $S[g]$  in the form of spatial averages

$$S[g] \equiv \sum_c s[g_c] \quad (6.1)$$

where  $c$  is a subset of pixel sites whose substate is  $g_c$ , and where the sum is taken over all  $c$  in the equivalence class generated by symmetry transformations  $L$ . In practice this means that an identical  $c$  is associated with each pixel site.

I shall construct the set of sampling functions iteratively by adding new sampling functions to the set until ideally a sufficient set of statistics is obtained. In order to implement this scheme I require a means of determining whether a sampling function measures a significant amount of information which has not already been measured by another sampling function.

Consider a set of sampling functions  $g' = S[g]$  which is insufficient, so that

$$P[g|g'] \neq P[g] \quad (6.2)$$

This inequality provides the means for testing new sampling functions, because any sampling function which can detect a significant difference between  $P[g|g']$  and  $P[g]$  must be measuring information which is not contained in the  $S[g]$ . The data  $g$  provide all the information which is available about  $P[g]$ , so I must provide a means of generating an analogous representation of  $P[g|g']$  in order to implement the difference test. I shall therefore present an algorithm for generating synthetic datasets  $g_{\text{synth}}$  which are sampled from the conditional PDF  $P[g|g']$ . Thus the difference test is a comparison of the statistical properties of  $g$  and  $g_{\text{synth}}$ .

I shall assume that the  $S[g]$  have no intrinsic stochastic character (or hidden variables), so that the same  $g$  always leads to the same  $S[g]$ .  $P[g|g']$  is then given by

$$P[g|g'] = \begin{cases} 0 & S[g] \neq g' \\ b & S[g] = g' \end{cases} \quad (6.3)$$

where  $b$  is constant. The problem of sampling from  $P[g|g']$  is subject to the multiple constraints  $g' = S[g]$ , and so optimisation by simulated annealing (OSA) provides a convenient method of obtaining a solution [9]. The basic idea of OSA is to impose the constraints only weakly at first in an attempt to find approximate solutions, and then to tighten the constraints gradually until an exact solution is found. In my application there are multiple solutions, and I require that they should be found with equal probability.

I shall implement an OSA scheme by defining a PDF

$$P[g_{\text{synth}}] \propto \exp \left\{ -\frac{(S[g_{\text{synth}}] - S[g])^2}{2a^2} \right\} \quad (6.4)$$

which in the limit  $a \rightarrow 0$  yields the required  $P[g|g']$ . The parameter  $a$  may be regarded as the temperature parameter of the simulated annealing, which causes  $g_{\text{synth}}$  to be constrained only weakly when  $a$  is large. The OSA scheme is therefore operated by choosing a large initial temperature, and then running a Monte Carlo Metropolis sampling scheme [15] on  $P[g_{\text{synth}}]$  and gradually reducing the temperature (in what is called an annealing schedule) until  $g_{\text{synth}}$  satisfies the constraint  $P[g|g']$ . Provided that the Monte Carlo scheme is ergodic the permitted  $g_{\text{synth}}$  will be generated with equal probability as required.

Once  $g_{\text{synth}}$  is generated a new sampling function  $S_{\text{new}}[g]$  which is scalar and is given by

$$S_{\text{new}}[g] \equiv \sum_c s_{\text{new}}[g_c] \quad (6.5)$$

may be tested for significant statistical differences between  $g$  and  $g_{\text{synth}}$ . Because the sampling functions are spatial averages the  $t$  test may be used [4]. This takes the form

$$\begin{aligned} x &\equiv \frac{S_{\text{new}}[g]}{N+1} \\ x_{\text{synth}} &\equiv \frac{S_{\text{new}}[g_{\text{synth}}]}{N_{\text{synth}}+1} \\ y^2 &\equiv \frac{\sum_c (s_{\text{new}}[g_c] - x)^2 + \sum_c (s_{\text{new}}[g_{\text{synth},c}] - x_{\text{synth}})^2}{N + N_{\text{synth}}} \\ t &\equiv \frac{x - x_{\text{synth}}}{y} \sqrt{\frac{(N+1)(N_{\text{synth}}+1)}{(N+N_{\text{synth}}+2)}} \end{aligned} \quad (6.6)$$

where  $N$  ( $N_{\text{synth}}$ ) is the number of terms in  $S_{\text{new}}$  for  $g$  ( $g_{\text{synth}}$ ). The value of  $t$  is then looked up in a standard  $t$  distribution table with  $N + N_{\text{synth}}$  degrees of freedom. The level of significance should be chosen to be quite high

in order to ensure that only those  $S[g]$  which distinguish well between  $P[g]$  and  $P[g_{\text{synth}}]$  (or equivalently  $P[g|g']$ ) are included in the set of statistics.

When a new sampling function  $S_{\text{new}}[g]$  has been added to the set of statistics, thus increasing the dimensionality of  $g'$  by one, the synthetic image  $g_{\text{synth}}$  should be recomputed using OSA in order to incorporate the new information which resides in  $g'$ . The process of testing and adding statistics then continues iteratively in this fashion.

The ultimate goal of this iterative process is to build a set of statistics which ensures that  $P[g] = P[g|g']$ , and which therefore preserves all information. However it is not feasible to test for this equality because it would require an exhaustive analysis of  $g$  and  $g_{\text{synth}}$ , so I am forced to rely upon testing various statistics as explained above. This testing procedure can never assure me that a sufficient set of statistics has been built, rather it tells me only that any statistical differences between  $P[g]$  and  $P[g|g']$  are orthogonal to all the tests which I have applied so far.

I am now in a position to present a method of designing a suitable sampling function system when there are a number of different clutter types  $f$  to analyse. For each  $f$  my method will lead to a suitable set of sampling functions which ensure that  $P[f|g'] = P[f|g]$  or a close approximation thereof. I shall denote the set which corresponds to clutter type  $f$  as  $S_f$ . The  $S_f$  will not in general be the same for different  $f$ , but the union  $S_0$

$$S_0 = \bigcup_f S_f \quad (6.7)$$

will retain information for all the clutter types under study. When the number of clutter types is not too great it is feasible to construct  $S_0$  because the total number of sampling functions involved will not be too great.

## VII. RECOVERING AN MRF MODEL FROM THE STATISTICS

In [13] I prove the equivalence of a sufficient set of statistics and the set of potentials of a MRF. In order to find the particular linear combination of the sampling functions which reproduces the MRF potential correctly I must solve the following equations

$$\begin{aligned} P[g] &\propto \exp \left\{ -\sum_c U_c[g_c] \right\} \\ U_c[g_c] &\equiv \mathbf{k} \cdot \mathbf{s}[g_c] \end{aligned} \quad (7.1)$$

for a suitable value of  $\mathbf{k}$ . In cases where the set of statistics is “over-sufficient” there will be multiple solutions for  $\mathbf{k}$ . In [13], I present a derivation of the dimensionality of the basis of potentials (or equivalently sampling functions) of a MRF: this is an upper limit to the number of linearly independent statistics that are required to construct any particular MRF potential.

I shall use a generalisation of what has been called the ‘‘Boltzmann machine learning algorithm’’ (BMLA) [1] to determine a suitable value of  $\mathbf{k}$ : one which ensures that  $S[g]$  produces the same value whether it is applied to the original data or to synthetic data produced by a Monte Carlo realisation of the above PDF. The BMLA determines the set of quadratic interaction terms which are required between a set of two component spins (an Ising model), but a generalisation permits arbitrary interactions between arbitrary state variables to be determined [10]. The algorithm is simple: if  $S[g]$  is too large then increment the corresponding component of  $\mathbf{k}$  by a positive amount and (vice versa if  $S[g]$  is too small). These changes in  $\mathbf{k}$  have the effect of altering  $P[g]$  in such a way as to reduce the error in  $S[g]$ , but the changes must be small in order that interactions amongst the various potentials and their changes do not swamp the desired change in  $S[g]$ .

$P[g]$  is then conveniently represented as a product of simple factors each of which is derived from a linear combination of nonlinear sampling functions. This  $P[g]$  may then be used in a Bayesian analysis of this type of clutter.

### VIII. APPLICATIONS OF MRFS

One application of my method is the discrimination of clutter types each of which has its own (vectorial) value of  $f$ . I shall return to using the notation  $P[g|f]$  because now I am considering multiple clutter types. Using the union  $S_0$  of the sets of sampling functions it is now easy to perform a Bayesian analysis of  $g$  to decide which clutter type caused it. Thus

$$P[f|g] \propto P[f] P[g|f] \quad (8.1)$$

where  $P[f]$  is the *a priori* PDF which expresses the relative frequency of occurrence of each clutter type, which should be set to be a constant if there is no prior knowledge of this information.  $P[g|f]$  may be evaluated for each possible value of  $f$  by using the corresponding MRF potential which now consists of a linear combination of a subset of members of  $S_0$  (by construction). The result of this analysis is an explicit numerical result for the  $P[f|g]$  for each possible  $f$ , which may be used in order to discriminate between the various  $f$ .

Another application of my method is the detection of inhomogeneities which are embedded within what is otherwise clutter with spatially homogeneous statistics. If the inhomogeneity consists of only a small part of all the data then it should not affect the determination of the MRF potential  $U[g]$  adversely because  $f$  is almost a pure single clutter type: I shall therefore assume that  $U[g]$  accurately predicts  $P[g|f]$ . In particular, large values of

$U_c[g_c]$  imply a small  $p[g_c]$ , and so a threshold potential could be set in order to detect when unlikely substates  $g_c$  are encountered. The inhomogeneity is just such an unlikely substate and so it may be detected in this fashion.

There are many further applications of my method in the field of image segmentation where *ad hoc* MRF models have been used extensively in Bayesian reconstruction schemes [5].

### IX. CONCLUSIONS

I have presented a scheme for deriving Bayesian processing of images which have spatially homogeneous statistics. The Bayesian processing is constructed adaptively out of number of sampling functions of the data, whose structure is determined by the form of the data itself. This approach circumvents information losses which inevitably occur with non-adaptive data processing schemes. These results may be used to advantage in circumstances where it is difficult to quantify *a priori* the statistical properties of clutter.

### Appendix A: Markov Random Fields

A Markov random field (MRF) is defined by a factorisable  $P[g]$ , where  $P[g]$  is a product over factors each of which depend on a subset of the co-ordinates; these are also called Gibbs distributions [8]. The generic form of an MRF is

$$P[g] \propto \prod_c p[g_c] \quad (A1)$$

where  $c$  is a subset of pixel sites whose substate is  $g_c$ . This representation may be transformed into the form

$$P[g] \propto \exp \left\{ - \sum_c U_c[g_c] \right\} \quad (A2)$$

where  $U_c[g_c]$  is called a potential: this is the notation which is used in statistical mechanics. Monte Carlo methods may then be used to generate samples from  $P[g]$  with the correct distribution: the Metropolis algorithm is one such Monte Carlo technique [15].

When  $P[g]$  is translationally invariant (under all  $L$ ) the set of all subsets of sites  $C_0$  may be generated thus

$$C_0 \equiv \{\{L \cdot C_1 : \forall L\}, \{L \cdot C_2 : \forall L\}, \dots\} \quad (A3)$$

where  $C_1, C_2$ , etc are the distinct sets of sites which are associated with a single pixel.

---

[1] D H Ackley, G E Hinton, and T J Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Science 9

(1985), no. 2, 147–169.

- [2] H P Baltes, *Inverse source problems in optics: Topics in current physics*, Springer-Verlag, Berlin, 1978.
- [3] ———, *Inverse scattering problems in optics: Topics in current physics*, Springer-Verlag, Berlin, 1980.
- [4] R A Fisher, *Statistical methods for research workers*, Oliver and Boyd, Edinburgh and London, 1958.
- [5] S Geman and D Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence **6** (1984), no. 6, 721–741.
- [6] E Jakeman and R J Tough, *Non-Gaussian models for the statistics of scattered waves*, Advances in Physics **37** (1988), no. 5, 471–529.
- [7] E T Jaynes, *Prior probabilities*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 3, 227–241.
- [8] R Kindermann and J L Snell, *Markov random fields and their applications*, Contemporary Mathematics, vol. 1, American Mathematical Society, Providence, 1980.
- [9] S Kirkpatrick, C D Gelatt, and M P Vecchi, *Optimisation by simulated annealing*, Science **220** (1983), no. 4598, 671–680.
- [10] S P Luttrell, *The implications of Boltzmann-type machines for SAR data processing: a preliminary survey*, Memorandum 3815, Royal Signals and Radar Establishment, Malvern, 1985.
- [11] ———, *A new method of sample optimisation*, Optica Acta **32** (1985), no. 3, 255–257.
- [12] ———, *The use of transinformation in the design of data sampling schemes for inverse problems*, Inverse Problems **1** (1985), no. 3, 199–218.
- [13] ———, *The use of Markov random field models to derive sampling schemes for inverse texture problems*, Inverse Problems **3** (1987), no. 2, 289–300.
- [14] S P Luttrell and C J Oliver, *Prior knowledge in synthetic aperture radar processing*, Journal of Physics D: Applied Physics **19** (1986), no. 3, 333–356.
- [15] N Metropolis, A W Rosenbluth, M N Rosenbluth, A H Teller, and E Teller, *Equations of state calculations by fast computing machines*, Journal of Chemical Physics **21** (1953), no. 6, 1087–1092.
- [16] E R Pike, J G McWhirter, M Bertero, and C de Mol, *Generalised information theory for inverse problems in signal processing*, Proceedings of the IEE F **131** (1984), no. 6, 660–667.
- [17] C E Shannon, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 3, 379–423.
- [18] ———, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 6, 623–656.

# Markov Random Fields: A Strategy for Clutter Modelling \*

Stephen P. Luttrell

*Royal Signals and Radar Establishment, St Andrews Rd, Malvern, Worcestershire, WR14 3PS, UK*

We briefly review the need for models of image texture (or clutter). The use of detailed physical models is prohibited by the difficulty of characterising the scatterer distribution and the scattering process which cause the texture, and so we resort to phenomenological modelling. We propose that certain statistics of the image be measured, and that synthetic images which are consistent with these measurements be generated by a Monte Carlo method - this allows us to test the need for further statistics. We point out the relationship of this scheme to the well-known maximum entropy method for constructing least committal probability distributions. Furthermore we show how a Markov random field model is generated by this process. Finally we suggest some methods of selecting useful statistics, which we demonstrate by analysing a synthetic aperture radar image of a wood and a sonar image of the sea bed.

## I. LIST OF SYMBOLS

$\mathbf{x}$	= sample position in image
$N$	= number of pixels (or samples) in image
$T[\mathbf{x}]$	= image pixel (i.e. sample) value at position $\mathbf{x}$
$\mathbf{s}_j$	= vector of local statistics of $T[\mathbf{x}]$ at pixel $j$
$\mathbf{S}[T]$	= vector of statistics of $T[\mathbf{x}]$
$S_0$	= the value of $\mathbf{S}$ for a particular image
$P[T]$	= PDF over $T[\mathbf{x}]$
$P_{\text{synth}}[T]$	= PDF over synthetic images $T[\mathbf{x}]$
$P[\mathbf{S}]$	= PDF over $\mathbf{S}[T[\mathbf{x}]]$
$Z[\mathbf{S}]$	= the number of $T$ which map to $\mathbf{S}$ (Jacobian)
$E$	= ensemble of images
$M$	= number of images in ensemble
$\langle \mathbf{S} \rangle$	= ensemble average of $\mathbf{S}[T]$

## II. CHARACTERISING TEXTURE IN IMAGES

We shall be concerned with building phenomenological models which describe the structure of textured coherent images. Typical examples of coherent imaging systems are synthetic aperture radar (SAR) and sonar, and such images frequently consist of well defined regions of homogeneous texture. Image texture is often called clutter because it is usually regarded as the background against which objects of interest must be detected. Therefore a good model of image texture is required before we can proceed to the detection and analysis of embedded objects.

The most important property of most of the textures of interest is that they are stochastic rather than deterministic in character, so that for a given texture the image pixel (i.e. sample) values are not rigidly constrained.

Rather, there is a loose relationship or correlation between pixel values which produces the overall impression of texture. There are circumstances where models in which a mixture of stochastic and deterministic behaviour is desirable, such as modelling images of urban regions. Some such techniques have been described elsewhere [1], so we shall not dwell upon these.

When confronted with a textured image we must decide which statistics to measure in order to characterise the image structure. This is not a trivial problem in general, because we do not know in advance what properties of the image need to be measured. The usual approach [1] has been to measure a few simple statistics (e.g. moments, autocorrelation function, mean, variance, entropy, co-occurrence matrix). This method has the advantage of simplicity, but it provides no hint of the necessity or otherwise of the chosen statistics in any particular case.

## III. TEXTURE MODELS

In information theoretic terms the probability density function (PDF) over images having a particular texture contains all the information that is required to derive the properties of images of that texture. It is very difficult (and usually impossible) to derive such a PDF entirely from first principles, so approximations are made in order to construct a physical model of the various processes which underlie the generation of the image. Image formation consists of four essential processes: arrangement of the scatterers, illumination of the scatterers, scattering of the illumination, and focussing of the scattered illumination (or beamforming). Thus a model of the scatterer distribution is used as input to a model of the scattering process, which in turn is used as input to a model of the focussing process: the result of this is a stochastic model of the image texture. We now describe these stages in more detail.

The model which is used to describe correlations in the scatterer distribution is usually highly simplified. In order to construct such a model an understanding of the processes which gives rise to scatterer correlations is re-

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 22, 2019.

This appeared in Proceedings of the AGARD Conference on Scattering and Propagation in Random Media, Rome, 7.1-7.8, 1987.

quired - this is usually hard to express in simple terms. For example when modelling the distribution of scatterers in a wooded area there are many length scales to consider (e.g. clusters of trees, trees, branches, twigs, etc), each of which has its own characteristic properties in general. In such cases it is necessary to use a grossly simplified model of the physical processes which underlie the scatterer distribution. However the variety of textures which can be modelled with a restricted set of physical parameters is severely limited, and so phenomenology is used when constructing more realistic models.

The scattering process is usually modelled using the Born (or single scattering) approximation. The validity of this is not obvious, but it leads to a simple picture of the scattering process which allows the eventual image to be interpreted in the light of the scattering model chosen. Circumstances where this model might break down typically occur when strong specular reflections are possible. Phenomenology may be used to construct more complicated scattering models in circumstances where the single scattering approximation is invalid, but at the cost of complicating image interpretation.

The focussing process itself is well understood in most situations. The point spread function is used to describe this part of the overall imaging model, and it is provided by an independent instrumental calibration or by calibration *in situ* using a bright point target. Phenomenology is not usually required to model this stage of image formation.

#### IV. PHENOMENOLOGICAL TEXTURE MODELS

We have seen that the difficulty of constructing realistic physical models suggests that phenomenology should be used. The trade-off which we seek here is to simplify the construction of the model at the expense of complicating the physical interpretation of the image. If a purely descriptive model of the image is all that is required then a phenomenological model alone will suffice: this occurs when we need only to discriminate between various textured regions in an image, for instance.

There are two complementary approaches to phenomenological texture modelling. Either phenomenology is included only in the model of the scatter distribution, or an entirely phenomenological model of the image itself is developed. An account of the first approach has been given at this meeting by Oliver, so we shall concentrate on the latter approach.

An advantage of the purely phenomenological approach is that it turns out to be very easy to model high order statistical properties (eg  $n$ -point moments). This is important when we wish to model textures with very complicated correlations. The corresponding disadvantage is that the physical origin of the various statistical properties is completely lost in phenomenology. However for the purposes of building a purely descriptive frame-

work within which objects embedded in clutter might be analysed this is acceptable. We should point out that hybrid models may be developed in which physics is used to constrain the structure of some parts of the model, whilst the remainder of the model is purely phenomenological. We shall be using translational symmetry to constrain the structure of our texture models, so strictly our method is a member of the hybrid class.

#### V. THE PROPOSED PHENOMENOLOGICAL MODEL

The model which we propose for texture modelling consists of three parts: the choice and measurement of suitable statistics, the generation of synthetic textured images using the measured statistics, and the refinement of the choice of statistics based upon a comparison of the synthetic and the original textured images. The model is thus built in an iterative fashion whereby the choice of statistics is refined until a satisfactory match between the synthetic and the original images is obtained.

We shall denote position in an image by  $\mathbf{x}$ , the image itself by  $T(\mathbf{x})$ , and the vector of statistics which is obtained from  $T(\mathbf{x})$  by  $\mathbf{S}[T]$ . We shall assume throughout that  $\mathbf{S}[T]$  has the form of a spatial average of some local statistic. Thus

$$\mathbf{S}[T] \equiv \frac{1}{N} \sum_{j=1}^N \mathbf{s}_j \quad (5.1)$$

where  $\mathbf{s}_j$  is a vector of local statistics evaluated at pixel  $j$ , and  $N$  is the number of image pixels.  $\mathbf{S}[T]$  has a PDF which is given by

$$P[\mathbf{S}] = Z[\mathbf{S}] P[T] \quad (5.2)$$

where  $Z[\mathbf{S}]$  is the Jacobian of the transformation  $T \rightarrow \mathbf{S}$  (i.e. the number of distinct  $T(\mathbf{x})$  which map to a single value of  $\mathbf{S}$ ), and  $P[T]$  is the PDF over images  $T(\mathbf{x})$  of a particular texture. We shall denote the particular value of  $\mathbf{S}$  which is measured in a textured image by the symbol  $\mathbf{S}_0$ . The value of  $\mathbf{S}_0$  usually contains insufficient information to determine  $P[\mathbf{S}]$ , but an exception to this occurs when  $N \rightarrow \infty$  so that  $P[\mathbf{S}]$  is asymptotically a delta function.  $\mathbf{S}_0$  then completely characterises  $P[\mathbf{S}]$ , and so an inversion of Equation 5.2 to obtain  $P[T]$  may be attempted.

This asymptotic method relies on the crucial assumption that  $N$  is large enough that fluctuations in the value of  $\mathbf{S}_0$  (measured from independent realisations of the same texture) are small compared to its absolute value. We shall not present a rigorous analysis of the information loss (insofar as specifying  $P[T]$  is concerned) that accompanies such fluctuations, because we shall adopt the viewpoint that  $\mathbf{S} = \mathbf{S}_0$  summarises all the information that we wish to preserve about the image  $T(\mathbf{x})$ . Further information can then be obtained only by measuring further statistics as we shall describe later.

$P[T]$  must satisfy the constraint

$$P[T] = 0 \quad \mathbf{S} \neq \mathbf{S}_0 \quad (5.3)$$

This does not specify what form  $P[T]$  must take when  $\mathbf{S} = \mathbf{S}_0$  because such information is not contained in  $\mathbf{S} = \mathbf{S}_0$ . We emphasise that  $\mathbf{S} = \mathbf{S}_0$  is the sole source of information about the original image, so that we have no choice but to define  $P[T]$  to be constant when  $\mathbf{S} = \mathbf{S}_0$ : this is the least committal way of performing the reconstruction of the PDF  $P[T]$ . Only one of the images which is permitted by such a PDF is the original image, but the set of all permitted images forms an ensemble which is representative of the actual information which is contained in  $\mathbf{S} = \mathbf{S}_0$  alone.

We then generate synthetic images from the permitted set by defining an approximate  $P[\mathbf{S}]$  by

$$P[\mathbf{S}] \propto \exp \left\{ -\frac{(\mathbf{S} - \mathbf{S}_0)^2}{2a^2} \right\} \quad (5.4)$$

which has the correct form in the limit  $a \rightarrow 0$ . We have chosen to use a Gaussian PDF purely for convenience; it has no fundamental significance. We then define a  $P[T]$  by inserting Equation 5.4 into Equation 5.2 ignoring the contribution of the  $Z[\mathbf{S}]$  factor which is effectively constant in the limit  $a \rightarrow 0$ . We then use a Monte Carlo (MC) technique to sample from  $P[\mathbf{S}]$  by repeatedly attempting to update a pixel values according to the prescription

$$\begin{aligned} T[\mathbf{x}] &\rightarrow T'[\mathbf{x}] \text{ with probability } \frac{P'}{P + P'} \\ P &\equiv P[\mathbf{S}[T]] \\ P' &\equiv P[\mathbf{S}[T']] \end{aligned} \quad (5.5)$$

The order in which we select pixels for updating is random in order to ensure that translational symmetry is preserved by the MC algorithm. We refer to a set of  $N$  such updates as a Monte Carlo cycle, because such a set will cause an average of one attempted update of each pixel in an  $N$  pixel image. The particular update  $T \rightarrow T'$  which is attempted at any stage may be chosen flexibly, because the properties of the MC method guarantee convergence provided that the sequence of images generated is ergodic. In our work we have represented pixel values as integers in the range [0, 255], and we have selected at random within this range new pixel values to be attempted according to Equation 5.5. Other prescriptions could be used in order to enhance the ability of the MC algorithm to explore the set of allowed images.

The parameter  $a$  is steadily decreased towards zero whilst the MC algorithm proceeds, until finally  $\mathbf{S} = \mathbf{S}_0$  and a sample synthetic image emerges. This particular way of using the MC is known as optimisation by simulated annealing (OSA) by analogy with the process that is used in crystal growth to remove unwanted defects [4]. The parameter  $a$  in Equation 5.4 is usually

referred to as the temperature, and the sequence of values of  $a$  which is used by the OSA algorithm is called the annealing schedule. We offer no simple prescription for a general annealing algorithm because the best choice is highly problem dependent. However in our work (see the results described later) we have found that 5 Monte Carlo cycles at each temperature is sufficient to allow equilibration, and that the temperature may be reduced by a factor of 5 after each set of 5 cycles. The initial temperature was chosen to be sufficiently high that the statistics were only weakly constrained.

## VI. RELATIONSHIP TO ENTROPY MAXIMISATION

The OSA algorithm which was described above leads to a synthetic image  $T_0[\mathbf{x}]$  which is sampled noncommittally from the set of images which satisfy the constraint  $\mathbf{S} = \mathbf{S}_0$ , and we shall denote the PDF over synthetic images by  $P_{\text{synth}}[T]$ .  $P_{\text{synth}}[T]$  has a very rich underlying structure which is derived as follows.

Consider an ensemble  $E$  of images defined by

$$E \equiv \{T_1, T_2, \dots, T_M\} \quad (6.1)$$

where each member of  $E$  is drawn independently from  $P[T]$ . Define the average statistic

$$\langle \mathbf{S} \rangle \equiv \frac{1}{M} \sum_{j=1}^M \mathbf{S}[T_j] \quad (6.2)$$

When the number of image pixels  $N$  is large enough that the relative fluctuations of  $\mathbf{S}_0$  are negligibly small the constraint  $\langle \mathbf{S} \rangle = \mathbf{S}_0$  is equivalent to the constraint  $\mathbf{S} = \mathbf{S}_0$ . Now we wish to find that PDF over images  $T[\mathbf{x}]$  which maximises the number of ways in which the ensemble  $E$  can be generated (using independent samples) consistent with the constraint  $\mathbf{S} = \mathbf{S}_0$ .

The optimisation problem posed above is easily solved, and gives rise to the familiar maximum entropy solution when  $M$  is large [2]

$$P_{\text{synth}}[T] \propto \exp \{-\mathbf{k} \cdot \mathbf{S}[T]\} \quad (6.3)$$

where the constant vector  $\mathbf{k}$  must be chosen so that  $\langle \mathbf{S} \rangle = \mathbf{S}_0$ . The existence of such a maximum entropy solution is not invariably guaranteed for a particular constraint  $\langle \mathbf{S} \rangle = \mathbf{S}_0$ , and failure to obtain a solution indicates that the set of statistics being used is insufficient to characterise the texture fully. The converse does not hold however: an insufficient set of statistics does not preclude the existence of a maximum entropy solution.

Recall that we are considering only those statistics  $\mathbf{S}[T]$  which have the form of a spatial average as defined in Equation 5.1. Equation 6.3 may now be written as

$$P_{\text{synth}}[T] \propto \prod_{j=1}^N \exp \{-\mathbf{k} \cdot \mathbf{s}_j\} \quad (6.4)$$

The simple global constraint  $\mathbf{S} = \mathbf{S}_0$ , which was expressed in Equation 5.3 has therefore induced a rich structure in  $P_{\text{synth}}[T]$  which is expressed in Equation 6.4. The  $j$ th term in the product is dependent only on the local vector of statistics  $\mathbf{s}_j$ , and so the overall probability measure is a product of factors with only local pixel dependencies.

## VII. RELATIONSHIP TO MARKOV RANDOM FIELD THEORY

The form for  $P_{\text{synth}}[T]$  which was obtained in Equation 6.4 defines a Markov random field (MRF) [3]. The form of the statistics vector  $\mathbf{S}[T]$  may be chosen arbitrarily in principle, and so a correspondingly complicated MRF structure may be generated according to Equation 6.4. The scalar product  $\mathbf{k} \cdot \mathbf{s}_j$  records the information which is contained in the constraint  $\mathbf{S} = \mathbf{S}_0$ : the form of  $\mathbf{S}$  is recorded in the  $\mathbf{s}_j$  vector, and the particular value  $\mathbf{S}_0$  which is measured is recorded in the  $\mathbf{k}$  vector.

The OSA algorithm provides a quick way of assessing the usefulness of a particular set of statistics  $\mathbf{S}$  for modelling a textured image. Derivation of the vector  $\mathbf{k}$  requires more extensive computation, and so the MRF formulation should only be derived if the detailed form of the  $P_{\text{synth}}[T]$  which is implied by  $\mathbf{S} = \mathbf{S}_0$  is required.

An advantage of the MRF formulation is that an explicit PDF is obtained, which may be used to derive explicit conditional probabilities. For instance the ratio of probabilities that a pixel can take each of two possible values whilst all other pixel values are held constant is given by

$$\frac{P[T']}{P[T]} = \prod_j \exp \left\{ -\mathbf{k} \cdot (\mathbf{s}'_j - \mathbf{s}_j) \right\} \quad (7.1)$$

where the product is over only those factors in which  $\mathbf{s}_j$  involves the pixel of interest. This expression may be used to derive the MRF formulation of the MC update scheme in Equation 5.5. Another advantage of the MRF formulation which emerges from Equation 7.1 if the  $\mathbf{s}_j$  are local statistics is that the MC update scheme depends only on local computations.

## VIII. CHOICE OF STATISTICS

We could adopt the assumption that nothing at all is known about the structure of the image prior to making any measurements. However in practice there are many useful strategies for designing statistics which facilitate model building. We have assumed throughout that the statistical properties of the image are translation invariant (see Equation 5.1). This is a very useful approximation, and it provides a good starting point for image analysis.

We must now make additional assumptions about which particular form of  $\mathbf{S}$  should be chosen. We can make much progress by appealing to simple dimensional arguments. For instance if we suspect that the image structure contains certain length scales, then it is useful to construct statistics which are sensitive to structure on such length scales. A particular example of this strategy occurs when we identify the scale which corresponds to the point spread function width of the imaging system. The image will usually be locally smooth on such a scale, so a statistic which responds to the presence or absence of such local smoothness is essential. Other length scales will derive from the structure of the object which is being imaged - these are more difficult to quantify.

Another fundamentally important point to observe when constructing  $\mathbf{S}$  is to ensure as far as possible that its various components measure independent image information. A priori this is an impossible condition to satisfy because we do not know in advance what we are measuring. An interesting case arises however when it is known in advance that the image is described by an MRF model: a necessary and sufficient set of statistics may then be constructed [5]. However more generally there is a simple rule of thumb which may be used to reduce substantially the interdependencies of the various statistics.

For simplicity we shall describe a one dimensional  $n$  pixel image  $T(\mathbf{x})$ , which is written out as a vector of pixel intensities

$$T(\mathbf{x}) = (I_1, I_2, \dots, I_N) \quad (8.1)$$

Define a doublet of local statistics thus

$$\mathbf{s}_j = (I_j, I_j I_{j+1}) \quad (8.2)$$

The first component of  $\mathbf{s}_j$  measures the mean pixel intensity, and the second component measures the autocorrelation for a one pixel displacement. This is an example of a poor choice of statistics because  $\langle I_j I_{j+1} \rangle$  implicitly contains a lot of information about  $\langle I_j \rangle$ . A far better choice is the doublet

$$\mathbf{s}_j = (I_j, I_j I_{j+1} - \langle I_j \rangle \langle I_{j+1} \rangle) \quad (8.3)$$

where the second component is now zero when  $I_j$  and  $I_{j+1}$  have their mean values. This new component measures the degree of correlation of the intensities of adjacent pixels.

The general principle which underlies the above strategy may be written in a recursive manner. Thus define the basic image statistic by

$$s_j^{(1)} = I_j \quad (8.4)$$

from which all other statistics will be derived. Define a derived statistic by

$$s_j^{(2)} = f(s_j^{(1)}, s_{j+1}^{(1)}) - f(\langle s_j^{(1)} \rangle, \langle s_{j+1}^{(1)} \rangle) \quad (8.5)$$

where  $f$  is a suitably chosen function of two variables. Clearly when  $f(z_1, z_2) = z_1 z_2$  the doublet in Equation 8.3 is recovered. Equation 8.5 may be generalised further by using it recursively to construct further statistics. The function  $f$  need not be the same at each stage of the recursion, nor need it be a function of two variables only. If the first term in Equation 8.5 has the general form  $f(z_1, z_2, \dots, z_k)$ , then the second term should have the form  $f(\langle z_1 \rangle, \langle z_2 \rangle, \dots, \langle z_k \rangle)$ . This recursive method of defining new statistics has the advantage that the second term can always be obtained from existing measurements of the spatial average of simpler statistics, and so the computational overhead in producing a desirable behaviour for the new statistic is acceptable. It is important to note that this method has been justified on the grounds of simplicity alone: it is not guaranteed to produce anything more than a qualitative improvement in the independence of the statistics.

A problem may arise in the OSA algorithm when the components of  $\mathbf{S}_0$  have widely differing absolute values. For instance this may occur when the dimensions of the various statistics are different (e.g. different powers of intensity). Either the statistics should be redefined so that they have similar absolute values, or Equation 5.4 should be replaced by

$$P[\mathbf{S}] \propto \exp \left\{ - \sum_j \frac{(S_j - S_{0,j})^2}{2a_j^2} \right\} \quad (8.6)$$

where now the  $a_j$  define different temperatures for each statistic. Typically the ratio of the various  $a_j$  should be chosen to be equal to the ratio of the corresponding  $S_{0,j}$  in order to ensure that the  $S_j - S_{0,j}$  are given equal weight in Equation 8.6.

## IX. RESULTS

We shall now examine how a simple set of statistics can successfully model textured SAR and sonar images. For simplicity we shall restrict ourselves to using the first six moments and some local 2-point statistics. Thus we define the intensity of pixel  $(j, k)$  of an image to be  $I_{j,k}$ , and then we define the vector of local statistics to be of the form

$$\mathbf{s}_{j,k} = (s_{j,k}^{(0)}, s_{j,k}^{(1)}, s_{j,k}^{(2)}, s_{j,k}^{(4)}, s_{j,k}^{(8)}) \quad (9.1)$$

where

$$\begin{aligned} \mathbf{s}_{j,k}^{(0)} &= (I_{j,k}, (I_{j,k})^2, \dots, (I_{j,k})^6) \\ \mathbf{s}_{j,k}^{(1)} &= I_{j,k} (I_{j+1,k}, I_{j+1,k+1}, I_{j,k+1}, I_{j-1,k+1}) \\ \mathbf{s}_{j,k}^{(2)} &= I_{j,k} \left( \begin{array}{c} I_{j+2,k}, I_{j+2,k+1}, I_{j+2,k+2}, I_{j+1,k+2}, \\ I_{j,k+2}, I_{j-1,k+2}, I_{j-2,k+2}, I_{j-2,k+1} \end{array} \right) \\ \mathbf{s}_{j,k}^{(4)} &= I_{j,k} \left( \begin{array}{c} I_{j+4,k}, \dots, I_{j+4,k+4}, \\ I_{j+3,k+4}, \dots, I_{j-4,k+4}, \\ I_{j-4,k+3}, \dots, I_{j-4,k+1} \end{array} \right) \\ \mathbf{s}_{j,k}^{(8)} &= I_{j,k} \left( \begin{array}{c} I_{j+8,k}, \dots, I_{j+8,k+8}, \\ I_{j+7,k+8}, \dots, I_{j-8,k+8}, \\ I_{j-8,k+7}, \dots, I_{j-8,k+1} \end{array} \right) \end{aligned} \quad (9.2)$$

where  $\mathbf{s}_{j,k}^{(0)}$ ,  $\mathbf{s}_{j,k}^{(1)}$ ,  $\mathbf{s}_{j,k}^{(2)}$ ,  $\mathbf{s}_{j,k}^{(4)}$ , and  $\mathbf{s}_{j,k}^{(8)}$  are understood to have the product of means subtracted out as prescribed in Equation 8.5. The four components of  $\mathbf{s}_{j,k}$  measure the degree of correlation between  $I_{j,k}$  and the nearest pixel in the east, north-east, north and north-west directions. The remaining points of the compass are contained in  $\mathbf{s}_{j+1,k}$ ,  $\mathbf{s}_{j+1,k+1}$ ,  $\mathbf{s}_{j,k+1}$ , and  $\mathbf{s}_{j-1,k+1}$ . The purpose of remaining statistics is to measure the analogous degrees of correlation at separations of 2, 4 and 8 pixels, where the pixel separation is the half-length of the side of a square (with sides parallel to the north-south and east-west directions) centred on one pixel and passing through the other pixel. This choice of statistics is somewhat arbitrary and possibly redundant, but it will suffice to demonstrate the power of our technique.

The image in Figure 1(a) is a  $128 \times 128$  SAR image of a wooded region.

There are several distinctive features to note in this image. Firstly there is an obvious correlation in the north-south direction which smooths the image on a length scale of about two pixels. This anisotropy is caused by the different physical means which are used to create two orthogonal components of the (factorisable) point spread function in the SAR. The image also has a clumpy structure which has a range of length scales ranging from a few pixels out to many tens of pixels. However the dominant structure is found at length scales less than of order 10-20 pixels. There are also a few much darker regions in the image: these are regions which are not illuminated by the radar (i.e. shadows).

The  $64 \times 64$  synthetic images which are shown in Figure 1(b) to Figure 1(e) correspond respectively to using  $\mathbf{s} = (\mathbf{s}^{(0)}, \mathbf{s}^{(1)})$  to  $\mathbf{s} = (\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \mathbf{s}^{(3)}, \mathbf{s}^{(4)})$ .

Thus each successive image has an extra set of statistics included. There is a clear improvement in the quality of the synthetic image as longer range statistics are introduced, until finally Figure 1(e) shows a marked similarity to Figure 1(a). There are nevertheless discrepancies which remain because there are no statistics which are sensitive to structure on a scale greater than 8 pixels, nor are there any statistics which depend on more than 2 pixel values. It is interesting to note that the set of statistics which is used to produce Figure 1(e) is highly

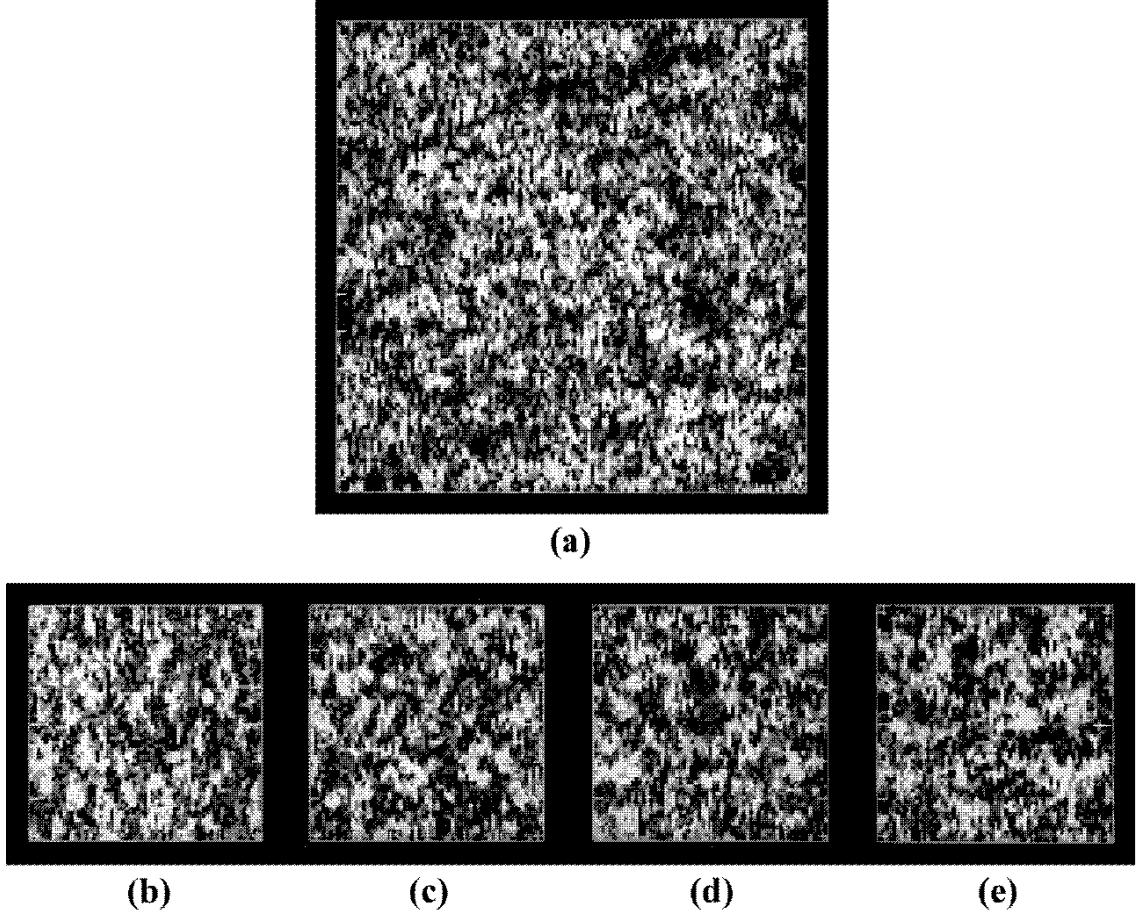


Figure 1:

redundant. A more careful approach where a statistic is added to the existing set of statistics only when inconsistencies between the original and the synthetic images are observed gives a much more efficient representation of the textured image structure.

The  $128 \times 128$  image in Figure 2(a) is a sonar image of the sea bed.

It shows a completely different texture to that seen in Figure 1(a), and so it provides an independent test of our method. Figure 2(b) to Figure 2(e) are produced in an analogous fashion to Figure 1(b) to Figure 1(e).

As before the synthetic image quality improves as we pass from Figure 2(b) to Figure 2(e), with the texture of Figure 2(e) showing a marked similarity to that of the original image in Figure 2(a).

The  $64 \times 64$  image in Figure 3(a) is another sonar image of the sea bed which shows the presence of sandwaves.

This image has been preprocessed to reduce its resolution by a factor of 8 in order to bring the characteristic structure within the length scale (8 pixels) measured by the statistics which we are using. This image is not ideally suited to our method because its statistical properties are not homogeneous. However the synthetic image obtained using the statistics of Equation 9.1 which

is shown in Figure 3(b) again has a marked similarity to the original.

This result is probably fortuitous because the statistics measured from Figure 3(a) are a spatial average over an inhomogeneous texture, and so represent no single texture within the image. To a much lesser extent this remark also applies to the images in Figure 1(a) and Figure 2(a).

## X. CONCLUSIONS

The results which we have presented demonstrate that our synthetic image generation method produces good results when using a limited set of simple statistics. This is even more impressive when the set is pruned to remove redundant statistics. It is interesting to note that we have produced a workable scheme for modelling textured images using only translation invariance, and simple dimensional arguments. Of course we do not pretend that all textures can be modelled using the 1-point and 2-point statistics of Equation 9.2; these were used only to demonstrate how our method is used in simple cases. We have shown how it is possible to use MRF models to de-

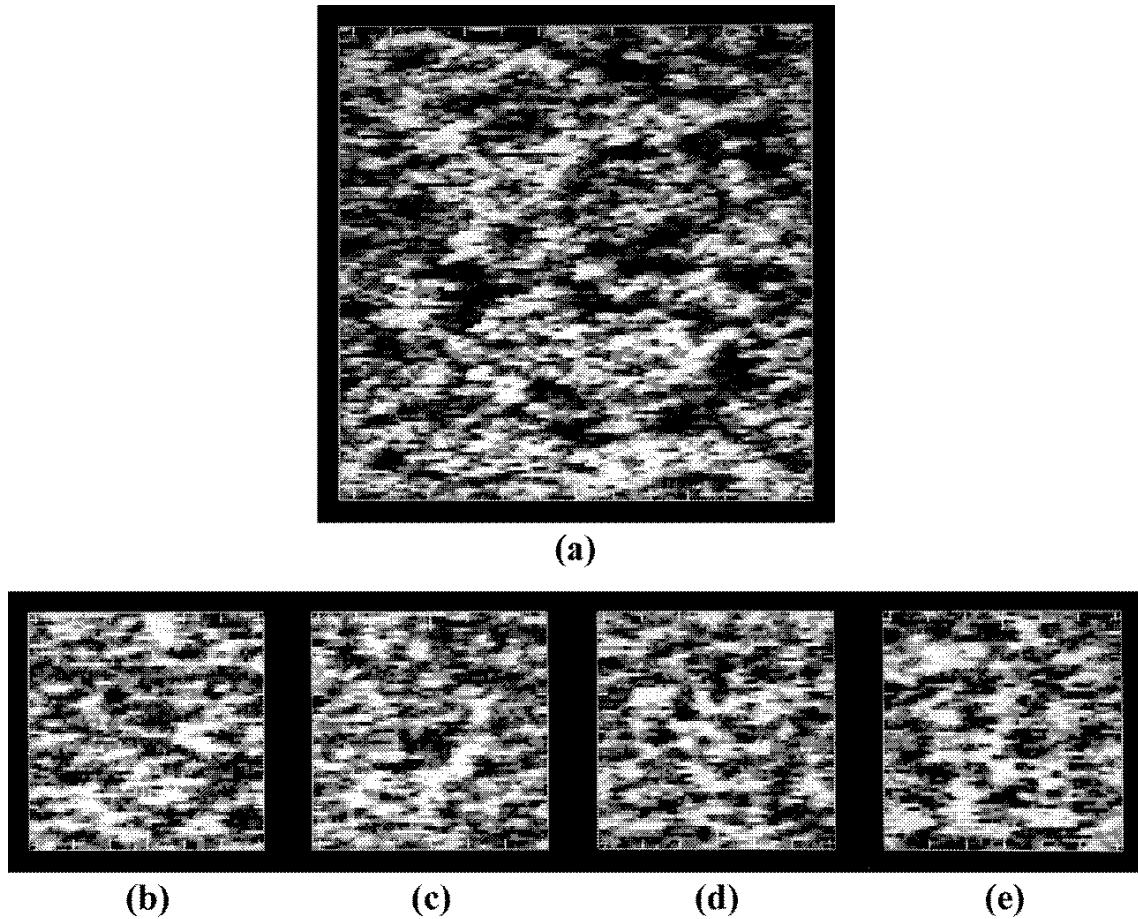


Figure 2:

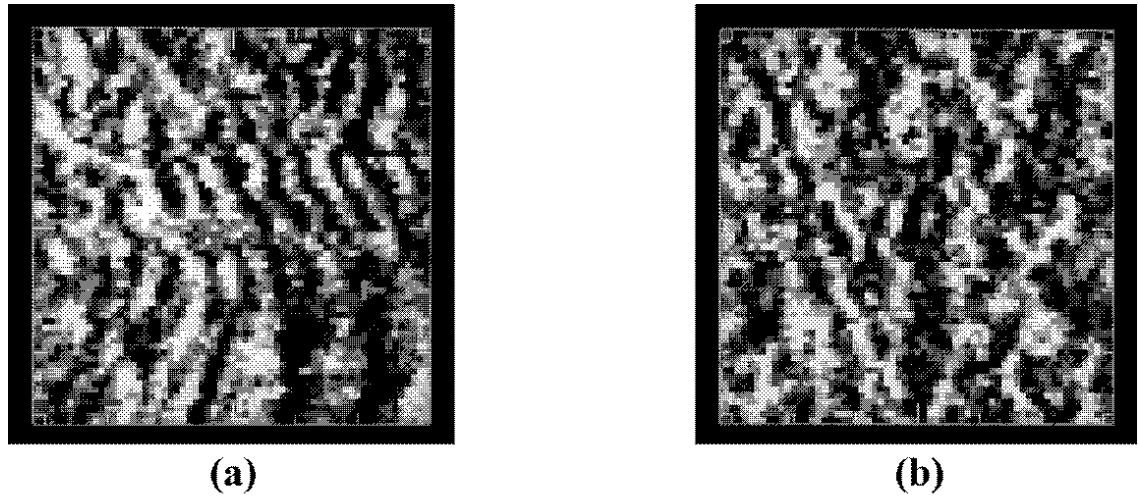


Figure 3:

scribe textures without actually solving explicitly for the MRF parameters. This is a great computational saving in circumstances where such parameters are not explicitly required, and it provides a significant advance over previous MRF modelling schemes for textured images [1].

An example of the type of image which will require

more complicated statistics is a SAR image of a town, where there are many extended linear and angular features. We will then need high-order statistics which respond to and distinguish between such features. The power of our method is such that we can systematically design a set of such statistics which extracts only that information from the textured image  $T[\mathbf{x}]$  which is required to describe its textural structure. It is interesting to note that the use of high order statistics can produce models in which there is a strong deterministic element, so we see that the separation of models into stochastic and deterministic classes is somewhat artificial. The MC algorithm must be very carefully designed when high-order statistics are used because there is a real possibility that ergodicity might not be achieved during the annealing process.

The choice of statistics is not entirely arbitrary, because physical insight is available to narrow down the variety of possible statistics which might be considered. This strategy will ensure that the set of statistics represents the textural structure of  $T[\mathbf{x}]$  economically - translational invariance is a trivial example of the use of a physical constraint to build a model. The optimum trade-off between phenomenology and physics must be chosen to match the particular type of image and prior knowledge that is available.

Finally the answer to the question “Markov random fields: a strategy for clutter modelling?” is emphatically “yes”, because of their flexibility in describing a great variety of textured images, and because of the great computational saving which is offered by our technique.

- 
- [1] L Van Gool, P Dewaele, and A Oosterlinck, *Texture analysis anno 1983*, Computer Vision, Graphics, and Image Processing **29** (1985), no. 3, 336–357.
  - [2] E T Jaynes, *Prior probabilities*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 3, 227–241.
  - [3] R Kindermann and J L Snell, *Markov random fields and their applications*, Contemporary Mathematics, vol. 1, American Mathematical Society, Providence, 1980.
  - [4] S Kirkpatrick, C D Gelatt, and M P Vecchi, *Optimisation by simulated annealing*, Science **220** (1983), no. 4598, 671–680.
  - [5] S P Luttrell, *The use of Markov random field models to derive sampling schemes for inverse texture problems*, Inverse Problems **3** (1987), no. 2, 289–300.

# Markov Random Field Image Models of Targets and Clutter \*

S P Luttrell

*Royal Signals and Radar Establishment, Great Malvern, WORCS, UK*

We review Markov random field theories of radar clutter, and we explain their structure in terms of sampling functions and information theory. We explain some recent advances in Markov random field theories of target imaging, and explain their relationship in information theoretic terms to super-resolution.

## I. RADAR IMAGES

In remote sensing applications such as radar imaging of terrain or sea the raw data is presented as an image (or sequence of images) consisting of an array of pixel values. It is not easy to automate the interpretation of such images because the sequence of processes which leads to image formation is not usually easily defined. Loosely speaking this sequence consists of:

(1) Illumination (coherent or incoherent) with EM radiation

(2) Scattering of the radiation.

(3) Receiving the scattered radiation

(4) Processing to obtain the image

Stages 1, 3 and 4 are usually well understood, but stage 2 is always difficult to characterise.

The primary role of image interpretation is to provide the answers to specific questions (such as "Is there a target present. If so at what location, and what is it?"). This is achieved by invoking all of the processes which underlie stage 2, which involves not only specifying how radiation scatters (i.e. Maxwell's equations, or a watered down version thereof) but also a specification of the types of objects which we expect to be present a priori.

Conventionally we decompose the world into two classes of objects:

(i) Clutter

(ii) Targets

where CLUTTER comprises all those objects in which we have no interest, and TARGETS comprises the set of objects of interest. It is important to note that this definition is subjective.

We shall be concerned with building models of radar clutter and targets which satisfy two criteria:

(i) The models have an underlying physical basis

(ii) The models lead naturally to parallel image processing algorithms

Previous methods of automatic image interpretation have not usually satisfied both these criteria.

The methods which we outline here are covered in more detail in Luttrell and Oliver [7], and Luttrell [4–6].

## II. CLUTTER

Clutter pervades most of a radar image and its properties must be understood before any attempt at target extraction is made. We shall model clutter as a quasi-stationary spatial stochastic process, and we shall denote the sample values which comprise a clutter image by the vector  $\mathbf{T}$ ; the letter "T" is suggested by the textured appearance of clutter. We shall derive our results for a one dimensional image, but they are easily generalised to two dimensions. For an  $N$  pixel image  $\mathbf{T}$  is given explicitly by

$$\mathbf{T} \equiv (T_1, T_2, \dots, T_N) \quad (2.1)$$

where component  $T_k$  ( $k = 1, \dots, N$ ) is the sample value of pixel  $k$ . The PDF  $P[\mathbf{T}]$  specifies the stochastic properties of the clutter, and we shall define a vector of statistics  $\mathbf{S}[\mathbf{T}]$  and its average value  $\langle \mathbf{S} \rangle$  measured over  $P[\mathbf{T}]$

$$\langle \mathbf{S} \rangle \equiv \int [d\mathbf{T}] P[\mathbf{T}] \mathbf{S}[\mathbf{T}] \quad (2.2)$$

We shall restrict our attention to statistics of the form

$$\mathbf{S}[\mathbf{T}] \equiv \frac{1}{N} \sum_L \mathbf{s}[L \mathbf{T}] \quad (2.3)$$

where  $\mathbf{s}[\mathbf{T}]$  is a vector of local functions (ideal for a parallel processor) and  $L$  is an image translation operator. The PDF over statistics  $P[\mathbf{S}]$  in terms of  $P[\mathbf{T}]$  by

$$P[\mathbf{S}] = \int [d\mathbf{T}] P[\mathbf{T}] \delta[\mathbf{S} - \mathbf{S}[\mathbf{T}]] \quad (2.4)$$

where the delta function restricts the integration to the feasible set of  $\mathbf{T}$ .

We shall assume that the image size is sufficiently large that

$$\langle \mathbf{S} \rangle \simeq \mathbf{S} \quad (2.5)$$

to a good approximation; strictly this requires that the image is ergodic. We may now recover  $P[\mathbf{T}]$  from the measured statistics  $\mathbf{S}$  by using the maximum entropy method of Jaynes [2], which leads to a non-committal estimate  $Q[\mathbf{T}]$  of  $P[\mathbf{T}]$

$$Q[\mathbf{T}] = \frac{1}{Z} \prod_L \exp(-\lambda_s[L \mathbf{T}]) \quad (2.6)$$

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This appeared in Proceedings of the 1st US/UK Workshop on Parallel Processing, pp. 786-793, 1987. © Controller, Her Majesty's Stationery Office, London, 1987.

$Q[\mathbf{T}]$  is a product of local factors and describes a Markov random field (MRF), Kindermann and Snell [3].

The  $\lambda$  parameters are obtained by using the “Boltzmann machine learning algorithm”, Ackley et al [1], from neural network theory. The constraints which we must satisfy are summarised by

$$\langle \mathbf{S} \rangle_Q = \langle \mathbf{S} \rangle_P \quad (2.7)$$

where the subscripts indicate the PDF which is used in the averaging process. Define a “Lyapunov function”  $G$  by

$$G \equiv - \int [d\mathbf{T}] P[\mathbf{T}] \log \left( \frac{P[\mathbf{T}]}{Q[\mathbf{T}]} \right) \quad (2.8)$$

where  $G \leq 0$ , and  $G = 0$  if and only if  $P[\mathbf{T}] = Q[\mathbf{T}]$  for all  $\mathbf{T}$ . The gradient of  $G$  with respect to  $\lambda$  is then given by

$$\text{grad}(G) = \langle \mathbf{S} \rangle_Q - \langle \mathbf{S} \rangle_P \quad (2.9)$$

so that the constraint equation now becomes

$$\text{grad}(G) = \mathbf{0} \quad (2.10)$$

The goal is therefore to maximise  $G$  with respect to  $\lambda$  by using the difference  $\langle \mathbf{S} \rangle_Q - \langle \mathbf{S} \rangle_P$  to estimate the direction in which to vary  $\lambda$  at each stage. This stage involves Monte Carlo simulations for which a parallel processing architecture is ideally suited.

We now require a method of building up a set of statistics which is suited to a particular clutter type. The key to this procedure is the expression for  $\text{grad}(G)$ . Suppose we already have an  $m$  dimensional set of statistics  $\mathbf{S}[\mathbf{T}] = (S_1[\mathbf{T}], \dots, S_m[\mathbf{T}])$  and its associated  $\lambda$  which satisfy  $\text{grad}(G) = \mathbf{0}$ . Then any novel statistic  $S_{m+1}[\mathbf{T}]$ , for which  $\lambda_{m+1} = 0$  currently, may be inserted into the expression for  $\text{grad}(G)$  to estimate the value of  $\frac{\partial G}{\partial \lambda_{m+1}}$ . If this gradient is close to zero then  $S_{m+1}$  is redundant in the sense that it does not measure any structure in  $P[\mathbf{T}]$  which has not already been measured by  $\mathbf{S}[\mathbf{T}]$ , and so should be discarded as a worthless additional measurement to make. However, if the gradient is large then  $S_{m+1}[\mathbf{T}]$  is measuring novel structure in  $P[\mathbf{T}]$  and so it should be considered as a viable candidate for inclusion in the set of statistics. This process may be iterated to build up an economical set of statistics by rejecting any statistics which do not measure novel structure in  $P[\mathbf{T}]$ . It is possible to take advantage of the ergodicity of the clutter image when building such a set of clutter statistics, which removes the need to use the “Boltzmann machine learning algorithm” to calculate  $\lambda$  at each stage: this leads to a significant speed-up of the algorithm.

$Q[\mathbf{T}]$  is now expressed as a product of local factors (see Equation 2.6)

$$Q[\mathbf{T}] = \prod_L q[L \mathbf{t}] \quad (2.11)$$

where  $\mathbf{t}$  is a suitably chosen local set of pixels. For a particular clutter image  $\mathbf{T}$  the  $q[L \mathbf{t}]$  may also be displayed as an image if so desired, because as  $L$  ranges over all possible translations the function  $q[L \mathbf{t}]$  is separately defined at every pixel position; we shall call such an image a probability image.  $Q[\mathbf{T}]$  is then the product of the pixel values of the probability image. Loosely speaking bright areas in the probability image correspond to parts of the clutter which are typical of the whole clutter image, and dark areas are atypical. This property may be used to determine the presence and location of inhomogeneities (or targets embedded in clutter) in the original image by setting a threshold below which the value of  $q[\mathbf{t}]$  is considered to be so low that the presence of an inhomogeneity is suspected.

A more rigorous Bayesian decision may be made if the PDF which describes the type of targets being sought in the clutter is available. A Bayesian framework may then be constructed in which the relative probability of object+clutter versus clutter alone is calculated. For objects which typically occupy only one pixel and for which their one pixel PDF is a sufficient description it is simple to construct the Bayesian framework. Thus assume that the object PDF is given by  $O(\mathbf{T})$ , then the likelihood ratio  $R$  that an object is present at pixel  $k$  is given by

$$R = \frac{O(T_k) \int dT_k Q[\mathbf{T}]}{Q[\mathbf{T}]} \quad (2.12)$$

Introducing the product form of Equation 2.11 leads to a lot of cancellation which finally yields

$$R = \frac{O(T_k) \int dT_k \prod q[L \mathbf{t}]}{\prod q[L \mathbf{t}]} \quad (2.13)$$

where the product over  $L$  involves only those  $q[L \mathbf{t}]$  which depend on  $T_k$ ; a small number of factors in general.

This procedure for obtaining the likelihood ratio  $R$  will yield a result whose accuracy is limited by the range of the local statistics, but which otherwise is a completely flexible and data driven scheme. In a sense it is a generalisation of the constant false alarm rate (CFAR) method of target detection to incorporate a more sophisticated adaptation algorithm to adjust to the local clutter statistics.

### III. TARGETS

Target analysis can proceed once detection has occurred within the context of a particular clutter model, as above. In order to make progress we require an a priori model of what targets look like in order to interpret the target image. We shall not dwell on detailed models here, rather we shall investigate to what extent limited prior knowledge of typical target structure may be used.

We shall use a different notation in this section because we are dealing not with a large clutter image, but with a small region which is deemed to be target. Furthermore

we shall be concerned only with coherent radar images which therefore have complex pixel values. Introduce the following definitions

- $\sigma$  = Scattering cross section (function of position)
- $f$  = Complex radiation field scattered from target
- $U$  = Linear operator for image formation
- $g$  = Complex image data
- $n$  = Complex image noise

These are related by

$$g = Uf + n \quad (3.1)$$

A simple stochastic physical model for the scattering process itself is

$$P[f] \propto \exp(-f^\dagger \sigma^{-1} f) \quad (3.2)$$

where each component of  $f$  has an independent Gaussian distribution controlled by the local underlying cross section. A suitable stochastic model of the image formation process is

$$P[g|f] \propto \exp(-\nu^{-1}|g - Uf|^2) \quad (3.3)$$

where  $n$  has been modelled as white Gaussian noise with variance  $\nu$ .

The a posteriori PDF  $P[f|g]$  is then given by Bayes' theorem

$$\begin{aligned} P[f|g] &\propto P[g|f] P[f] \\ &\propto \exp[-(f - f_0)^\dagger M (f - f_0)] \end{aligned} \quad (3.4)$$

where

$$\begin{aligned} f_0 &\equiv \sigma U^\dagger [U \sigma U^\dagger + \nu I]^{-1} g \\ M &\equiv \nu^{-1} U^\dagger U + \sigma^{-1} I \end{aligned} \quad (3.5)$$

$f_0$  is the maximum a posteriori (MAP) reconstruction of the field  $f$ , and it may be presented as a single representative interpretation of  $g$  given an estimate of the cross section  $\sigma$ . When  $\sigma$  has the form of a localised region of strong scattering surrounded by an extended weakly scattering region,  $f_0$  typically shows many target details which were not present in  $g$ . This is called SUPER-RESOLUTION, and this technique (and closely related techniques) are increasingly being used in the interpretation of coherent radar and optical images.

Prior knowledge of target structure is introduced by controlling what forms  $\sigma$  is permitted to take. In the above super-resolution method this is done in an ad hoc fashion, and so we shall suggest a more rigorous alternative.

The sequence which leads to the final complex image is

$$\sigma \longrightarrow f \longrightarrow g \quad (3.6)$$

and we are interested in reconstructing  $\sigma$  (not  $f$ ). Using Bayes, theorem we obtain the a posteriori PDF  $P[\sigma|g]$  as

$$\begin{aligned} P[\sigma|g] &\propto P[g|\sigma] P[\sigma] \\ &\propto P[\sigma] \int [df] P[g|f] P[f|\sigma] \\ &\propto P[\sigma] \frac{\exp(-g^\dagger \tilde{M}^{-1} g)}{\det(\tilde{M})} \end{aligned} \quad (3.7)$$

where

$$\tilde{M} \equiv U \sigma U^\dagger + \nu I \quad (3.8)$$

$P[\sigma]$  provides the prior knowledge about  $\sigma$ , and the exponential and determinant factors provide the "likelihood function" which permits the data  $g$  to constrain  $\sigma$ . This can be written in "Hamiltonian" form as

$$\log P[\sigma|g] = -H + \text{constant} \quad (3.9)$$

where

$$\begin{aligned} H &\equiv H_1 + H_2 + H_3 \\ H_1 &= \text{prior knowledge Hamiltonian} \\ H_2 &= g^\dagger [U \sigma U^\dagger + \nu I]^{-1} g \\ H_3 &= \log \det [U \sigma U^\dagger + \nu I] \end{aligned} \quad (3.10)$$

For small perturbations of  $\sigma$  about some reference we may evaluate the ensuing change to  $P[\sigma|g]$ , and after some algebra we obtain

$$\frac{\partial \log \{P[\sigma|g]\}}{\partial \log \sigma_i} = -\frac{\partial H_1}{\partial \log \sigma_i} + F[f_0] \quad (3.11)$$

The first term expresses the effect of the prior knowledge, and the second the effect of the data. It is a non-trivial result that the data term is a known function only of the MAP reconstruction  $f_0$  which we obtained earlier: in fact this remains true to all orders of differentiation with respect to  $\sigma$ .

These results form the basis of a Markov random field approach to modelling and reconstructing targets from coherent radar images. Thus we must define a prior "target" Hamiltonian  $H_1$  which specifies the typical characteristics of targets in a simple Markovian way, and then the total Hamiltonian can be maximised by using a gradient ascent algorithm which involves the MAP reconstruction  $f_0$  at each stage. It is probably necessary to use an "annealing schedule" in which the effects of  $H_1$ ,  $H_2$  and  $H_3$  are introduced progressively to constrain the form of  $\sigma$ . This method extends the simulated annealing image segmentation methods to cases where the effect of correlations produced by the imaging system (i.e. the point spread function) can be partially deconvolved.

The need to compute  $f_0$  at each stage can be avoided by introducing "dummy variables" into the a posteriori PDF. Thus

$$\begin{aligned} P[\sigma, f|g] &\propto P[g|\sigma, f] P[\sigma, f] \\ &\propto P[g|f] P[f|\sigma] P[\sigma] \end{aligned} \quad (3.12)$$

$$\log P[\sigma, f|g] = -H' + \text{constant} \quad (3.13)$$

where

$$\begin{aligned}
 H' &\equiv H_1' + H_2' + H_3' \\
 H_1' &= H_1 = \text{prior knowledge Hamiltonian (see Equation 3.10)} \\
 H_2' &= f^\dagger \sigma^{-1} f \\
 H_3' &= \nu^{-1} |g - U f|^2
 \end{aligned} \tag{3.14}$$

$H'$  has a far simpler structure than  $H$ , and so a Monte Carlo procedure may easily be used to sample from  $P[\sigma, f|g]$  and suitable averages may be taken in order to extract information about  $\sigma$ . Indeed if the range of the imaging operator  $U$  is finite then  $H'$  is explicitly Markovian in both  $\sigma$  and  $f$ , and so the Monte Carlo procedure may be efficiently implemented on a parallel architecture.

#### IV. CONCLUSIONS

Markov random field models have been introduced for both clutter and targets by appealing to the underlying physics of the image formation process. The advantage of our approach to clutter modelling is that it is adaptive to the clutter statistics, and provides a rigorous means of formulating Bayesian decisions about the presence of targets in the clutter. The unification between target and clutter analysis whereby both are conducted within a Markov random field framework is particularly satisfying, and paves the way for a more tightly knit global

modelling scheme.

The way ahead lies in the development of hierarchical clutter modelling techniques to deal with those clutter types which are not easily described by a small set of local statistics: this leads directly to clutter models which are analogous to neural network models with “hidden layers”, which also map very nicely onto parallel architectures! We also believe that it is highly desirable to remove as much of the stochastic “rattling about” which is needed by the Monte Carlo method as possible, because it can be computationally intensive (even on a parallel processor).

The first of these requirements is currently being addressed at the hardware level at RSRE, where a special purpose processor for rapidly evaluating arbitrary local functions of an image is under construction. The second of these requirements is much more difficult to address, and it is closely related to an important part of the solution to the first requirement, namely: what hierarchy of what statistics is needed for image analysis? This problem is currently the subject of intensive theoretical work.

- [1] D H Ackley, G E Hinton, and T J Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Science **9** (1985), no. 2, 147–169.
- [2] E T Jaynes, *Prior probabilities*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 3, 227–241.
- [3] R Kindermann and J L Snell, *Markov random fields and their applications*, Contemporary Mathematics, vol. 1, American Mathematical Society, Providence, 1980.
- [4] S P Luttrell, *Markov random fields: a strategy for clutter modelling*, Proceedings of AGARD conference on scattering and propagation in random media (Rome), AGARD, 1987, pp. 7.1–7.8.
- [5] ———, *The use of Markov random field models in sampling scheme design*, Proceedings of SPIE international symposium on inverse problems in optics (New York), SPIE, 1987.
- [6] ———, *The use of Markov random field models to derive sampling schemes for inverse texture problems*, Inverse Problems **3** (1987), no. 2, 289–300.
- [7] S P Luttrell and C J Oliver, *Prior knowledge in synthetic aperture radar processing*, Journal of Physics D: Applied Physics **19** (1986), no. 3, 333–356.

# Designing Markov Random Field Structures for Clutter Modelling \*

S. P. Luttrell  
Royal Signals and Radar Establishment, UK

## I. INTRODUCTION

A thorough understanding of clutter statistics is a prerequisite for the successful analysis of radar images. Usually very simple statistics such as moments and correlation properties are used, perhaps based on an underlying physical model of the scattering and imaging process.

In this paper we use the maximum entropy method to reconstruct clutter probability density functions (PDF) from observed statistical properties; this leads to representations of clutter in terms of Markov random fields (MRF). Furthermore we show how the set of statistics which is used for each clutter type may be optimised in order to yield a more compact probabilistic model.

The principal advantage of our results is that MRF clutter models may be mapped directly onto parallel image processing hardware, and they provide a rigorous framework for Bayesian decision making concerning the presence of objects embedded in clutter. Image segmentation is another very useful application of these MRF models.

## II. ESTIMATING CLUTTER STATISTICS

For convenience we shall denote the sample values which comprise an image by the vector  $\mathbf{T}$ . The image dimensionality (one or two usually) is not fundamentally important for our purposes, so we shall restrict ourselves to a one dimensional notation. For an  $N$  pixel image  $\mathbf{T}$  is given explicitly by

$$\mathbf{T} \equiv (T_1, T_2, \dots, T_N) \quad (2.1)$$

where each component  $T_k$  ( $k = 1, \dots, N$ ) is the sample value of pixel  $k$ . We shall be concerned with scalar valued  $T_k$  (such as the moduli of complex pixel values obtained from a coherent radar), but the generalisation to vector valued pixels is straightforward.

If we consider the ensemble of all images  $\mathbf{T}$  which belong to a particular class (e.g. all images of woodland), then the associated PDF  $P[\mathbf{T}]$  summarises our state of knowledge about  $\mathbf{T}$  given only that we know that it is a member of the ensemble. A different such PDF is obtained for each type of ensemble which is defined, and

the set of all PDFs of interest forms a catalogue of information about the ensembles of interest. If we denote a statistic, or sampling function, by  $S[\mathbf{T}]$  then its average value  $\langle S \rangle$  measured over the ensemble is given by

$$\langle S \rangle \equiv \int [d\mathbf{T}] P[\mathbf{T}] S[\mathbf{T}] \quad (2.2)$$

Ignoring edge effects let us define a translation operator  $L$  with the property

$$L T_k = T_{k+1} \quad (2.3)$$

then a necessary and sufficient condition for the image statistics to be homogeneous is

$$P[L^m \mathbf{T}] = P[\mathbf{T}] \quad m = 0, \pm 1, \pm 2, \dots \quad (2.4)$$

so for homogeneous clutter we shall restrict our attention to statistics of the form

$$S[\mathbf{T}] \equiv \frac{1}{N} \sum_m s[L^m \mathbf{T}] \quad (2.5)$$

where  $s[\mathbf{T}]$  is a function which depends on a (localised) subset of the components of  $\mathbf{T}$ , and where the summation is over all translations. An immediate corollary of this definition of  $S[\mathbf{T}]$  is that

$$S[L^m \mathbf{T}] = S[\mathbf{T}] \quad (2.6)$$

whether or not  $P[\mathbf{T}]$  describes homogeneous clutter, so such a global statistic characterises only the homogeneous part of  $P[\mathbf{T}]$  and retains no information about inhomogeneities. For convenience we shall henceforth use the symbol  $L$  to denote a generic translation, so that  $L^m \rightarrow L$  and summation over  $m$  is replaced by summation over  $L$ .

It is well known, Oberhettinger [5], that the average values of the set of all  $n$ -point moments ( $n = 1, 2, \dots$ ) of the  $T_k$  completely characterise  $P[\mathbf{T}]$  whatever the PDF, however in practice only a few moments are measured because of estimation errors which inevitably occur and because of mathematical difficulties which arise when manipulating high order moments. Typically for synthetic aperture radar images of woodland the normalised second moment and the autocorrelation function provide much useful information about  $P[\mathbf{T}]$ , as Oliver [6] has demonstrated in his clutter simulation studies. In [6] an underlying physical model is used to deduce the values of otherwise unmeasured moments, so a mixture of physics and empirical measurement permits a successful characterisation of  $P[\mathbf{T}]$  without resorting to exhaustive measurement

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This appeared in Proceedings of the IEE International Conference RADAR87, pp. 222-226, 1987. © Controller, Her Majesty's Stationery Office, London, 1987.

of all the moments. However the validity of such underlying physical models has frequently been questioned despite their obvious success in certain applications, which has led to other approaches to the statistical analysis of radar clutter, each with its own proponents.

We shall not propose any particular detailed clutter model in this paper, rather we shall deduce what we can about the clutter structure from a set of measured statistics in an attempt to assess the useful information content of the measurements. This approach not only leads to a means of refining the set of statistics, but also permits an explicit probabilistic clutter model to be constructed.

### III. RECOVERING THE CLUTTER PDF

We shall now develop suitable theoretical machinery for characterising clutter using only the prior knowledge that clutter is homogeneous. In general we measure a low dimensional vector of statistics  $\mathbf{S}[\mathbf{T}]$  which summarises the structure of  $\mathbf{T}$  by performing a many to one mapping  $\mathbf{T} \rightarrow \mathbf{S}$ ; this is a form of data compression. Because the mapping is not invertible  $\mathbf{T}$  cannot be completely recovered from knowledge of  $\mathbf{S}$  alone, rather only the feasible set of  $\mathbf{T}$  which might have caused  $\mathbf{S}$  may be obtained. We may express the PDF over statistics  $P[\mathbf{S}]$  in terms of  $P[\mathbf{T}]$  by

$$P[\mathbf{S}] = \int [d\mathbf{T}] P[\mathbf{T}] \delta[\mathbf{S} - \mathbf{S}[\mathbf{T}]] \quad (3.1)$$

where the delta function restricts the integration to the feasible set of  $\mathbf{T}$ . A special case of interest occurs if  $P[\mathbf{T}]$  is a function only of  $\mathbf{S}[\mathbf{T}]$ , because then  $P[\mathbf{S}]$  reduces to

$$P[\mathbf{S}] = J[\mathbf{S}] P[\mathbf{T}] \quad (3.2)$$

where  $J[\mathbf{S}]$  is the Jacobian of the mapping  $\mathbf{T} \rightarrow \mathbf{S}$  (i.e. the number of  $\mathbf{T}$  which map to  $\mathbf{S}$ ). Therefore if we measure  $P[\mathbf{S}]$  and calculate  $J[\mathbf{S}]$  from the known  $\mathbf{S}[\mathbf{T}]$  we may use Equation 3.2 to calculate  $P[\mathbf{T}]$ . Thus  $\mathbf{S}$  comprises a sufficient set of statistics from which all knowledge of the ensemble of  $\mathbf{T}$  as described by  $P[\mathbf{T}]$  may be recovered.

We have not described how to obtain a sufficient set of statistics because their form depends on  $P[\mathbf{T}]$ . In practice we usually use a predetermined set of statistics  $S[\mathbf{T}]$  which are of the form given in Equation 2.5 and we shall denote their observed value by  $\mathbf{S}$ . We shall assume that the image size is sufficiently large that

$$\langle \mathbf{S} \rangle \simeq \mathbf{S} \quad (3.3)$$

to a good approximation; strictly this requires that the image is ergodic. This approximation corresponds to assuming that most of the mass of  $P[\mathbf{S}]$  is contained within a sharply defined peak, and so we ignore the detailed shape of  $P[\mathbf{S}]$  when we use the measured  $\mathbf{S}$  to describe all of its properties.

Within this approximation there are many permissible  $P[\mathbf{T}]$  so we must define a scheme for selecting a single

PDF to represent the clutter structure which is measured by  $\mathbf{S}[\mathbf{T}]$ . We shall appeal to the elegant information theoretic arguments of Jaynes [2] in order to recover a PDF from measurements of expectation values alone. Thus if we are provided with a set of constraints (i.e. expectation values) on the form of a PDF, and we have no further information about the PDF, we are compelled to select the PDF with the maximum entropy from the feasible set of PDFs. This prescription ensures that only that structure which is positively demanded by the constraints is present in the PDF, and that the PDF is maximally non-committal with respect to information which has yet to be measured. An alternative, but equivalent, way of expressing this is to note that the maximum entropy PDF permits the constraints to be satisfied by the maximum number of possible distinct realisations, so that the least constraint is placed by the PDF upon exactly which particular realisation occurs in practice.

The maximum entropy problem is formulated by defining the entropy  $H$  of  $P[\mathbf{T}]$  by

$$H \equiv - \int [d\mathbf{T}] P[\mathbf{T}] \log P[\mathbf{T}] \quad (3.4)$$

which is maximised subject to the constraint

$$\mathbf{S} = \int [d\mathbf{T}] P[\mathbf{T}] \mathbf{S}[\mathbf{T}] \quad (3.5)$$

Introducing Lagrange multipliers  $\boldsymbol{\lambda}$ , leads after a short calculation to the maximum entropy PDF  $Q[\mathbf{T}]$  in the form

$$Q[\mathbf{T}] = \frac{1}{Z} \exp \{-\boldsymbol{\lambda} \cdot \mathbf{S}[\mathbf{T}]\} \quad (3.6)$$

where  $\boldsymbol{\lambda}$  is chosen to satisfy the constraints and  $Z$  is a normalising factor (the partition function). Furthermore by using the explicit form for  $\mathbf{S}[\mathbf{T}]$  we may write  $Q[\mathbf{T}]$  in the form

$$Q[\mathbf{T}] = \frac{1}{Z} \prod_L \exp \{-\boldsymbol{\lambda}_L \cdot \mathbf{s}[L \mathbf{T}]\} \quad (3.7)$$

which is a product of factors each of which depends only locally on  $\mathbf{T}$  (via the statistics  $\mathbf{s}[\mathbf{T}]$ ). This form of PDF is well known from Markov random field (MRF) theory, Kindermann and Snell [3], and it is called a Gibbs distribution in statistical thermodynamics.

### IV. ESTIMATING THE UNDETERMINED MULTIPLIERS

We shall now describe how to obtain  $\boldsymbol{\lambda}$  by appealing to a simple generalisation of the “Boltzmann machine learning algorithm”, Ackley et al [1], which is used in neural network theory to obtain MRF representations of PDFs. The constraints which we must satisfy are summarised by

$$\langle \mathbf{S} \rangle_Q = \langle \mathbf{S} \rangle_P \quad (4.1)$$

where the subscripts indicate the PDF which is used in the averaging process, and the task is to vary  $\lambda$ , and thus  $Q[\mathbf{T}]$ , in order to satisfy these constraints. We shall introduce a “Lyapunov function”  $G$  which is defined by

$$G \equiv - \int [d\mathbf{T}] P[\mathbf{T}] \log \left\{ \frac{P[\mathbf{T}]}{Q[\mathbf{T}]} \right\} \quad (4.2)$$

where  $G \leq 0$ , and  $G = 0$  if and only if  $P[\mathbf{T}] = Q[\mathbf{T}]$  for all  $\mathbf{T}$ . The gradient of  $G$  with respect to  $\lambda$  is then given by

$$\text{grad}(G) = \langle \mathbf{S} \rangle_Q - \langle \mathbf{S} \rangle_P \quad (4.3)$$

so that the constraint equation now becomes

$$\text{grad}(G) = \mathbf{0} \quad (4.4)$$

The goal is therefore to maximise  $G$  with respect to  $\lambda$  by using the difference  $\langle \mathbf{S} \rangle_Q - \langle \mathbf{S} \rangle_P$  to estimate the direction in which to vary  $\lambda$  at each stage.

$\langle \mathbf{S} \rangle_P$  is estimated from the original homogeneous clutter image, but  $\langle \mathbf{S} \rangle_Q$  is not so easily obtained. For each value of  $\lambda$  we must use a Monte Carlo (MC) simulation to sample from  $Q[\mathbf{T}]$  in order to generate a synthetic homogeneous clutter image from which to estimate  $\langle \mathbf{S} \rangle_Q$ . This procedure is laborious on a conventional serial computer, but a large speed increase can be obtained by using a parallel computer because of the local nature of  $\mathbf{s}[\mathbf{T}]$ .

## V. MARGINAL PDFS

We have shown how an arbitrary set of statistics  $\mathbf{S}[\mathbf{T}]$  can be used to deduce a maximum entropy approximation to  $P[\mathbf{T}]$ , but it involves laborious numerical simulations using MC techniques. We shall now indicate how the calculations can be simplified by choosing more carefully which particular statistics we use in the first place.

If we choose  $\mathbf{S}[\mathbf{T}]$  to be a set of statistics which exhaustively characterise the local properties of  $P[\mathbf{T}]$  then  $Q[\mathbf{T}]$  is much easier to recover, as we would expect. However we usually pay the price for this convenience by having to enlarge the dimensionality of  $\mathbf{S}$ . Such a set of statistics is provided by a localised marginal PDF  $P[\mathbf{t}]$  which is defined by

$$P[\mathbf{t}] \equiv - \int [d\mathbf{T}'] P[\mathbf{T}] \quad (5.1)$$

where  $\mathbf{T}'$  consists of those components of  $\mathbf{T}$  which remain after removing the components  $\mathbf{t}$  which are involved in the marginal PDF. Because we assume that the clutter is homogeneous (and ergodic) we may estimate  $P[\mathbf{t}]$  by performing a spatial average thus

$$\mathbf{H} \equiv \frac{1}{N} \sum_L \mathbf{h}[L \mathbf{t}] \quad (5.2)$$

where  $\mathbf{h}[\mathbf{t}]$  is the “histogramming operation” which maps  $\mathbf{t}$  into a bin of a histogram  $\mathbf{h}$  of all possible states  $\mathbf{t}$ ,

and  $\mathbf{H}$  is the estimate of the histogram which is obtained from the whole image. For convenience we shall write  $\mathbf{H}$  as  $H[\mathbf{t}]$  in order to conform to our notation  $P[\mathbf{t}]$  for marginal PDFs. Note that histogramming can only be performed with a digitised representation of  $\mathbf{T}$ , and so the marginal PDF  $P[\mathbf{t}]$  can never be directly estimated in its analogue form. In the large image approximation

$$P[\mathbf{t}] \simeq H[\mathbf{t}] \quad (5.3)$$

Provided that  $\mathbf{t}$  is chosen carefully knowledge of  $P[\mathbf{t}]$  allows us to recover  $Q[\mathbf{T}]$  by direct construction. To illustrate this point consider an infinite one dimensional homogeneous clutter image given by

$$\mathbf{T} = (\dots, T_{-2}, T_{-1}, T_0, T_1, T_2, \dots) \quad (5.4)$$

together with an estimated marginal PDF

$$P[\mathbf{t}] = P[T_{k-1}, T_k] \quad (5.5)$$

which is invariant under translations  $L$ . Let us attempt to find a first order Markovian approximation  $Q[\mathbf{T}]$  to  $P[\mathbf{T}]$  in the form

$$Q[\mathbf{T}] = \prod_k P[T_k | T_{k-1}] \quad (5.6)$$

However

$$P[T_k | T_{k-1}] = \frac{P[T_{k-1}, T_k]}{\int dT_k P[T_{k-1}, T_k]} \quad (5.7)$$

whence  $Q[\mathbf{T}]$  may be expressed entirely in terms of the estimated marginal PDF in Equation 5.5. Thus we have a consistent Markovian (maximum entropy) PDF which satisfies the required constraints. This is truly a maximum entropy solution because we have assumed only first order Markovianity, and have therefore introduced the least possible structure into  $Q[\mathbf{T}]$  which is required by the constraints.  $Q[\mathbf{T}]$  is explicitly a product over local factors and it can readily be transformed into a Gibbs distribution form to yield

$$Q[\mathbf{T}] = \prod_k \exp \{ \log \{ P[T_k | T_{k-1}] \} \} \quad (5.8)$$

where the argument of the exponential is estimated from the measured histogram (Equation 5.2, Equation 5.3, Equation 5.5 and Equation 5.7).

This one dimensional argument may easily be extended to marginal PDFs which involve more than two components, which leads to higher order Markov models. In two or more dimensions the procedure is more delicate because the simple one-dimensional Markov chain is replaced by the full Markov random field apparatus. However it still proves to be possible to build up the full joint PDF  $P[\mathbf{T}]$  piecemeal by assuming a suitable Markovian structure which may be determined entirely from a marginal PDF. Some of these results have been reported by Luttrell [4], and further results will be published elsewhere.

## VI. BUILDING A SET OF STATISTICS

We have seen how an exhaustive set of local statistics, a marginal PDF, leads to a very simple way of recovering an MRF approximation to  $P[\mathbf{T}]$ . Alternatively we may sacrifice this convenience by attempting to construct a low dimensional set of local statistics in order to keep the dimensionality of  $\boldsymbol{\lambda}$  as small as possible, and thus alleviate the computational burden which is associated with determining  $\boldsymbol{\lambda}$  by using Equation 4.3. Thus we shall present a prescription for building a set of statistics which is specifically designed to extract the structure of  $P[\mathbf{T}]$  efficiently.

The key to this procedure is the expression for  $\text{grad}(G)$ . Suppose we already have an  $m$ -dimensional set of statistics  $\mathbf{S}[\mathbf{T}] = (S_1[\mathbf{T}], \dots, S_m[\mathbf{T}])$  and its associated  $\boldsymbol{\lambda}$  which satisfy  $\text{grad}(G) = \mathbf{0}$ . Then any novel statistic  $S_{m+1}[\mathbf{T}]$ , for which  $\lambda_{m+1} = 0$  currently, may be inserted into the expression for  $\text{grad}(G)$  to estimate the value of  $\frac{\partial G}{\partial \lambda_{m+1}}$ . If this gradient is close to zero then  $S_{m+1}$  is redundant in the sense that it does not measure any structure in  $P[\mathbf{T}]$  which has not already been measured by  $\mathbf{S}[\mathbf{T}]$ , and so should be discarded as a worthless additional measurement to make. However, if the gradient is large then  $S_{m+1}$  is measuring novel structure in  $P[\mathbf{T}]$  and so it should be considered as a viable candidate for inclusion in the set of statistics. This process may be iterated to build up an economical set of statistics by rejecting any statistics which do not measure novel structure in  $P[\mathbf{T}]$ .

The procedure which we outlined above requires  $\boldsymbol{\lambda}$  to be recalculated after each iteration in order to update  $Q[\mathbf{T}]$  when the dimensionality of  $\mathbf{S}[\mathbf{T}]$  is increased by one. In fact we need only generate a non-committal realisation  $\mathbf{T}$  which satisfies the constraints  $\langle \mathbf{S} \rangle_Q = \langle \mathbf{S} \rangle_P$  in order to guarantee that  $\mathbf{T}$  is an unbiased sample from  $Q[\mathbf{T}]$ . To achieve this we use an optimisation by simulated annealing (OSA) procedure. Thus define the PDF

$$Q[\mathbf{T}, \mathbf{a}] = \exp \left\{ -\frac{|\langle \mathbf{S} \rangle_Q - \langle \mathbf{S} \rangle_P|^2}{2a^2} \right\} \quad (6.1)$$

where the quantities  $\langle \dots \rangle$  are understood to be spatial averages and the parameter  $a$  plays the role of a temperature in OSA. When  $a > 0$  the PDF  $Q[\mathbf{T}, \mathbf{a}]$  corresponds to  $Q[\mathbf{T}]$  with the required constraints only weakly imposed, and as  $a \rightarrow 0$  the constraints are imposed more and more stringently so that

$$Q[\mathbf{T}] \simeq \lim_{a \rightarrow 0} Q[\mathbf{T}, \mathbf{a}] \quad (6.2)$$

We may use MC to sample from  $Q[\mathbf{T}, \mathbf{a}]$  ( $a > 0$ ) in order to generate a realisation  $\mathbf{T}$  which approximately satisfies the constraints, then by gradually reducing  $a$  towards zero in an ‘annealing schedule’ we may obtain a realisation of  $\mathbf{T}$  which obeys the constraints exactly. This form for  $Q[\mathbf{T}, \mathbf{a}]$  forces  $\mathbf{T}$  to satisfy only the global

constraint  $\langle \mathbf{S} \rangle_Q = \langle \mathbf{S} \rangle_P$  as  $a \rightarrow 0$ , so the realisation  $\mathbf{T}$  which is obtained in the limit is otherwise non-committal, as required. The gradient  $\frac{\partial G}{\partial \lambda_{m+1}}$  (for  $\lambda_{m+1} = 0$ ) may thus be evaluated for a novel statistic  $S_{m+1}[\mathbf{T}]$  without determining  $\boldsymbol{\lambda}$  itself.

When we are satisfied that a good set of statistics has been constructed because, for instance, we cannot find any novel statistic which has a significant gradient, we may determine  $\boldsymbol{\lambda}$  in order to recover the maximum entropy (Markov random field) form of  $Q[\mathbf{T}]$ . This will generally be less computationally intensive than if the set of statistics had been selected arbitrarily, because our systematic construction procedure will lead to a set of statistics with reduced dimensionality.

## VII. APPLICATION OF THEORETICAL RESULTS

Whatever means we use to obtain  $Q[\mathbf{T}]$  it has the form of a product of local factors, and so it may be written as

$$Q[\mathbf{T}] = \prod_L q[L \mathbf{t}] \quad (7.1)$$

where  $\mathbf{t}$  is a suitably chosen marginal set of pixels, and  $q[\mathbf{t}]$  may be obtained directly from either of the product forms in Equation 3.7 and Equation 5.6. For a particular clutter image  $\mathbf{T}$  the  $q[L \mathbf{t}]$  may also be displayed as an image if so desired, because as  $L$  ranges over all possible translations the function  $q[L \mathbf{t}]$  is separately defined at every pixel position; we shall call such an image a probability image.  $Q[\mathbf{T}]$  is then the product of the pixel values of the probability image. Loosely speaking bright areas in the probability image correspond to parts of the clutter which are typical of the whole clutter image, and dark areas are atypical. This property may be used to determine the presence and location of inhomogeneities (or objects embedded in clutter) in the original image by setting a threshold below which the value of  $q[\mathbf{t}]$  is considered to be so low that the presence of an inhomogeneity is suspected.

A more rigorous Bayesian decision may be made if the PDF which describes the type of objects being sought in the clutter is available. A Bayesian framework may then be constructed in which the relative probability of object + clutter versus clutter alone is calculated. For objects which typically occupy only one pixel and for which their one pixel PDF is a sufficient description it is simple to construct the Bayesian framework. Thus assume that the object PDF is given by  $U(\mathbf{T})$ , then the likelihood ratio  $R$  that an object is present at pixel  $k$  is given by

$$R = \frac{U(T_k) \int dT_k Q[\mathbf{T}]}{Q[\mathbf{T}]} \quad (7.2)$$

Introducing the product form of Equation 7.1 leads to

a lot of cancellation which finally yields

$$R = \frac{U(T_k) \int dT_k \prod_L q[L|t]}{\prod_L q[L|t]} \quad (7.3)$$

where the product over L involves only those  $q[L|t]$  which depend on  $T_k$ ; a small number of factors in general. This expression for R is particularly simple to evaluate when  $q[t]$  is determined from marginal PDFs, because all of the required operations (products, ratios and sums) may be performed on the associated histogram with ease.

This procedure for obtaining the likelihood ratio R will yield a result whose accuracy is limited by the range of the local statistics, but which otherwise is a completely flexible, non-committal, and data driven scheme. In a sense it is a generalisation of the constant false alarm rate (CFAR) method of target detection to incorporate a more sophisticated adaptation algorithm to adjust to the local clutter statistics.

### VIII. CONCLUSIONS

We have shown how for a given set of statistics an MRF probabilistic clutter model may be recovered by using the

maximum entropy method. This PDF is maximally non-committal with respect to information which has yet to be measured, and so it provides an unbiased characterisation of the information which is measured by a set of statistics. In particular we have shown that by making an appropriate choice of statistics we may facilitate the recovery of the MRF model to the extent that the calculations reduce to simple histogram manipulations. Alternatively we have shown how the set of statistics might be constructed in such a way that an economical MRF model is obtained, although the calculations here are not so simple.

Whatever the technique which is used to obtain the MRF model it has an ideal form for implementation on parallel image processing hardware, and simple Bayesian decision making algorithms may easily be constructed which take advantage of the MRF form of the clutter model. Provided that the range of the local statistics which is considered is large enough, the MRF model will adapt to and faithfully represent the local statistical properties of homogeneous clutter, and so image processing algorithms which amount to a generalisation of CFAR may be constructed.

- 
- [1] D H Ackley, G E Hinton, and T J Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Science **9** (1985), no. 2, 147–169.
  - [2] E T Jaynes, *Prior probabilities*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 3, 227–241.
  - [3] R Kindermann and J L Snell, *Markov random fields and their applications*, Contemporary Mathematics, vol. 1, American Mathematical Society, Providence, 1980.
  - [4] S P Luttrell, *The use of Markov random field models to derive sampling schemes for inverse texture problems*, Inverse Problems **3** (1987), no. 2, 289–300.
  - [5] F Oberhettinger, *Fourier transforms of distributions and their inverses*, Academic Press, New York, 1973.
  - [6] C J Oliver, *The interpretation and simulation of clutter textures in coherent images*, Inverse Problems **2** (1986), no. 4, 481–518.



# Self-Organising Multilayer Topographic Mappings \*

S P Luttrell

Royal Signals and Radar Establishment, St Andrews Road, Malvern, WORCS, WR14 3PS, UK

We use a minimum distortion measure argument to derive the need for topographic mappings in unsupervised multilayer networks, and we perform some numerical experiments to demonstrate the power of multilayer topographic mappings.

## I. INTRODUCTION

A fundamental problem in image analysis is feature space design, which is difficult if we have little a priori knowledge of the image structure. A neural network approach to feature space design could be either supervised or unsupervised. As an example of the supervised approach, a multilayer perceptron could in principle be used to cluster images in each class onto a single output unit response. On the other hand an unsupervised network would merely process each image into a particular pattern of responses with no particular interpretation (such as class membership) being made.

We shall adopt the unsupervised approach because it permits a factorisation of the overall problem of classification into two stages: a preliminary organisation of the image (using the unsupervised network), followed by some post processing to classify the image (using some as yet unspecified technique). We note the similarity of this factorisation to that employed by the human visual system. Early visual processing extracts a few simple features such as blobs and edges, which are then used as input to the higher visual processing levels.

In Section II we derive the need for topographic mappings in unsupervised multilayer networks. In Section III we propose a modification of the topographic mapping learning algorithm [1] which increases its rate of its convergence. In Section IV we conduct some numerical experiments to demonstrate the principles developed in Section II and Section III.

## II. MULTILAYER UNSUPERVISED NEURAL NETWORKS

The task which faces us can be defined succinctly. Defining the input pattern as  $\mathbf{x}_{in}$  we must find a mapping  $T : \mathbf{x}_{in} \rightarrow \mathbf{x}_{out}$  which produces an output pattern  $\mathbf{x}_{out}$  which satisfies three criteria

- (i)  $\dim(\mathbf{x}_{out}) \ll \dim(\mathbf{x}_{in})$
- (ii)  $d(\mathbf{x}_{in}, \mathbf{x}'_{in})$  must be as small as possible
- (iii)  $T$  is constructed out of elementary computations

where  $\dim(\mathbf{x})$  denotes the dimension of  $\mathbf{x}$ , and  $d(\mathbf{x}_{in}, \mathbf{x}'_{in})$  denote a distortion measure which is used to assess the quality of reconstruction  $\mathbf{x}'_{in}$  from  $\mathbf{x}_{out}$ . Criteria (i) and (ii) are in conflict because dimension reduction unavoidably incurs loss of information. Criterion (iii) is imposed because the set of all possible  $T$  is too large to search, whereas a factorised  $T$  offers the possibility of deriving a simple learning algorithm.

Most neural network designs have a high degree of connectivity between the units which makes them expensive to simulate. Furthermore it can be difficult to interpret what information processing is being performed if the network does not spontaneously simplify its connections during training. We shall restrict our attention to feed-forward multilayer networks where the units within each layer have no interconnections, and the interconnections between layers are limited in range.

We shall represent the function of unit  $j$  in layer  $L$  as  $y = T_j^L(\mathbf{x})$  where  $\mathbf{x}$  is the set of layer  $L - 1$  outputs which the unit sees, and  $y$  takes values in some suitably chosen set. The overall network mapping  $T$  may then be represented symbolically as a product

$$T = \dots \left( \times_k T_k^2 \right) \dots \left( \times_j T_j^1 \right) \quad (2.1)$$

$T_j^L(\mathbf{x})$  must satisfy stringent constraints if it is to preserve a lot of information about  $\mathbf{x}$ . Consider the two step process  $\mathbf{x} \rightarrow y \rightarrow \mathbf{x}'$ , where  $y = T_j^L(\mathbf{x})$  and  $\mathbf{x}'$  is chosen at random from set of  $\mathbf{x}$  which satisfy  $y = T_j^L(\mathbf{x}')$ . The optimum  $T_j^L$  should minimise the average of the distortion measure  $d(\mathbf{x}, \mathbf{x}')$ . We shall assume that  $d(\mathbf{x}, \mathbf{x}')$  is localised in the sense that the set of solutions  $\mathbf{x}'$  of  $y = T_j^L(\mathbf{x}')$  must form a small (solid) volume rather than several disjoint pieces. A further constraint on  $T_j^L(\mathbf{x})$  arises out of the imposed distortion measure because each layer processes the output of a previous layer. Thus  $T_j^L(\mathbf{x})$  has a distortion measure imposed on its output (i.e. the input of layer  $L + 1$ ). A simple example will suffice to illustrate this point. Define a three layer network as

$$\begin{aligned} y_1 &= T_1^1(x_1, x_2) \\ y_2 &= T_2^1(x_3, x_4) \\ z &= T_1^2(y_1, y_2) \end{aligned} \quad (2.2)$$

The imposition of a distortion measure  $d(\mathbf{y}, \mathbf{y}')$  between  $\mathbf{y}$  and  $\mathbf{y}'$  (where  $\mathbf{y}'$  is reconstructed from  $\mathbf{z}$ ) implies that small changes  $\mathbf{y} \rightarrow \mathbf{y} + \Delta$  are deemed not to change

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 11, 2019.

This appeared in Proceedings of the 2<sup>nd</sup> IEEE International Conference on Neural Networks, San Diego, Vol. 1, 93-100, 1988.

the meaning of  $\mathbf{y}$  significantly, in which case the set of solutions  $\mathbf{x}'$  and  $\mathbf{x}''$  of  $y_1 = T_1^1(\mathbf{x}')$  and  $y_2 = T_2^1(\mathbf{x}'')$  must also be insensitive to  $\mathbf{y} \rightarrow \mathbf{y} + \Delta$ . This constraint on the invertibility of the functions  $T_1^1(x_1, x_2)$  and  $T_2^1(x_3, x_4)$  forces them to be topographic mappings. Note that we do not yet need to introduce a distortion measure for the final output  $\mathbf{z}$ , so strictly there is no need for  $T_1^2(y_1, y_2)$  to be topographic.

We have argued that topographic mappings are necessary in multilayer unsupervised neural networks; they follow from our imposition of a distortion measure. There is no such need to introduce a distortion measure in a supervised neural network, because such networks are committed to producing a particular input-output mapping without necessarily preserving much information about  $\mathbf{x}_{in}$  itself. This does not deny the possibility that a supervised network might fortuitously retain enough information in  $\mathbf{x}_{out}$  for  $\mathbf{x}_{in}$  to be reconstructed with little distortion.

We should point out that factorising the  $T$  into a product of simple operations (see Equation 2.1) can make a fast look-up table implementation of a trained network feasible. An operation  $\mathbf{y} = T_j^L(\mathbf{x})$  then requires a  $2^r \times s$  bits look-up table, where  $r$  is the number of bits required to represent  $\mathbf{x}$  and  $s$  the number required for  $\mathbf{y}$ .  $r = 20$  and  $s = 8$  (1 MByte) is feasible using current memory technology, but the dimension of  $\mathbf{x}$  is severely limited for grey level images because it contributes exponentially to the look-up table size (via the  $2^r$  factor).

### III. A LEARNING ALGORITHM

There exists an unsupervised training procedure [1] which leads to topographic mappings, which can also be shown to minimise a logarithmic distortion measure. The algorithm could be applied direct to our multilayer network by training  $T_j^1$  then  $T_j^2$ , etc. Each  $T_j^L$  then becomes a vector quantiser which is represented by a “Peano curve” (PC) of  $n_c$  code vectors  $\mathbf{v}_{j,k}^L$  ( $k = 1, 2, \dots, n_c$ ). The output  $\mathbf{y} = T_j^L(\mathbf{x})$  then takes an integer value in the interval  $[1, n_c]$ .

We find that the training process can be dominated by the time it takes the PC to untie itself from the knotted configuration in which it starts. The original algorithm uses an update prescription in which the set of  $\mathbf{v}_{j,k}^L$  which is updated is monotonically decreased in size from a large initial value. A large update set size  $u_c$  causes the  $\mathbf{v}_{j,k}^L$  to be highly correlated which allows them to concentrate on large scale structure only, and the progressive reduction in  $u_c$  allows the effect of ever smaller scale structure to be felt. However,  $n_c$  is held constant, so that in the early stages of the training process the effective number of degrees of freedom which are available to the PC is very much less than  $u_c$ . A large part of the computation is thus spent in creating correlated motions of the  $\mathbf{v}_{j,k}^L$ . This problem is analogous to that encountered in the numerical simulation of physical systems in which long

correlation lengths are present.

We therefore propose that a small fixed  $u_c$  should be used together with a monotonically increasing  $n_c$ . A suitable update prescription ( $u_c = 3$ ) for the  $\mathbf{v}_{j,k}^L$  which represent  $T_j^L$  in the mapping  $\mathbf{y} = T_j^L(\mathbf{x})$  is

$$\begin{aligned}\mathbf{v}_{j,k'}^L &\longrightarrow \mathbf{v}_{j,k'}^L + \epsilon_0 (\mathbf{x} - \mathbf{v}_{j,k'}^L) \\ \mathbf{v}_{j,k' \pm 1}^L &\longrightarrow \mathbf{v}_{j,k' \pm 1}^L + \epsilon_1 (\mathbf{x} - \mathbf{v}_{j,k' \pm 1}^L)\end{aligned}\quad (3.1)$$

where  $k'$  minimises  $\|\mathbf{x} - \mathbf{v}_{j,k'}^L\|$ . The  $\epsilon_0$  parameter determines the update rate, and the ratio ( $0 \leq \frac{\epsilon_1}{\epsilon_0} < 1$ ) determines the stiffness of the Peano curve. A non-zero stiffness causes the mapping to be topographic. Note that the endpoints of the PC have a one-sided neighbourhood except where we choose to use a circular wraparound prescription.

If we set  $n_c = 2$  initially and then train  $\mathbf{v}_{j,1}^L$  and  $\mathbf{v}_{j,2}^L$  according to Equation 3.1 we obtain a very crude vector quantisation. We can increase  $n_c$  by code vector insertion in which extra code vectors  $\mathbf{v}_{j,k}^L$  are generated by using the prescription

$$\mathbf{v}_{j,k}^L = \frac{1}{2} (\mathbf{v}_{j,k}^L + \mathbf{v}_{j,k+1}^L) \quad k = 1, 2, \dots, n_c - 1 \quad (3.2)$$

The number of code vectors is thereby increased to  $2n_c - 1$ . Ever smaller scale structure is felt as  $n_c$  is increased until finally the required finesse is obtained. In both the original algorithm and in ours the ratio  $\frac{u_c}{n_c}$  is monotonically decreased.

### IV. SIMPLE IMAGE PROCESSING: THE LINE DETECTOR

We have shown elsewhere how multilayer topographic mappings can be used to compress images with little distortion in the reconstruction [2]. Here we shall demonstrate the ability of such a network to form useful feature detectors by training it recognise lines. Throughout this section the training set consists of  $3 \times 3$  pixel images (8 bits per pixel) containing a line at one of eight orientations passing through the centre pixel. The training set is shown ( $a = 255$ ,  $b = 127$ ) in the table.

$3 \times 3$	000	00b	00a	0bb	0a0	bb0	a00	b00
training	aaa	bab	0a0	0a0	0a0	0a0	bab	
images	000	b00	a00	bb0	0a0	0bb	00a	00b

We do not vary the grey level of these images because we seek only to demonstrate the principle of multilayer topographic mapping formation. All the results which we present here have been obtained with  $\epsilon_0 = 0.25$  and  $\frac{\epsilon_1}{\epsilon_0} = \frac{1}{3}$ , and we have presented  $30n_c$  training images between each episode of code vector insertion. The PC stiffness is large enough to prevent wild fluctuations in the PC configuration, yet small enough to prevent the PC collapsing towards the centre of gravity of the distribution of input patterns. We have not found it necessary

to decrease  $\epsilon_0$  monotonically during training, because the fluctuations which thereby remain prove to be insignificant. We also train the  $\mathbf{v}_{j,k}^L$  (for all  $j$  and  $k$ ) before proceeding to the  $\mathbf{v}_{j,k'}^{L+1}$ , although this is not a necessary constraint.

We shall use two different network structures:

(N1) A two layer network with a single output unit which sees the whole  $3 \times 3$  input image.

(N2) A three layer network in which the hidden layer has four units and there is one output unit. Each hidden unit sees a different  $2 \times 2$  corner of the  $3 \times 3$  input image.

N1 is an example of a direct input-output mapping with input dimension 9, whereas N2 is a factorised mapping with input dimension 4 for all units. N1 is highly inconvenient to realise as a look-up table, whereas N2 has a compact look-up table realisation.

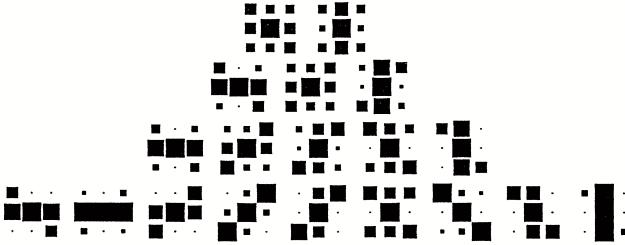


Figure 1: N1 trained without wraparound.

In Figure 1 we present a typical set of  $\mathbf{v}_{1,k}^1$  which is learnt by N1. The rows of Figure 1 show the  $n_c = 2, 3, 5, 9$  results respectively (after  $30n_c$  updates in each case) when the index  $k$  does not wrap around ( $k = 1, 2, \dots, n_c$  left to right in each row). The  $\mathbf{v}_{1,k}^1$  clearly resemble the training images especially for  $n_c = 9$ . Note how the orientation of the line tends to change systematically as  $k$  is varied for a given  $n_c$  (this is the topographic property). However not all is perfect because in the last row of Figure 1 ( $n_c = 9$ ) this trend changes direction at  $k = 6$ , so that  $k$  is not a monotonic function of orientation as we would prefer. However Figure 1 is a typical example of a worst case result when N1 is trained in this way, and a monotonic trend is usually recovered if we train more carefully (i.e. smaller  $\epsilon_0$  and a larger number of updates between code vector insertions).

A simple way of enforcing a monotonic trend is to join the ends of the PC to make a ‘‘Peano circle’’. This wraparound prescription affects the update Equation 3.1 only. The results are shown in Figure 2 using the same training parameters.

The quality of the  $n_c = 9$  result should be compared with that in Figure 1. The training set is parameterised by essentially one angular degree of freedom, so its topology is that of a circle ( $S^1$ ). Similarly, the topology of the output  $y = T_1^1(\mathbf{x}_{in})$  is also  $S^1$  because we have introduced wraparound, so the learning algorithm now finds it very easy to obtain a good topographic mapping.

We now repeat the above two experiments with N2. The hidden units have mappings  $T_j^1$  ( $j = 1, 2, 3, 4$ ) which

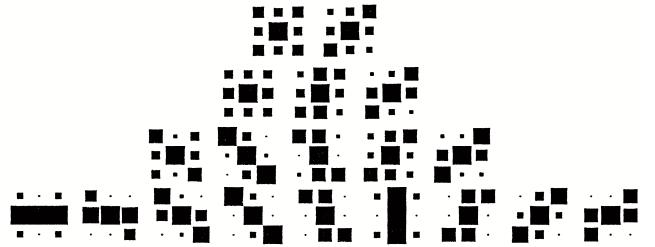


Figure 2: N1 trained with wraparound.

feed their outputs to a single output unit via a mapping  $T_1^2$ . The topologies of the  $2 \times 2$  subimages are related by permutations of the pixels, so we shall derive  $T_j^1$  ( $j = 2, 3, 4$ ) from  $T_1^1$  by a suitable pixel permutations.

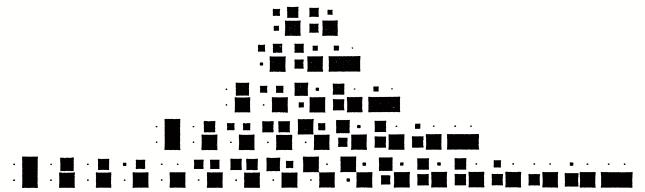


Figure 3.1

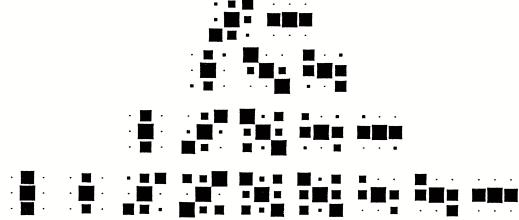


Figure 3.2

Figure 3: Figure 3.1: Hidden layer of N2 trained without wraparound. Figure 3.2: Output layer of N2 trained without wraparound.

In Figure 3.1 we present a typical set of  $\mathbf{v}_{1,k}^1$  ( $k = 1, 2, \dots, n_c$ ) for  $n_c = 2, 3, 5, 9, 17$ , corresponding to the top left hand  $2 \times 2$  subimage of the  $3 \times 3$  input image. The  $\mathbf{v}_{1,k}^1$  show a close resemblance to appropriate  $2 \times 2$  subimages chosen from the training set. In the last row of Figure 3.1 ( $n_c = 1$ ) we see that the monotonic trend in orientation is broken between  $k = 5$  and  $k = 6$ . We then trained the  $\mathbf{v}_{1,k}^2$  on the outputs on the  $n_c = 17$  version of the hidden layer to produce the  $\mathbf{v}_{1,k}^2$  ( $k = 1, 2, \dots, n_c$ ) for  $n_c = 2, 3, 5, 9$  shown in Figure 3.2. Note that  $n_c$  now refers to the number code vectors  $\mathbf{v}_{1,k}^2$  (not  $\mathbf{v}_{1,k}^1$ ). The last row of Figure 3.2 ( $n_c = 9$ ) has a break in the monotonic trend as before.

We now repeat the above experiment with wraparound of all Peano curves. We justify this as follows. Each  $2 \times 2$  subimage of the  $3 \times 3$  training image produces a training set with an  $S^1$  topology, so the topology associated with each unit in the hidden layer may be set to  $S^1$ : the 4

hidden units then define a 4-torus ( $S^1 \times S^1 \times S^1 \times S^1$ ). We also give the output unit an  $S^1$  topology because overall the training set of  $3 \times 3$  images has an  $S^1$  topology.

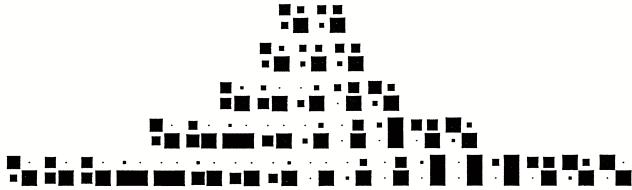


Figure 4.1

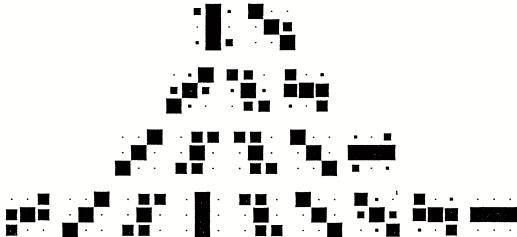


Figure 4.2

Figure 4: Figure 4.1: Hidden layer of N2 trained with wraparound. Figure 4.2: Output layer of N2 trained with wraparound

In Figure 4.1 and Figure 4.2 we present the results which we then obtained (compare Figure 3.1 and Figure 3.2). The trend in the orientation is monotonic throughout, and the results in Figure 4.2 compare very

favourably with those in Figure 2 (the two layer network).

In all the examples (Figure 1 to Figure 4.2 which we have presented the learning algorithm converged in a few seconds on a microcomputer. However we do anticipate that using networks of this type on a large image will involve very many associative memory operations (to find code vector indices). A look-up table representation of the mappings would then be very useful.

## V. CONCLUSIONS

Minimisation of distortion measures requires multilayer mappings to be topographic. Strictly, we have shown this only for treelike multilayer networks. We have also shown how to modify the original topographic mapping learning algorithm [1] to increase its convergence rate.

A three layer network can form line-like feature detectors which are just as good as those in a two layer network. However we have found it necessary to impose explicitly a topological constrain on the learning algorithm in order to obtain the “perfect” results of Figure 2 and Figure 4.2. This constraint is equivalent to introducing the prior knowledge that the training set of images has the topology of a circle ( $S^1$ ). We have also found (although we do not report it here) that more careful training without this extra topological constraint also yields results of this quality.

- 
- [1] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.  
[2] S P Luttrell, *Image compression using a multilayer neural*

- network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.

# Image Compression Using a Neural Network \*

S P Luttrell

Royal Signals and Radar Establishment, St Andrews Rd, Malvern, WR14 3PS, UK

Data compression of speckled images poses a non-trivial model identification problem. We train an unsupervised neural network on a set of archetype images in order to form an internal representation (or model) of the image features. We find that a multi-layer topographic mapping network has the necessary properties successfully to compress and reconstruct imagery. We show how to extend and improve upon existing learning algorithms for this type of network, and we express the network learning dynamics as a diffusion equation. We then present some examples of the application of this technique to synthetic aperture radar images.

Keywords: neural network, topographic mapping, synthetic aperture radar, speckle.

## I. INTRODUCTION

Successful data compression requires knowledge of the source statistics [8]. We shall be concerned with the compression of coherent images (specifically synthetic aperture radar (SAR) images) for which there do not exist any complete *a priori* statistical models. In a recent study [10] a conventional predictive compression system was used with some success on SAR imagery. In this paper we shall present a neural network solution to the same problem.

The main difficulty which is encountered when constructing a data compression system is identifying the correct data model. There has been much interest recently in neural networks [3] because they can be trained to form an internal representation (model) of structure in the training set. Such a model is non-trivial in the sense that the neural network forms a non-linear mapping from data space to its internal space, so it may extract complex features from the data. A neural network approach to model identification is ideal in situations where we have little or no *a priori* knowledge, but we have available a large amount of training data.

Model identification is a particular form of inverse problem in which we must recover the underlying rules and regulations (e.g. physical laws) which generate data sets. This inverse problem must be solved as a prelude to the conventional inverse problem of recovering the cause (e.g. placement of scatterers) of a particular data set. Various approaches are used to solve this problem in SAR image analysis, ranging from the use of generalised noise processes to derive clutter models [9], to a statistical analysis of a number of images to extract a heuristic image model [5]. In each case the large number of image pixels is "explained" in terms of a much smaller number of causal factors.

Conventionally in image processing [1] a feature space

of plausible image measurements (or statistics) is defined, and their means and covariance matrices for different image classes are used to extract a feature subspace suitable for solving whatever problem one has in mind. In particular, the prediction of the next pixel value in a row of a SAR image based on knowledge of features which depend only on previous pixel values would provide a solution to the SAR image compression problem. For this purpose we seek that set of features which is maximally correlated with the next pixel value, but which at the same time has minimum internal redundancy. This interpretation emerges naturally in the maximum entropy approach of [5].

On the other hand, neural network processing of images allows feature spaces to be crafted *ab initio* based on a set of representative images. The advantage of such networks is their adaptability when faced with the task of constructing feature spaces with subtle properties. Neural networks may be trained both in a supervised and an unsupervised fashion. For instance, the multilayer perceptron (supervised) may be used to distinguish between sonar returns from a rock and a metallic cylinder on the seabed [2]. For texture modelling, the maximum entropy approach in [5] leads to a rudimentary neural network model expressed as a Markov random field.

It is sometimes argued that a neural network model is not based on physical deductive reasoning, and is therefore a mere collection of *ad hoc* observations with limited predictive power. However, neural networks (like us!) turn out to be remarkably devious in their ability to reduce complicated structure to a few simple rules. This is very much in the spirit of physical model construction, so we find their use entirely palatable.

We shall concentrate on *unsupervised* neural networks because we wish to compress images for later analysis (and/or reconstruction) without prejudicing the information content of the feature space by supervising its formation. Typically supervised training leads to feature spaces which preserve only part of the information which is required to reconstruct an image. On the other hand unsupervised training has to be non-committal about what information it preserves, so it tends to preserve some information about everything.

In Section II we introduce a neural network structure

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

This appeared in Proceedings of the IGARSS88 Symposium on Remote Sensing: Moving Towards the 21st Century, Edinburgh, 1231-1238, 1988. © Controller, Her Majesty's Stationery Office, London, 1988.

which is suitable for compressing images, and in Section III we present a theoretical analysis of the properties of such networks. In Section IV we present the results of some experiments on SAR images.

## II. A NEURAL NETWORK

The data compression problem is one of discovering a mapping  $\mathbf{T}$  which takes an input  $\mathbf{x}_{in}$  into an output  $\mathbf{x}_{out}$  (both usually vector-valued) in such a way that a pseudo-inverse  $\tilde{\mathbf{T}}$  exists which performs the inverse mapping without too much distortion. Thus

$$\begin{aligned}\mathbf{T} \mathbf{x}_{in} &= \mathbf{x}_{out} \\ \tilde{\mathbf{T}} \mathbf{x}_{out} &= \tilde{\mathbf{x}}_{in} \\ \langle D(\mathbf{x}_{in}, \tilde{\mathbf{x}}_{in}) \rangle &< D_0\end{aligned}\quad (2.1)$$

where  $D(\mathbf{x}_{in}, \tilde{\mathbf{x}}_{in})$  is a distortion measure, angle brackets denote an average over  $\mathbf{x}_{in}$ , and  $D_0$  is some maximum acceptable average distortion. Furthermore data compression is achieved only if  $\mathbf{x}_{out}$  occupies fewer bits of memory than  $\mathbf{x}_{in}$ , where we ignore the fixed overhead used to store  $\mathbf{T}$  and  $\tilde{\mathbf{T}}$ . For image compression  $\mathbf{x}_{in}$  is a  $d_{in}$  dimensional vector of pixel values each component of which is represented by a small number of bits (usually 8), and  $\mathbf{x}_{out}$  is a  $d_{out}$  dimensional coded image. However  $d_{in}$  is usually enormous ( $10^6$  is not uncommon), so the compression mapping  $\mathbf{T}$  can operate on at most a very small proportion of the image pixels at a time: the mapping must be factorised.

A good review of vector quantisation (applied to the compression of speech) may be found in [7], and the network structure presented in [4] provides a means of adaptively designing vector quantisers. We have adopted an analogous approach in order to compress images, but we find that several extensions of the basic method of [4] are necessary in order to obtain a satisfactory real-time compression/reconstruction performance when  $d_{in}$  is large.

A vector quantiser mapping  $\mathbf{T}$  is specified as follows

$$\mathbf{T} \mathbf{x}_{in} = \mathbf{x}_{out} \equiv \mathbf{k}_0 \quad (2.2)$$

where  $d(\mathbf{x}_{in}, \mathbf{x}_{in}^k)$  is a suitable metric,  $\mathbf{k}_0$  minimises  $d(\mathbf{x}_{in}, \mathbf{x}_{in}^k)$  w.r.t.  $\mathbf{k}$ , and the  $\mathbf{x}_{in}^k$  form a set of  $n$  input code vectors against which  $\mathbf{x}_{in}$  is compared. When  $d(\mathbf{x}_{in}, \mathbf{x}_{in}^k)$  is Euclidean the  $\mathbf{x}_{in}^k$  partition the input space into  $n$  convex polyhedra whose surfaces lie on the hyperplanes  $d(\mathbf{x}_{in}, \mathbf{x}_{in}^j) = d(\mathbf{x}_{in}, \mathbf{x}_{in}^k)$   $j \neq k$ . The pseudo-inverse mapping  $\tilde{\mathbf{T}}$  is then given by

$$\tilde{\mathbf{T}} \mathbf{x}_{out} = \tilde{\mathbf{x}}_{in} \equiv \mathbf{x}_{in}^k \quad (2.3)$$

The task facing the neural network is to learn an optimal set of  $\mathbf{x}_{in}^k$  which minimises some average over  $\mathbf{x}_{in}$  of the reconstruction distortion  $D(\mathbf{x}_{in}, \tilde{\mathbf{x}}_{in})$ . It turns out that a further desirable property may easily be obtained from a neural network, namely that  $\tilde{\mathbf{x}}_{in} = \tilde{\mathbf{T}} \mathbf{x}_{out}$  is a continuous function of  $\mathbf{x}_{out}$ . This type of mapping is called

topographic because it attempts to preserve the topological structure of the input space in its coded output space representation.

### A. The network learning algorithm

[4] specifies the following type of algorithm for adaptively altering the  $\mathbf{x}_{in}^k$

$$\begin{aligned}\mathbf{x}_{in}^k(t+1) &= \mathbf{x}_{in}^k(t) + \epsilon(|\mathbf{k} - \mathbf{k}_0|, t) (\mathbf{x}_{in} - \mathbf{x}_{in}^k(t)) \\ |\mathbf{k} - \mathbf{k}_0| &\leq r_{max}(t)\end{aligned}\quad (2.4)$$

where  $\mathbf{k}_0$  minimises  $d(\mathbf{x}_{in}, \mathbf{x}_{in}^k)$  with respect to  $\mathbf{k}$ ,  $\epsilon(|\mathbf{k} - \mathbf{k}_0|, t)$  is an update scale factor which depends both on the number of prior adaptation cycles  $t$  and the output space separation  $|\mathbf{k} - \mathbf{k}_0|$ , and  $r_{max}(t)$  specifies the maximum range (in output space) over which a particular  $\mathbf{k}_0$  may exert its updating influence. Both the update scale factor and the maximum range are monotonically decreasing functions of  $t$  [4].

Physically the update equations may be regarded as modelling the motion of a damped stiff  $d_{out}$  dimensional hypersheet (specified as a mesh of code vectors  $\mathbf{x}_{in}^k$ ) in the  $d_{in}$  dimensional input space. The external force term generates the update step size, and the stiffness derives from the finite range  $r_{max}$  which also leads to the topographic property.

### B. Factorised mappings and look-up tables

For image processing we must factorise  $\mathbf{T}$  in order to break up  $\mathbf{x}_{in}$  into low dimensional pieces. We choose to break  $\mathbf{x}_{in}$  into non-overlapping contiguous rectangular pieces of image, and factorise  $\mathbf{T}$  into a corresponding product of identical mappings. For simplicity we shall henceforth use the full image notation  $\mathbf{x}_{in}$  to describe a single rectangular region, unless otherwise stated. The heavy computational demands of searching for the vector  $\mathbf{x}_{in}^k$  which minimises  $d(\mathbf{x}_{in}, \mathbf{x}_{in}^k)$  for each image rectangle are removed if we implement the mapping  $\mathbf{T} \mathbf{x}_{in} = \mathbf{x}_{out}$  in table look-up form. However this, in turn, places a tight upper bound on the number of bits  $A$  which we are permitted to use to represent  $\mathbf{x}_{in}$ , because  $2^A$  is the size of the address space of the look-up table. We shall assume that both  $\mathbf{x}_{in}$  and  $\mathbf{k}$  are represented using 8 bits, so  $d_{in} = 2$  is the only feasible choice given current memory technology. Whilst a worthwhile image compression can be obtained with such a small value of  $d_{in}$ , we would prefer to use a larger value in order to make use of any image redundancy which exists within multiplets of three or more pixels.

Further compression is made possible by iterating the coding procedure to produce an encoded version of the previous output, and so on. In general such an iteration is not permitted unless the vector quantisers are topographic (as they are here). Non-topographic mappings

lead to intermediate encodings of the image which do not have a simple metric structure, so they cannot be further encoded by a simple vector quantiser.

### C. Improved learning algorithm

We shall now present an improvement in the learning algorithm. The original algorithm [4] keeps the number  $n$  of code vectors  $\mathbf{x}_{in}^k$  fixed throughout the training process. This leads to long training times (tens of thousands of presentations) for moderate values of  $n$  (a few hundred). In the stiff sheet analogy the cause of such long run times is the need to simulate the dynamics of every point  $\mathbf{x}_{in}^k$  (labelled by  $k$ ) of the sheet. In practice the dynamics of the  $\mathbf{x}_{in}^k$  are highly correlated by the non-zero value of  $r_{max}(t)$  used in the learning algorithm, so much of the simulation is redundant. We choose to replace the original learning algorithm by one in which a small value of  $n$  is used initially to simulate only the long range structure of the sheet, and then  $n$  is progressively increased as  $t$  increases.

We find that the following changes to Equation 2.4 (here expressed only for  $d_{out} = 1$ ) yield a highly effective scheme for image compression (and many other) purposes.

$$\begin{aligned} r_{max}(t) &= 1 \\ \epsilon(0, t) &= \epsilon_0 \\ \epsilon(1, t) &= \epsilon_1 \\ n \rightarrow n(t) &\in \{2, 3, 5, 9, \dots, 2^{i+1}, \dots\} \\ n(0) &= 2 \end{aligned} \quad (2.5)$$

We step through the set of  $n(t)$  values by inserting a code vector  $\mathbf{x}_{in}^k$  in halfway between each pair of adjacent existing code vectors. Thus

$$\mathbf{x}'_{in}^k \equiv \frac{1}{2} (\mathbf{x}_{in}^k + \mathbf{x}_{in}^{k+1}) \quad k = 1, 2, \dots, n-1 \quad (2.6)$$

and then relabel all the code vectors into the range  $1, 2, \dots, 2n-1$ . In order to minimise the amount of redundant computation we insert code vectors only when the  $n$  old code vectors have converged to a random motion commensurate with the update noise level due to the non-zero values of  $\epsilon_0$  and  $\epsilon_1$ .

The monotonically decreasing values of  $\epsilon(|\mathbf{k} - \mathbf{k}_0|, t)$  and  $r_{max}(t)$  together with a fixed value of  $n$  which were used in Equation 2.4 correspond to the fixed values of  $\epsilon$  and  $r_{max}$  together with a monotonically increasing value of  $n(t)$  used in Equation 2.5. Equation 2.5 merely block renormalises the degrees of freedom of the stiff sheet to concentrate on the large scale structure (in  $\mathbf{x}_{in}$ -space) of the training set before proceeding to ever finer structure. For the image compression problem convergence to  $n = 257$  code vectors occurs in a few seconds (on a 5 MIPS MV20000).

### III. AN ANALYTIC MODEL

It is instructive to derive a simple analytic model which allows us to explore the properties of a vector quantiser without actually running simulations. To this end we shall introduce two new quantities  $P(\mathbf{x}_{in})$  and  $\rho(\mathbf{x}_{in})$  which are, respectively, the probability density function (PDF) of the training images, and the density in  $\mathbf{x}_{in}$ -space of the code vectors.  $\rho(\mathbf{x}_{in})$  is necessarily an approximate quantity because there is only a finite number  $n$  of code vectors. These quantities are normalised as

$$\begin{aligned} \int d\mathbf{x}_{in} P(\mathbf{x}_{in}) &= 1 \\ \int d\mathbf{x}_{in} \rho(\mathbf{x}_{in}) &= n \end{aligned} \quad (3.1)$$

Denote as  $S_k$  the set of  $\mathbf{x}_{in}$  which map to  $k$

$$S_k \equiv \{\mathbf{x}_{in} : T \mathbf{x}_{in} = k\} \quad (3.2)$$

By definition we recover  $\mathbf{x}_{in}^k$  from  $\rho(\mathbf{x}_{in})$  by integration

$$\mathbf{x}_{in}^k \equiv \frac{\int d\mathbf{x}_{in} \rho(\mathbf{x}_{in}) \mathbf{x}_{in}}{\int d\mathbf{x}_{in} \rho(\mathbf{x}_{in})} \quad (3.3)$$

The updates used in both Equation 2.4 and Equation 2.5 cause  $\mathbf{x}_{in}^k$  to be moved by an amount proportional to  $\mathbf{x}_{in} - \mathbf{x}_{in}^k$  for each  $\mathbf{x}_{in}$  in  $S_k$ . From Equation 2.4 (or Equation 2.5) for fixed  $t$  and  $k=k_0$  the average update step induced on  $\mathbf{x}_{in}^k$  is given by (ignoring the constant of proportionality)

$$\delta \mathbf{x}_{in}^k = \int_{S_k} d\mathbf{x}_{in} P(\mathbf{x}_{in}) (\mathbf{x}_{in} - \mathbf{x}_{in}^k) \quad (3.4)$$

The restriction to  $k=k_0$  destroys the topographic property but it leads to essentially identical results for  $\rho(\mathbf{x}_{in})$ . Using the Equation 3.3 we may rewrite Equation 3.4 as

$$\delta \mathbf{x}_{in}^k = \frac{\int d\mathbf{x}_{in} dy_{in} P(\mathbf{x}_{in}) \rho(y_{in}) (\mathbf{x}_{in} - y_{in})}{\int d\mathbf{x}_{in} \rho(\mathbf{x}_{in})} \quad (3.5)$$

Now assuming that  $P(\mathbf{x}_{in})$  and  $\rho(\mathbf{x}_{in})$  are slowly varying functions of  $\mathbf{x}_{in}$  in comparison with the size of  $S_k$ , we may approximate them using the first two terms of a Taylor expansion about any suitable point in  $S_k$  ( $\mathbf{x}_{in} = \mathbf{x}_{in}^k$ , say). Thus

$$\begin{aligned} P(\mathbf{x}_{in}) &= P_0 + \nabla P_0 \cdot (\mathbf{x}_{in} - \mathbf{x}_{in}^k) + \dots \\ \rho(\mathbf{x}_{in}) &= \rho_0 + \nabla \rho_0 \cdot (\mathbf{x}_{in} - \mathbf{x}_{in}^k) + \dots \end{aligned} \quad (3.6)$$

where the subscript 0 is used to denote evaluation at  $\mathbf{x}_{in} = \mathbf{x}_{in}^k$ . For clarity let us define

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_{in} - \mathbf{x}_{in}^k \\ \mathbf{y} &= \mathbf{y}_{in} - \mathbf{y}_{in}^k \\ S_k^0 &\equiv \{\mathbf{x} : \mathbf{x} + \mathbf{x}_{in}^k \in S_k\} \end{aligned} \quad (3.7)$$

whence from Equation 3.5, Equation 3.6 and Equation 3.7 we obtain an expression for the  $i^{\text{th}}$  component  $\delta \mathbf{x}_{\text{in},i}^k$  of  $\delta \mathbf{x}_{\text{in}}^k$

$$\delta \mathbf{x}_{\text{in},i}^k \simeq \frac{\delta_i^1 + \delta_i^2 + \delta_i^3}{\nu} \quad (3.8)$$

where

$$\begin{aligned} \delta_i^1 &\equiv P_0 \rho_0 \int_{S_k^0} d\mathbf{x} d\mathbf{y} (\mathbf{x}_i - \mathbf{y}_i) \\ \delta_i^2 &\equiv P_0 \sum_{j=1}^{d_{\text{in}}} \frac{\partial \rho_0}{\partial x_j} \int_{S_k^0} d\mathbf{x} d\mathbf{y} (\mathbf{x}_i - \mathbf{y}_i) y_j \\ \delta_i^3 &\equiv \rho_0 \sum_{j=1}^{d_{\text{in}}} \frac{\partial P_0}{\partial x_j} \int_{S_k^0} d\mathbf{x} d\mathbf{y} (\mathbf{x}_i - \mathbf{y}_i) \mathbf{x}_j \\ \nu &\equiv \rho_0 \int_{S_k^0} d\mathbf{x} + \sum_{j=1}^{d_{\text{in}}} \frac{\partial \rho_0}{\partial x_j} \int_{S_k^0} d\mathbf{x} x_j \end{aligned} \quad (3.9)$$

The  $\delta_i^1$  term vanishes by symmetry which removes all leading order terms from the numerator, so the derivative term in  $\nu$  may now be discarded. The  $x_i y_j$  parts of  $\delta_i^2$  and  $\delta_i^3$  are small compared with the  $x_i x_j$  and  $y_i y_j$  parts, and so may be discarded. Equation 3.8 and Equation 3.9 thus simplify to

$$\delta \mathbf{x}_{\text{in},i}^k \simeq \sum_{j=1}^{d_{\text{in}}} \left( \int_{S_k^0} d\mathbf{x} x_i x_j \right) \left( \frac{\partial P_0}{\partial x_j} - P_0 \frac{\partial \log(\rho_0)}{\partial x_j} \right) \quad (3.10)$$

The integral in Equation 3.10 yields the inertia tensor for the volume  $S_k$  with  $\mathbf{x}_{\text{in}}^k$  used as origin. If we assume that the isotropic component is dominant (i.e. approximately proportional to  $\delta_{i,j}$ ) then

$$\begin{aligned} \delta \mathbf{x}_{\text{in}}^k &\propto \nabla P_0 - P_0 \nabla \log(\rho_0) \\ &= P_0 \nabla \log \left( \frac{P_0}{\rho_0} \right) \end{aligned} \quad (3.11)$$

We may define a current  $\mathbf{J}$  as

$$\begin{aligned} \mathbf{J} &\equiv \rho \mathbf{v} \\ &\propto \rho P \nabla \log \left( \frac{P_0}{\rho_0} \right) \end{aligned} \quad (3.12)$$

where  $\mathbf{v}$  is a velocity. This leads to the diffusion equation

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathbf{J} \quad (3.13)$$

for the dynamics of the code vector density under the influence of updates of the type given in Equation 2.4. In equilibrium  $\mathbf{J} = \mathbf{0}$  whence  $P(\mathbf{x}_{\text{in}}) \propto \rho(\mathbf{x}_{\text{in}})$ . An argument which leads to this result for  $d_{\text{in}} = 1$  has been presented

in [4]. The  $\mathbf{x}_{\text{in}}^k$  thus concentrate in regions where  $P(\mathbf{x}_{\text{in}})$  is largest. Now define the output probabilities as  $p_k$ , then in equilibrium

$$p_k \simeq \frac{P(\mathbf{x}_{\text{in}}^k)}{\rho(\mathbf{x}_{\text{in}}^k)} \simeq \text{constant} \quad (3.14)$$

where  $\frac{1}{\rho(\mathbf{x}_{\text{in}}^k)}$  is the volume of  $S_k$ . So, in equilibrium, the code vectors  $\mathbf{x}_{\text{in}}^k$  are used with equal probability.

We may also derive these results by minimising a logarithmic distortion measure. Thus the distortion volume associated with reconstructions  $\tilde{\mathbf{x}}_{\text{in}}$  from an original  $\mathbf{x}_{\text{in}}$  is  $\frac{1}{\rho(\mathbf{x}_{\text{in}})}$ . Define an average logarithmic distortion as

$$D \equiv \int d\mathbf{x}_{\text{in}} P(\mathbf{x}_{\text{in}}) \log \left( \frac{1}{\rho(\mathbf{x}_{\text{in}})} \right) \quad (3.15)$$

Introducing a Lagrange multiplier to hold  $\int d\mathbf{x}_{\text{in}} \rho(\mathbf{x}_{\text{in}})$  constant, and then functionally differentiating yields

$$\frac{\delta(D - \lambda \int d\mathbf{x}_{\text{in}} \rho(\mathbf{x}_{\text{in}}))}{\delta \rho(\mathbf{x})} = \int d\mathbf{x}_{\text{in}} \left( \frac{P(\mathbf{x}_{\text{in}})}{\rho(\mathbf{x}_{\text{in}})} - \lambda \right) \delta(\mathbf{x} - \mathbf{x}_{\text{in}}) \quad (3.16)$$

which leads to a stationary solution  $P(\mathbf{x}_{\text{in}}) \propto \rho(\mathbf{x}_{\text{in}})$  as required. Just as we have related the update prescription in Equation 2.4 to the distortion optimisation in Equation 3.16, we may relate other update prescriptions to other optimisations, and vice versa.

#### IV. APPLICATION TO SAR IMAGES

We have implemented a version of the multi-layer topographic mapping network which uses  $d_{\text{in}} = 2$ ,  $d_{\text{out}} = 1$  at each stage. For training we use  $\epsilon_0 = 0.1$  and  $\epsilon_1 = 0.02$  with  $32n$  updates between each episode of code vector insertion. To assist the formulation as a table look-up scheme we iterate until  $n = 257$  and then discard 1 vector (followed by re-equilibration) in order to obtain 256 code vectors which may be indexed using 8 bits. We also represent the input image using 8 bit pixels. Thus each stage of the hierarchical mapping is represented in a table with  $2^{16}$  byte-sized entries.

A suitable choice of which pairs of pixels to use as  $x_{\text{in}}$ , maybe arrived at by physical argument. SAR images have a lot of short length scale correlations which can be used to immediate advantage in image compression. Thus the first stage of the mapping should code adjacent pairs of pixels. Furthermore, the SAR imagery which we have used has a poorer range than azimuth resolution, so we choose the pairs of pixels to lie adjacent in the range direction to take advantage of the extra range correlations. The choice of pixel pairs at later stages of compression is less clear cut, so we have chosen to alternate the initial prescription between the range and azimuth directions - flipping once per stage of the hierarchical mapping. Thus a 2 stage mapping will compress

by a factor of 2 in each of the range and azimuth directions, and a 3 stage mapping will compress by a further factor of 2 in the range direction.

We start with autofocussed SAR images with a resolution of 1.5m. We then average the moduli of all contiguous non-overlapping  $2 \times 2$  pixel blocks to produce a smoother image (with  $\frac{1}{4}$  of the number pixels of the original image). This is similar to multilook limaging, and it reduces the effect of speckle without losing much significant image detail.

Figure 1 is a  $256 \times 256$  SAR image of an urban region produced in this way. Figure 2, Figure 3 and Figure 4 show the reconstructions of Figure 1 which were obtained after compression by factors of 2, 4 and 8 respectively. Visually Figure 2 and Figure 3 are very accurate renditions of the original, whilst Figure 4 is found somewhat wanting. A quantitative comparison of the reconstructions with the original can be found in [6].

We also processed a SAR image of a rural region. Figure 5, Figure 6, Figure 7 and Figure 8 are analogous to Figure 1, Figure 2, Figure 3 and Figure 4 *except that* the code vectors used were those derived from Figure 1 (not Figure 5). The quality of the reconstructions demonstrates that the data compression is not sensitive to the precise details of the image statistics. This allows us to

use a single set of look-up tables to compress a wide variety of SAR images, thus avoiding the associated storage overhead.

## V. CONCLUSIONS

We have shown that a multi-layer topographic neural network may successfully be used to compress SAR images by factors of 2, 4 and 8. The formulation of the multi-layer mapping as a set of table look-up operations leads to extremely fast compression and reconstruction of SAR images, so the technique could be used in real-time applications.

Once the basic network software has been written no further significant effort is required because model identification occurs automatically as the neural network is trained. The same software could, in principle, be used to learn suitable codings for other types of imagery. The most pleasing result is the fact that the neural network has proved itself to be capable of unsupervised learning of a non-trivial mapping from data space into an internal representation space.

- [1] P A Devijver and J Kittler, *Pattern recognition: a statistical approach*, Prentice-Hall, London, 1982.
- [2] R P Gorman and T J Sejnowski, *Analysis of hidden units in a layered network trained to classify sonar targets*, Neural Networks **1** (1988), no. 1, 75–89.
- [3] S Grossberg, *Nonlinear neural networks: Principles, mechanisms, and architectures*, Neural Networks **1** (1988), no. 1, 17–61.
- [4] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [5] S P Luttrell, *A maximum entropy approach to sampling function design*, Inverse Problems **4** (1988), no. 3, 829–841.
- [6] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
- [7] J Makhoul, S Roucos, and H Gish, *Vector quantisation in speech coding*, Proceedings of the IEEE **73** (1985), no. 11, 1551–1588.
- [8] R J McEliece, *The theory of information and coding*, Addison-Wesley, Reading, 1977.
- [9] C J Oliver, *On the simulation of coherent clutter textures with arbitrary spectra*, Inverse Problems **3** (1987), no. 3, 463–475.
- [10] S A Werness, *Statistical evaluation of predictive data compression systems*, IEEE Transactions on Acoustics, Speech, and Signal Processing **35** (1987), no. 8, 1190–1198.



Figure 1: SAR image of an urban region. Used to train code vectors for compression of both Figure 1 and Figure 5.



Figure 2: Reconstruction of Figure 1 after a range compression of 2.



Figure 3: Reconstruction of Figure 1 after range and azimuth compressions of 2.



Figure 4: Reconstruction of Figure 1 after a range compression of 4 and an azimuth compression of 2.

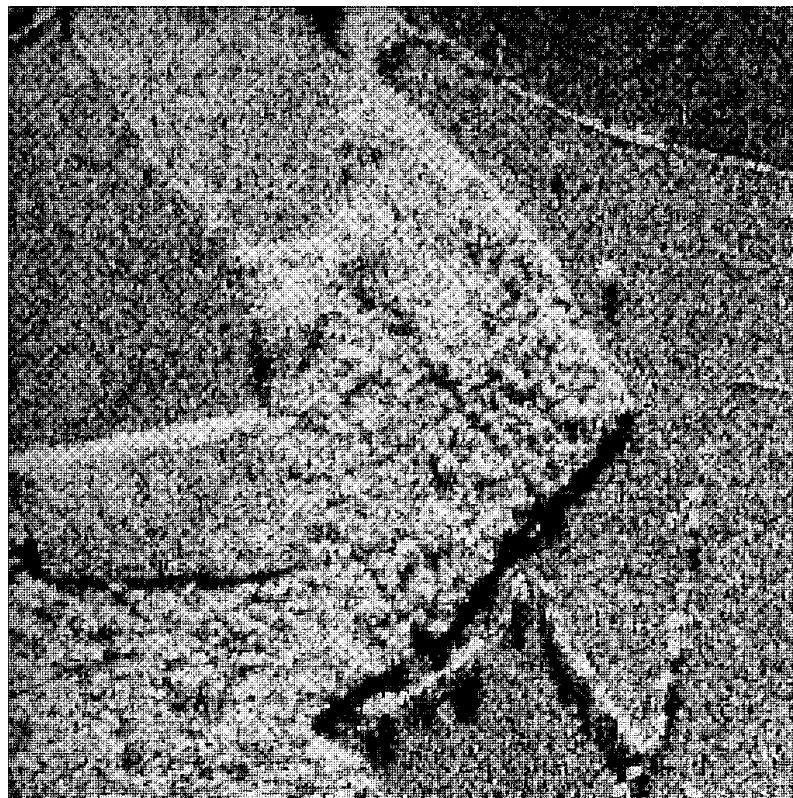


Figure 5: SAR image of a rural region.

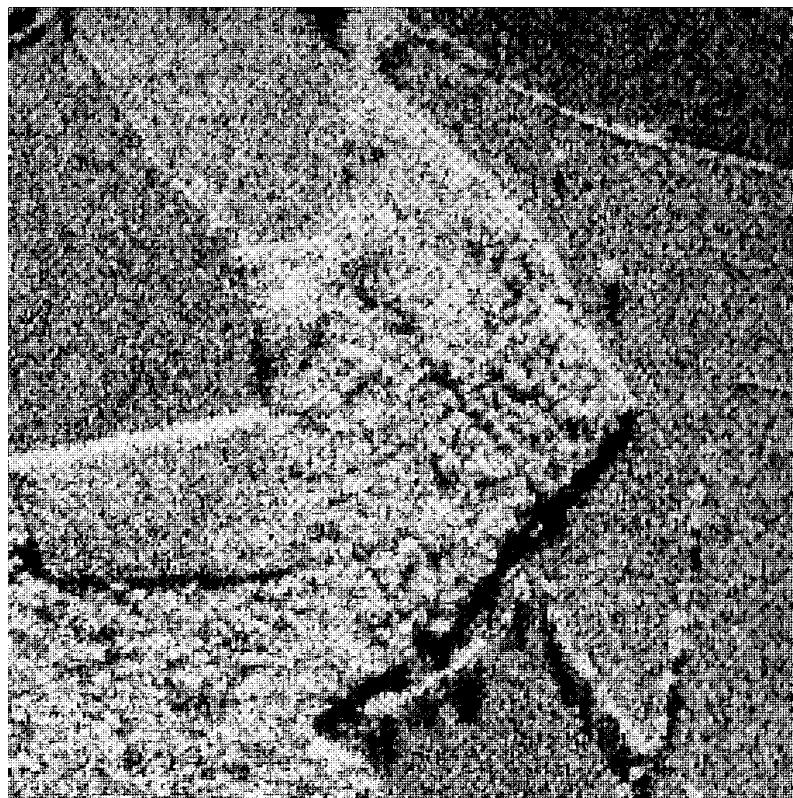


Figure 6: Reconstruction of Figure 5 after a range compression of 2.

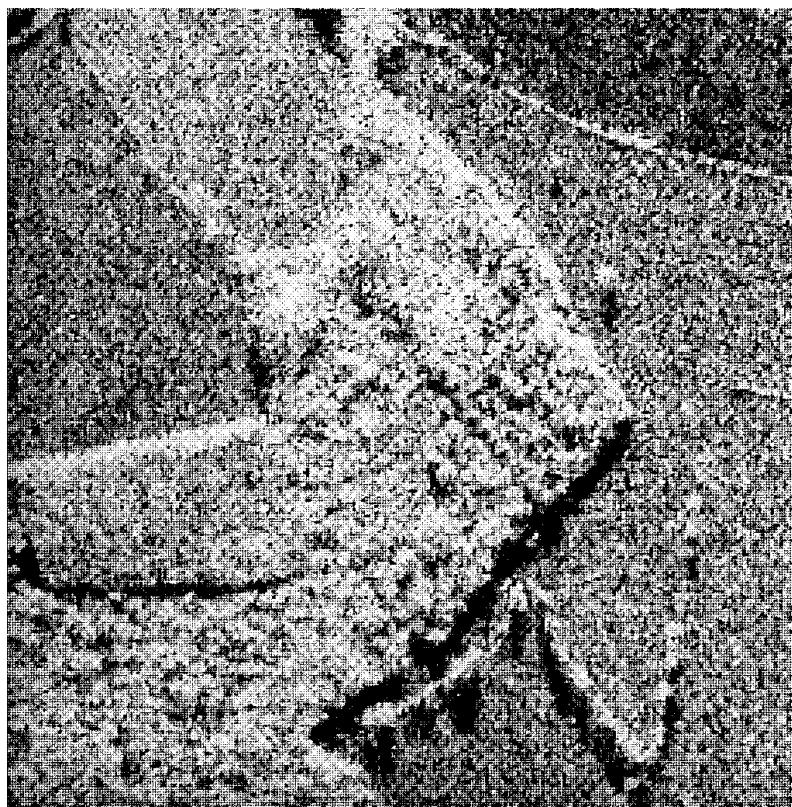


Figure 7: Reconstruction of Figure 5 after range and azimuth compressions of 2.



Figure 8: Reconstruction of Figure 5 after a range compression of 4 and an azimuth compression of 2.



# The Use of Bayesian and Entropic Methods in Neural Network Theory \*

S. P. Luttrell

*Royal Signals and Radar Establishment, St. Andrews Road, Malvern, WORCS., WR14 SPS, U.K.*

There has been much interest recently in the use of neural networks to solve complicated information processing problems such as those which arise in signal and image processing. In this paper we review Markov random field (MRF) neural network techniques for representing joint probability density functions (PDF). The “Boltzmann machine” serves as the paradigm, and we present a generalised version of its learning algorithm. We also present a technique for designing MRF potentials with low information redundancy for modelling image texture. To improve further the computational efficiency of such neural networks we introduce a novel method of cluster decomposing a PDF by using topographic mappings. The outcome of this programme is a means of designing sampling functions for extracting information from datasets (typically images).

## I. INTRODUCTION

The image processing community has shown much interest in the use of Markov random field models to describe probability density functions for use in Bayesian image reconstruction schemes [4, 5]. If we denote the field state as  $\mathbf{x}$  and the PDF over states as  $P(\mathbf{x})$  then an MRF is defined by a consistent set of conditional PDFs (called characteristics) amongst the components of  $\mathbf{x}$ . It follows from the Hammersley-Clifford theorem that corresponding to each consistently defined MRF there is an equivalent Gibbs distribution [2, 10, 21], so  $P(\mathbf{x})$  may be written as

$$P(\mathbf{x}) = \frac{1}{Z} \exp(-\mathbf{k} \cdot \mathbf{s}(\mathbf{x})) \quad (1.1)$$

where  $\mathbf{s}(\mathbf{x})$  is a vector potential,  $\mathbf{k}$  is a vector of coefficients, and  $Z$  is a partition function. We use an unconventional symbol  $\mathbf{s}$  to denote the potential because it is in fact a set of sampling functions of  $\mathbf{x}$ . Equation 1.1 defines a  $P(\mathbf{x})$  from which samples  $\mathbf{x}$  may be drawn by using a Monte Carlo scheme such as the Metropolis algorithm [20] or some variant thereof.

$P(\mathbf{x})$  is, of course, the maximum entropy PDF (with a uniform prior) which is consistent with the set of constraints  $\langle \mathbf{s}(\mathbf{x}) \rangle = \mathbf{s}_0$ , where  $\langle \dots \rangle$  denotes an average over  $\mathbf{x}$  [6–9]. Note that  $P(\mathbf{x})$  has the form given in Equation 1.1 if, and only if, the functional derivative  $\frac{\delta H}{\delta P(\mathbf{x})}$  lies in the function subspace spanned by the vector of functional derivatives  $\frac{\delta \langle \mathbf{s}(\mathbf{x}) \rangle}{\delta P(\mathbf{x})}$ , where  $H$  denotes the entropy of  $P(\mathbf{x})$ .

The purpose of this paper is to extend the above MRF scheme by introducing a greater degree of adaptability into the model. Thus in Section II A we shall explain how the Boltzmann machine neural network (and its generalisations) can be used to learn MRF models adaptively, and in Section II B we shall explain how economical MRF

models of image texture can be constructed. In Section III A we shall introduce a novel form of multilayer neural network which allows maximum entropy reconstructions of the input PDF to be constructed with minimal computational effort, and in Section III B we shall explain how topographic mappings can be used to implement the layer to layer transformations in such a network.

## II. G-MAXIMISATION MODELS

Equation 1.1 is inflexible because  $\mathbf{s}(\mathbf{x})$  must be selected by hand: there is no means of deriving  $\mathbf{s}(\mathbf{x})$  adaptively from a training set of samples  $\mathbf{x}$  following some observed PDF  $P_0(\mathbf{x})$  ( $\neq P(\mathbf{x})$  in general). In order to acquire  $\mathbf{s}(\mathbf{x})$  adaptively we need a measure of the similarity of the (true) observed PDF  $P_0(\mathbf{x})$  and the (maximum entropy) hypothesised PDF  $P(\mathbf{x})$  defined in Equation 1.1. Define the relative entropy  $G$

$$G \equiv - \int d\mathbf{x} P_0(\mathbf{x}) \log \left( \frac{P_0(\mathbf{x})}{P(\mathbf{x})} \right) \quad (2.1)$$

Assuming base 2 logarithms,  $2^{nG}$  is the probability that the hypothesised  $P(\mathbf{x})$  will generate high probability n-sample sequences of states  $\mathbf{x}$  which belong to the high probability set generated by the true  $P_0(\mathbf{x})$ , where  $n$  is asymptotically large. Note that  $G \leq 0$ , with  $G = 0$  iff  $P(\mathbf{x}) = P_0(\mathbf{x})$ . We shall deal with two types of adaptation in Section II A and Section II B.

### A. THE BOLTZMANN MACHINE

For a fixed set of potentials  $\mathbf{s}(\mathbf{x})$  we may optimise  $\mathbf{k}$  by G-maximisation using

$$\frac{\partial G}{\partial k_i} = \int d\mathbf{x} \frac{P_0(\mathbf{x})}{P(\mathbf{x})} \frac{\partial P(\mathbf{x})}{\partial k_i} = \langle s_i(\mathbf{x}) \rangle_{P(\mathbf{x})} - \langle s_i(\mathbf{x}) \rangle_{P_0(\mathbf{x})} \quad (2.2)$$

$\frac{\partial G}{\partial k_i} = 0$  when the constraints  $\langle \mathbf{s}(\mathbf{x}) \rangle = \mathbf{s}_0$  are satisfied, so hill-climbing  $G$  in  $\mathbf{k}$ -space yields the required maximum entropy PDF estimate. The first term in Equation 2.2 is estimated from Monte Carlo samples of  $P(\mathbf{x})$  defined in

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This appeared in Maximum Entropy and Bayesian Methods, Kluwer, J. Skilling (ed.), 363–370, 1989. © 1989 Controller, HMSO, London.

Equation 1.1, whereas the second term is estimated from the training set which implicitly defines  $P_0(\mathbf{x})$ .

We may construct a version of Equation 2.2 in which  $\mathbf{s}(\mathbf{x})$  itself is effectively learnt at the same time as the coefficient vector  $\mathbf{k}$ . Thus we introduce hidden variables  $\mathbf{h}$  by augmenting  $\mathbf{s}(\mathbf{x})$  to become  $\mathbf{s}(\mathbf{x}, \mathbf{h})$ . Denoting the associated Gibbs distribution as  $P(\mathbf{x}, \mathbf{h})$  leads to

$$P(\mathbf{x}, \mathbf{h}) = \frac{1}{Z} \int d\mathbf{h} \exp(-\mathbf{k} \cdot \mathbf{s}(\mathbf{x}, \mathbf{h})) \quad (2.3)$$

where  $Z$  is now the partition function over all states  $(\mathbf{x}, \mathbf{h})$ . Defining  $G$  as in Equation 1.1 then leads to

$$\frac{\partial G}{\partial k_i} = \langle s_i(\mathbf{x}, \mathbf{h}) \rangle_{P(\mathbf{x}, \mathbf{h})} - \langle s_i(\mathbf{x}, \mathbf{h}) \rangle_{P(\mathbf{h}|\mathbf{x})P_0(\mathbf{x})} \quad (2.4)$$

The first term in Equation 2.4 is estimated by using a Monte Carlo procedure, whereas the second term is a hybrid which uses the training set to provide samples from  $P_0(\mathbf{x})$  and a Monte Carlo procedure (with  $\mathbf{x}$  held constant) to provide samples from  $P(\mathbf{h}|\mathbf{x})$ . The advantage of introducing the hidden variables  $\mathbf{h}$  is that a complicated  $P(\mathbf{x})$  can be generated by using simple  $\mathbf{s}(\mathbf{x}, \mathbf{h})$  because the effect of  $\mathbf{h} - \mathbf{h}$  and  $\mathbf{h} - \mathbf{x}$  interactions “dresses” the bare  $\mathbf{x} - \mathbf{x}$  interactions. This amounts to learning  $\mathbf{s}(\mathbf{x})$  adaptively by adjusting the strengths of the interactions with and amongst the hidden variables.

The so-called Boltzmann machine [1] is a simple form of hidden variable model which uses binary variables  $\mathbf{x}$  and  $\mathbf{h}$  and quadratic interactions  $\mathbf{s}(\mathbf{x}, \mathbf{h})$  together with  $G$ -maximisation. More general hidden variable models have been discussed elsewhere [12, 22]. Whilst the Boltzmann machine is very flexible in its ability to adapt to the statistical properties of  $P_0(\mathbf{x})$ , it is computationally very inefficient due to the extensive Monte Carlo simulations which are required.

## B. DESIGNING POTENTIALS

There is another  $G$ -maximisation approach to learning  $\mathbf{s}(\mathbf{x})$  for which the constraints are not on  $\langle \mathbf{s}(\mathbf{x}) \rangle$  but on the whole PDF  $p_0(\mathbf{s})$  of  $\mathbf{s}(\mathbf{x})$ . In general  $p_0(\mathbf{s})$  is given by

$$p_0(\mathbf{s}) \equiv \int d\mathbf{x} P_0(\mathbf{x}) \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \quad (2.5)$$

where the Dirac delta function constrains the  $\mathbf{x}$  integral as required. If the  $\mathbf{x}$  dependence of the observed  $P_0(\mathbf{x})$  can be expressed entirely in terms of  $\mathbf{s}(\mathbf{x})$ , then  $\mathbf{s}(\mathbf{x})$  is a sufficient set of statistics [3, 16]. The maximum entropy reconstruction (with a uniform prior) will then be  $P_0(\mathbf{x})$  if the entire PDF  $p_0(\mathbf{s})$  is used as a constraint. On the other hand, when  $\mathbf{s}(\mathbf{x})$  is not a sufficient set of statistics, the maximum entropy method will, as usual, give the least committal reconstruction  $P(\mathbf{x})$  of  $P_0(\mathbf{x})$  which is

consistent with  $p_0(\mathbf{s})$  [18]

$$\begin{aligned} P(\mathbf{x}) &= \frac{p_0(\mathbf{s}(\mathbf{x}))}{Z(\mathbf{s}(\mathbf{x}))} \\ Z(\mathbf{s}) &= \int d\mathbf{x} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \end{aligned} \quad (2.6)$$

$Z(\mathbf{s})$  is proportional to the number of states  $\mathbf{x}$  which map to  $\mathbf{s}$ : it plays the role of a normalisation factor (it is not a partition function). With this interpretation the expression for  $P(\mathbf{x})$  in Equation 2.6 is intuitively obvious.

Using the definition of  $G$  in Equation 2.1, substituting in  $P(\mathbf{x})$  from Equation 2.6, and using the definition of  $p_0(\mathbf{s})$  in Equation 2.5 yields

$$G = G_0 + \int d\mathbf{s} p_0(\mathbf{s}) \log \left( \frac{p_0(\mathbf{s})}{Z(\mathbf{s})} \right) \quad (2.7)$$

where  $G_0$  is a constant. In order to optimise  $G$  we envisage two distinct types of change to  $\mathbf{s}(\mathbf{x})$ : a dimensionality preserving perturbation  $\mathbf{s}(\mathbf{x}) \rightarrow \mathbf{s}(\mathbf{x}) + \epsilon \mathbf{t}(x)$ , and a dimensionality increasing change  $\mathbf{s}(\mathbf{x}) \rightarrow (\mathbf{s}(\mathbf{x}), \mathbf{t}(\mathbf{x}))$ . We shall now present the results for these two cases.

For perturbations of the form  $\mathbf{s}(\mathbf{x}) \rightarrow \mathbf{s}(\mathbf{x}) + \epsilon \mathbf{t}(x)$  we require the functional derivative  $\frac{\delta G}{\delta \mathbf{s}(\mathbf{x})}$  which, in turn, requires the results

$$\begin{aligned} \frac{\delta p_0(\mathbf{s})}{\delta \mathbf{s}(\mathbf{x})} &= -P_0(\mathbf{x}) \nabla_{\mathbf{s}} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \\ \frac{\delta Z(\mathbf{s})}{\delta \mathbf{s}(\mathbf{x})} &= -\nabla_{\mathbf{s}} \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \end{aligned} \quad (2.8)$$

where  $\nabla_{\mathbf{s}}$  is the derivative operator wrt  $\mathbf{s}$ : these results permit functional differentiation to be replaced by ordinary differentiation. After some manipulation we then obtain the functional derivative in the form [18]

$$\frac{\delta G}{\delta \mathbf{s}(\mathbf{x})} = (P_0(\mathbf{x}) - P(\mathbf{x})) \nabla_{\mathbf{s}} \log \left( \frac{p_0(\mathbf{s})}{Z(\mathbf{s})} \right)_{\mathbf{s}=\mathbf{s}(\mathbf{x})} \quad (2.9)$$

which yields a change  $\Delta G_1$  in  $G$  given by

$$\Delta G_1 = \epsilon \int d\mathbf{s} dt (p_0(\mathbf{s}, \mathbf{t}) - p(\mathbf{s}, \mathbf{t})) \mathbf{t} \cdot \nabla_{\mathbf{s}} \log \left( \frac{p_0(\mathbf{s})}{Z(\mathbf{s})} \right) \quad (2.10)$$

where

$$\begin{aligned} p_0(\mathbf{s}, \mathbf{t}) &\equiv \int d\mathbf{x} P_0(\mathbf{x}) \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \delta(\mathbf{t} - \mathbf{t}(\mathbf{x})) \\ p(\mathbf{s}, \mathbf{t}) &\equiv \int d\mathbf{x} P(\mathbf{x}) \delta(\mathbf{s} - \mathbf{s}(\mathbf{x})) \delta(\mathbf{t} - \mathbf{t}(\mathbf{x})) \end{aligned} \quad (2.11)$$

Alternatively, for changes of the form  $\mathbf{s}(\mathbf{x}) \rightarrow (\mathbf{s}(\mathbf{x}), \mathbf{t}(\mathbf{x}))$  we obtain a change  $\Delta G_2$  in  $G$  given by [18]

$$\Delta G_2 = \int d\mathbf{s} dt p_0(\mathbf{s}, \mathbf{t}) \log \left( \frac{p_0(\mathbf{s}, \mathbf{t})}{p(\mathbf{s}, \mathbf{t})} \right) \quad (2.12)$$

Both  $\Delta G_1$  and  $\Delta G_2$  depend on a comparison of  $p_0(\mathbf{s}, \mathbf{t})$  and  $p(\mathbf{s}, \mathbf{t})$ . Any differences between  $p_0(\mathbf{s}, \mathbf{t})$  and  $p(\mathbf{s}, \mathbf{t})$

are caused by the presence of structure in  $P_0(\mathbf{x})$  which is not measured by  $\mathbf{s}(\mathbf{x})$  alone.

We have used the techniques outlined in this subsection to design MRF coherent image texture models [13–15], where we assumed that  $P_0(\mathbf{x})$  describes spatially stationary statistics (i.e.  $P_0(L\mathbf{x}) = P_0(\mathbf{x})$  where  $L$  is any image translation operator). It then suffices to consider only  $\mathbf{s}(\mathbf{x})$  for which  $s(L\mathbf{x}) = s(\mathbf{x})$ , which severely restricts the set of feasible  $\mathbf{s}(\mathbf{x})$ . The approach which we have presented in this subsection does not involve any hidden variables, so it has difficulty dealing with subtle properties of  $P_0(\mathbf{x})$  which are better described by introducing “spectator variables”. However it does successfully model short range textural properties.

### III. CLUSTER DECOMPOSITION MODEL

We now propose a novel scheme for representing PDFs which completely eliminates the need for Monte Carlo simulations, whilst retaining the flexibility of the adaptive approach. This improvement is obtained at the cost of imposing an artificial hierarchical structure on the PDF reconstruction.

#### A. MULTILAYER NEURAL NETWORK

For simplicity consider the following situation

$$\begin{aligned}\mathbf{x} &\equiv (\mathbf{x}_1, \mathbf{x}_2) \\ \mathbf{s}(\mathbf{x}) &\equiv (\mathbf{s}_1(\mathbf{x}_1), \mathbf{s}_2(\mathbf{x}_2))\end{aligned}\quad (3.1)$$

Now suppose that the estimated values of  $P_0(\mathbf{x}_1)$ ,  $P_0(\mathbf{x}_2)$  and  $p_0(\mathbf{s}_1, \mathbf{s}_2)$  are used as constraints on a maximum entropy reconstruction  $P(\mathbf{x})$  of  $P_0(\mathbf{x})$  (with a uniform prior). After some algebra which is similar to that which led to Equation 2.6 we obtain

$$P(\mathbf{x}) = P_0(\mathbf{x}_1) P_0(\mathbf{x}_2) \left( \frac{p_0(\mathbf{s}_1(\mathbf{x}_1), \mathbf{s}_2(\mathbf{x}_2))}{p_0(\mathbf{s}_1(\mathbf{x}_1)) p_0(\mathbf{s}_2(\mathbf{x}_2))} \right) \quad (3.2)$$

This expression has a natural interpretation. If  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are independent random variables then so also are  $\mathbf{s}_1$  and  $\mathbf{s}_2$ , yielding  $p_0(\mathbf{s}_1, \mathbf{s}_2) = p_0(\mathbf{s}_1) p_0(\mathbf{s}_2)$ , hence  $P(\mathbf{x}) = P_0(\mathbf{x}_1) P_0(\mathbf{x}_2)$ , as expected. On the other hand, if  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are mutually dependent then there is an additional correction term

This approach to estimating  $P_0(\mathbf{x})$  is usually simpler than specifying  $P_0(\mathbf{x})$  directly when  $\dim(\mathbf{s}_1) < \dim(\mathbf{x}_1)$  and  $\dim(\mathbf{s}_2) < \dim(\mathbf{x}_2)$ . This is because the cost of exhaustively specifying a PDF increases exponentially with the dimensionality of its underlying space, so specifying three low dimensional PDFs  $P_0(\mathbf{x}_1)$ ,  $P_0(\mathbf{x}_2)$  and  $p_0(\mathbf{s}_1, \mathbf{s}_2)$  is usually cheaper than specifying one high dimensional PDF  $P_0(\mathbf{x})$ .

The above decomposition of  $P_0(\mathbf{x})$  immediately generalises to

$$P(\mathbf{x}) = \left( \prod_{i=1}^n P_0(\mathbf{x}_i) \right) \left( \frac{p_0(\mathbf{s}(\mathbf{x}))}{\prod_{i=1}^n p_0(\mathbf{s}_i(\mathbf{x}_i))} \right) \quad (3.3)$$

where  $\mathbf{x} \equiv (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ ,  $\mathbf{s} \equiv (\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n)$ ,  $\mathbf{s}_i \equiv \mathbf{s}_i(\mathbf{x}_i)$  and  $\dim(\mathbf{s}_i) < \dim(\mathbf{x}_i)$  for  $i = 1, 2, \dots, n$ . Now suppose that  $\dim(\mathbf{x}_i)$  and  $\dim(\mathbf{s}_i)$  are small enough that  $P_0(\mathbf{x}_i)$  and  $p_0(\mathbf{s}_i)$  are easy to estimate ( $i = 1, 2, \dots, n$ ), then it remains to estimate  $p_0(\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n)$ . The original problem of estimating  $P_0(\mathbf{x})$  has been replaced by an analogous (but simpler) problem of estimating  $p_0(\mathbf{s})$  where  $\dim(\mathbf{s}) < \dim(\mathbf{x})$ . The maximum entropy procedure may be iterated to yield an estimate of  $p_0(\mathbf{s})$  itself, and so on until the dimensionalities encountered are low enough for a direct estimation of the remaining PDFs to be made. This produces a hierarchical cluster decomposition of the original  $\mathbf{x}$  because the layers of sampling functions form a tree-structure. This is a type of multi-layer neural network.

#### B. TOPOGRAPHIC SAMPLING FUNCTIONS

The main problem with this type of cluster decomposition is the selection of sampling functions. The  $\mathbf{s}_i(\mathbf{x}_i)$  must not only be good sampling functions insofar as the statistical properties of  $\mathbf{x}$  are concerned, but also the  $\mathbf{s}_i(\mathbf{x}_i)$  must stand as reduced dimension representations of the  $\mathbf{x}$ , themselves so that sampling process can be iterated.

A novel means of deriving a reduced dimension representation  $\mathbf{s}(\mathbf{x})$  of an input  $\mathbf{x}$  is to use topographic sampling functions [11]. Define a vector quantisation  $\mathbf{s}(\mathbf{x})$  of  $\mathbf{x}$  thus

$$\begin{aligned}s_0 \text{ minimises } & |\mathbf{x} - \mathbf{x}(\mathbf{s})|^2 \text{ wrt } \mathbf{s} \\ s_0 &\equiv \mathbf{s}(\mathbf{x})\end{aligned}\quad (3.4)$$

where  $\mathbf{x}(\mathbf{s})$  is a code book of quantisation vectors parameterised by  $\mathbf{s}$ . An update scheme which improves  $\mathbf{x}(\mathbf{s})$  in response to samples  $\mathbf{x}$  drawn from  $P_0(\mathbf{x})$  is

$$\mathbf{x}(\mathbf{s}) \longrightarrow \mathbf{x}(\mathbf{s}) + \epsilon(|\mathbf{s} - \mathbf{s}(\mathbf{x})|) (\mathbf{x} - \mathbf{x}(\mathbf{s})) \quad (3.5)$$

where  $\epsilon(r)$  is a non-negative monotonically decreasing function of  $r$ . The update function  $\epsilon(r)$  must have a finite width (in  $r$ ) to ensure that  $\mathbf{x}(\mathbf{s})$  is a continuous function of  $\mathbf{s}$ . The converse ( $\mathbf{s}(\mathbf{x})$  a continuous function of  $\mathbf{x}$ ) is usually not possible when  $\dim(\mathbf{s}) < \dim(\mathbf{x})$ . Physically  $\mathbf{x}(\mathbf{s})$  can be thought of as a  $\dim(\mathbf{s})$  dimensional manifold embedded in  $\dim(\mathbf{x})$  dimensions. Equation 3.5 describes the dynamical behaviour of this manifold in response to being “pulled” by  $\mathbf{x}$ : the response of the manifold is rather like that of a stiff sheet. A particularly desirable property of this learning algorithm is that  $p_0(\mathbf{s})$  tends to

become constant, thus maximising the output entropy. Furthermore, the algorithm can be shown to minimise  $\langle \log(V(\mathbf{x})) \rangle$  where  $V(\mathbf{x})$  is the error volume associated with the reconstruction of  $\mathbf{x}$  from  $\mathbf{s}$  using  $\mathbf{x}(\mathbf{s})$  [17]. It is these properties of the learning algorithm which lead to a  $\mathbf{s}(\mathbf{x})$  being called a topographic sampling function.

A hierarchy of topographic sampling functions can be derived by extending this optimisation scheme. This produces a cluster decomposed representation  $P(\mathbf{x})$  of  $P_0(\mathbf{x})$  as explained after Equation 3.3. Furthermore, various improvements can also be introduced which enormously speed up the convergence of the update scheme as originally proposed [19].

#### IV. CONCLUSIONS

There is a pressing need for representations of PDFs in situations where direct physical insight fails to pro-

vide a complete model. We have discussed two alternative techniques: MRFs and cluster decomposition. The MRF technique potentially can represent a PDF very accurately by using a sufficiently complicated set of potentials and/or hidden variables, but the computational cost of Monte Carlo simulation of MRFs can be unacceptable. The cluster decomposition technique imposes an artificial hierarchical structure on the PDF which can lead to inaccuracies in representation, but it involves no Monte Carlo simulations. For real time applications we recommend the use of cluster decomposition.

- 
- [1] D H Ackley, G E Hinton, and T J Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Science **9** (1985), no. 2, 147–169.
  - [2] J Besag, *Spatial interaction and the statistical analysis of lattice systems*, Journal of the Royal Statistical Society: Series B **36** (1974), no. 2, 192–236.
  - [3] M H DeGroot, *Optimal statistical decisions*, McGraw-Hill, New York, 1970.
  - [4] S Geman and D Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence **6** (1984), no. 6, 721–741.
  - [5] S Geman and C Graffigne, *Markov random field image models and their applications to computer vision*, Proceedings of the international congress of mathematicians (Providence) (A M Gleason, ed.), American Mathematical Society, 1986, pp. 1496–1517.
  - [6] E T Jaynes, *Information theory and statistical mechanics*, Physical Review **106** (1957), no. 4, 620–630.
  - [7] ———, *Information theory and statistical mechanics II*, Physical Review **108** (1957), no. 2, 171–190.
  - [8] ———, *Prior probabilities*, IEEE Transactions on Systems Science and Cybernetics **4** (1968), no. 3, 227–241.
  - [9] ———, *On the rationale of maximum entropy methods*, Proceedings of the IEEE **70** (1982), no. 9, 939–952.
  - [10] R Kindermann and J L Snell, *Markov random fields and their applications*, Contemporary Mathematics, vol. 1, American Mathematical Society, Providence, 1980.
  - [11] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
  - [12] S P Luttrell, *The implications of Boltzmann-type machines for SAR data processing: a preliminary survey*, Memorandum 3815, Royal Signals and Radar Establishment, Malvern, 1985.
  - [13] ———, *Markov random fields: a strategy for clutter modelling*, Proceedings of AGARD conference on scattering and propagation in random media (Rome), AGARD, 1987, pp. 7.1–7.8.
  - [14] ———, *Radar 87*, ch. Designing Markov random field structures for clutter modelling, pp. 222–226, IEE, London, 1987.
  - [15] ———, *The use of Markov random field models in sampling scheme design*, Proceedings of SPIE international symposium on inverse problems in optics (New York), SPIE, 1987.
  - [16] ———, *The use of Markov random field models to derive sampling schemes for inverse texture problems*, Inverse Problems **3** (1987), no. 2, 289–300.
  - [17] ———, *Image compression using a neural network*, Proceedings of IGARSS conference on remote sensing (Edinburgh), ESA, 1988, pp. 1231–1238.
  - [18] ———, *A maximum entropy approach to sampling function design*, Inverse Problems **4** (1988), no. 3, 829–841.
  - [19] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
  - [20] N Metropolis, A W Rosenbluth, M N Rosenbluth, A H Teller, and E Teller, *Equations of state calculations by fast computing machines*, Journal of Chemical Physics **21** (1953), no. 6, 1087–1092.
  - [21] C J Preston, *Gibbs states on countable sets*, University Press, Cambridge, 1974.
  - [22] T J Sejnowski, *Higher order Boltzmann machines*, Proceedings of conference on neural networks for computing (Snowbird), vol. 151, American Institute of Physics, pp. 398–403.

# INFERENCE THEORY \*

S. Luttrell  
*RSRE, Malvern, Worcs.*

**ABSTRACT:** This transcript reviews the need for low-level stochastic image models, especially for coherent images which are corrupted by speckle noise. A unified approach to inference in such models using Bayes' rule and Shannon's information theory is presented. In the gaussian PDF approximation exact results for sampling scheme optimisation and for inference from sample values are derived (super-resolution in particular is concentrated on). For non-gaussian PDFs the method of graphs for constructive PDF generation is outlined, and several research trends are indicated.

## I. INTRODUCTION

I shall assume throughout this paper that the processes that generate a dataset can be represented as a graph.

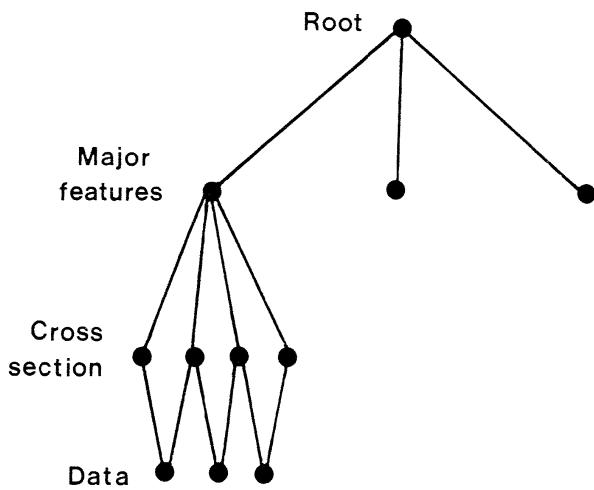


Figure 1:

The essential components of such a representation are  
(1) Nodes  $\equiv$  physical variables (e.g. elements of cross section), and  
(2) Links  $\equiv$  (mutual) interactions or dependencies amongst the variables.

Such image generating graphs can usually be divided into two components which are distinguished by the degree of randomness of their internal interactions: low level components of the graph (e.g. individual pixel variables) typically have noisy mutual dependencies, whereas high level components (e.g. regions and boundaries) typically have more structured mutual dependencies.

Statistical information theory provides a unified framework for studying systems that are characterised by probability density functions (PDFs), so I shall use it to analyse low level graphs. This work is particularly important

for the modelling of coherent images which all suffer from speckle noise.

## II. TRAINING AND BAYESIAN INFERENCE

I shall develop information theory by appealing to a standard experiment as in Figure 2.

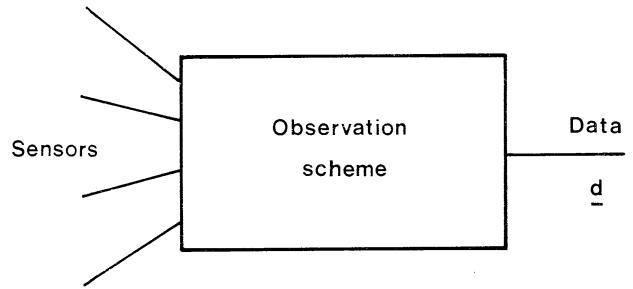


Figure 2:

Each observation on the system  $S$  produces a data vector  $d^{(j)}$  which is recorded. It is very convenient to gather together all the  $d^{(j)}$  into a dataset  $D$ , where

$$D \equiv \{ \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(N)} \} \quad (1)$$

for a set of  $N$  observations. Both changes in the state of  $S$  and in the measurement error cause the value of  $\mathbf{d}$  to vary. It is convenient to plot each  $\mathbf{d}^{(j)}$  in a data space, and it is appealing to represent the resulting scattergram as in Figure 3.

The scattergram does not record the order in which the  $\mathbf{d}^{(j)}$  were observed, so time dependencies in the variation of  $\mathbf{d}$  have been lost. This limitation will apply throughout this paper, although it can be circumvented by regarding the whole set of  $\mathbf{d}^{(j)}$  as a “super data vector” which is plotted in a “super data space”.

The above procedure is an unsupervised training procedure where all the  $\mathbf{d}^{(j)}$  are retained but their order is discarded. I have not yet introduced a separate means whereby the state of  $S$  can be observed directly so as to provide a class label for the corresponding  $\mathbf{d}^{(j)}$ . The

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 11, 2019.

This appeared in Mathematics in Remote Sensing, Clarendon Press, S. R. Brooks (ed.), 205-222, 1989. © Controller, Her Majesty's Stationery Office, London 1987.

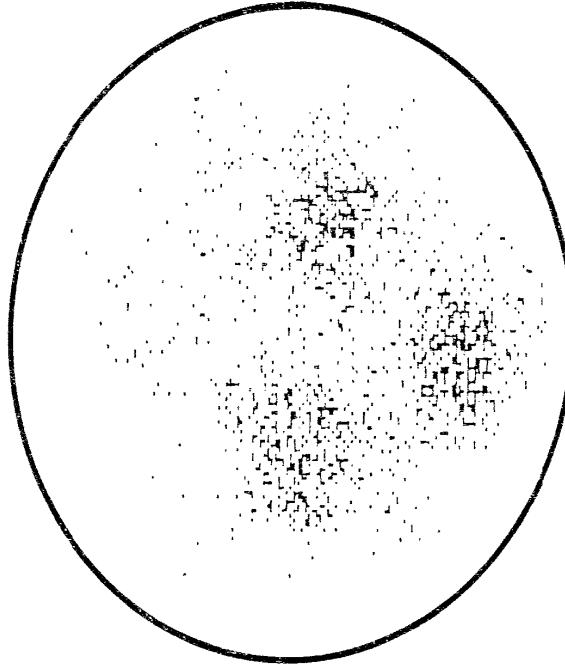


Figure 3:

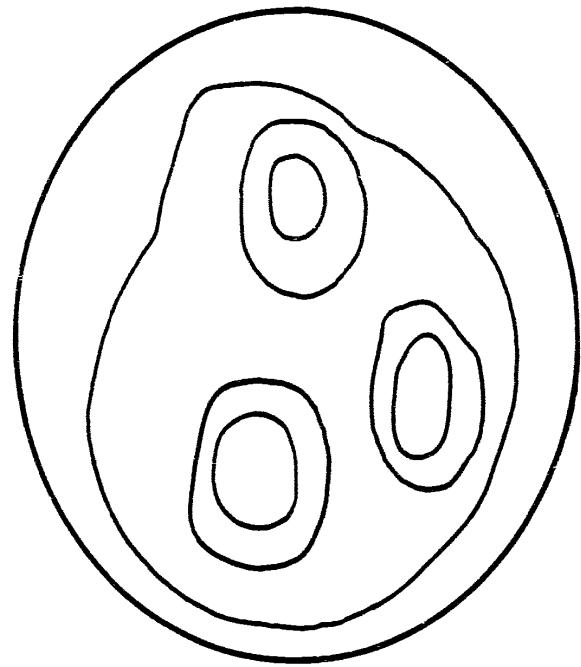


Figure 4:

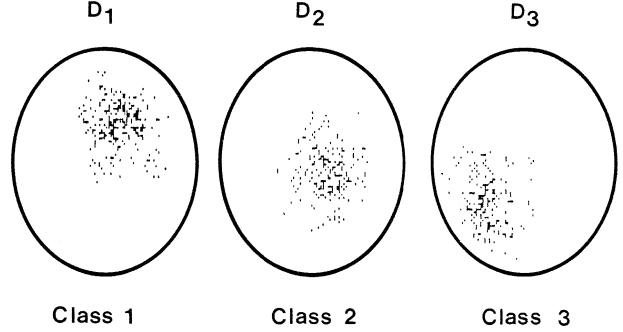


Figure 5:

scattergram is therefore a mixture of the scattergrams that would be obtained for the separate classes.

As the number of observations  $N$  increases the “cloud” of plotted observations becomes more dense, and it provides a better representation of the distribution of  $\mathbf{d}$ . In the limit  $N \rightarrow \infty$  the local density of the scattergram can be used to define a probability density function (PDF) in the usual way, and I shall use the generic notation “ $P(x)$ ” to denote “PDF of  $x$ ”. Clearly this limit is difficult to realise in practice, so the scattergram is usually used to estimate only a parametric fit to the PDF (e.g. gaussian PDF). This process of estimating a PDF from a finite set of observations is not part of information theory *per se*; it belongs to approximation theory which I shall not dwell upon here. The PDF that is obtained from the mixture scattergram is denoted by  $P(\mathbf{d})$  and is called the mixture PDF; this is shown schematically in Figure 4.

$P(\mathbf{d})$  can be decomposed “by eye” into a number of components (or modes), which can then be arbitrarily assigned class labels. Because the training procedure is unsupervised these class labels will not have a known relationship to the state  $S$ , and so it is very difficult to make any further justifiable analysis of inferences that can be drawn about  $S$  given  $\mathbf{d}$ .

I shall assume now that the observations are accompanied by some collateral information (ground truth) derived directly from  $S$ ; this is supervised training. This additional “conditioning signal” comprises a class label  $c^{(j)}$  that accompanies each  $\mathbf{d}^{(j)}$  thus augmenting  $\mathbf{d}$  to  $(c, \mathbf{d})$ . Obviously a scattergram can be constructed from

$(c, \mathbf{d})$  just as it was for  $\mathbf{d}$ ; this is equivalent to building a separate scattergram for each value of  $c$ .  $K$  datasets  $D_1, \dots, D_K$  (each as in Equation 1 are constructed when there are  $K$  classes, and each dataset  $D_j$  contains only those  $\mathbf{d}$  that derive from class  $c_j$  as shown in Figure 5 for the case  $K = 3$ . Each scattergram is used to estimate a class conditional PDF  $P(\mathbf{d}|c)$ , and the total number of points in each scattergram, is used to estimate the *a priori* PDF  $P(c)$ . Obviously

$$P(\mathbf{d}|c) P(c) = P(c, \mathbf{d}) \quad (2)$$

where the joint PDF  $P(c, \mathbf{d})$  is the estimate that would have been obtained from a scattergram consisting of plots of the  $(c, \mathbf{d})$ .

The above separation into scattergrams labelled by class  $c$  is not the only such separation that can be made.

Another separation is into scattergrams labelled by  $\mathbf{d}$  which leads to the data conditional PDF  $P(c|\mathbf{d})$  and the mixture PDF  $P(\mathbf{d})$ , with the relationships

$$P(c|\mathbf{d}) P(\mathbf{d}) = P(c, \mathbf{d}) \quad (3)$$

Comparing Equation 2 and Equation 3 leads to Bayes' rule

$$P(c|\mathbf{d}) = \frac{P(\mathbf{d}|c) P(c)}{P(\mathbf{d})} \quad (4)$$

All the quantities in Bayes' rule can be estimated from the various scattergrams that I referred to above.

The data conditional PDF  $P(c|\mathbf{d})$  is also called the *a posteriori* PDF, because it is the result of combining the *a priori* PDF  $P(c)$  with the class conditional PDF  $P(\mathbf{d}|c)$ . Loosely speaking, prior knowledge  $P(c)$  that was gleaned from a supervised training procedure is combined with information  $P(\mathbf{d}|c)$  that is derived each time an observation  $\mathbf{d}$  is made. It is very important to distinguish between the original training procedure and the Bayesian inference procedures: training measures PDFs directly by accumulating a large number of class labelled observations, whereas inference from a single unlabelled observation is the act of choosing particular conditional PDF from the trained PDFs (see Equation 4).

### III. AN INFORMATION MEASURE

I wish to derive a single measure of the information content of a PDF in order to make clear which properties of a particular sampling scheme and inference process convey useful information. In order to derive a single measure from a complicated PDF I must first of all reduce the PDF to a simpler form. I shall develop the argument in terms of  $P(c)$  although it can trivially be carried over to all other PDFs. Recall that  $P(c)$  was the limiting frequency of occurrence of each class  $c$ , so an operational definition of an information measure must refer to long sequences of class labels generated by  $S$ : this is the key to information theory.

Consider an  $N$  sample sequence ( $N$ -sequence) in the  $K$  class case (as above). Each sample is one of  $K$  possible classes, so the total number of possible  $N$ -sequences is given by

$$W_{\text{TOT}} = K^N \quad (5)$$

However these sequences do not all have the same probability; some are overwhelmingly more probable than others when  $N$  is large. Clearly as  $N \rightarrow \infty$  the number of times that class  $c$  occurs in the sequence is given by  $P(c)N$  - this is the definition of  $P(c)$  after all! For finite  $N$  there is a distribution about the value  $P(c)N$ , but this becomes insignificant for large  $N$ . The effective number of  $N$ -sequences is therefore given by

$$W_{\text{EFF}} = \frac{N!}{[P(c_1)N]! \cdots [P(c_K)N]!} \quad (6)$$

This expression can be simplified by using Stirling's approximation ( $\log(n!) \sim n \log(n) - n$ ) yielding

$$W_{\text{EFF}} = \exp \left[ -N \sum_c P(c) \log[P(c)] \right] \quad (7)$$

The argument of the exponential contains the entropy  $H(c)$  of  $P(c)$

$$H(c) \equiv - \sum_c P(c) \log[P(c)] \quad (8)$$

so that

$$W_{\text{EFF}} = \exp[NH(c)] \quad (9)$$

It is easy to verify the inequality

$$0 \leq W_{\text{EFF}} \leq W_{\text{TOT}} \quad (10)$$

with equality  $W_{\text{EFF}} = W_{\text{TOT}}$  if and only if all classes are equally likely. The base of the logarithm is unimportant; logarithms to base  $x$  can be used and compensated for by the replacement  $\exp \rightarrow x$ . In particular if *logarithms to base K* are used then

$$W_{\text{EFF}} = K^{NH(c)} \quad (11)$$

which is compared with Equation 5 in order to define

$$K_{\text{EFF}} \equiv K^{H(c)} \quad (12)$$

By the same reasoning that led to Equation 10 it follows that

$$0 \leq K_{\text{EFF}} \leq K \quad (13)$$

The steps of the argument leading to  $W_{\text{TOT}}$  and  $W_{\text{EFF}}$  are presented pictorially in Figure 6.

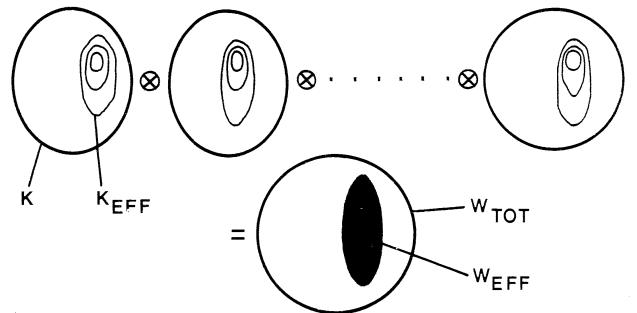


Figure 6:

The effective number of classes  $K_{\text{EFF}}$  that  $P(c)$  generates when  $N$ -sequences are considered is a natural measure to use when defining the information content of  $P(c)$ . From Equation 12 it is clear that the entropy  $H(c)$  conveys the uncertainty associated with a PDF  $H(c)$ . The smaller the entropy, the smaller the value of  $K_{\text{EFF}}$ , and therefore the smaller the uncertainty about what class  $c$  will be sampled next from  $P(c)$ .

#### IV. MUTUAL INFORMATION

The operation of Bayes' rule (Equation 4) is shown in Figure 7.

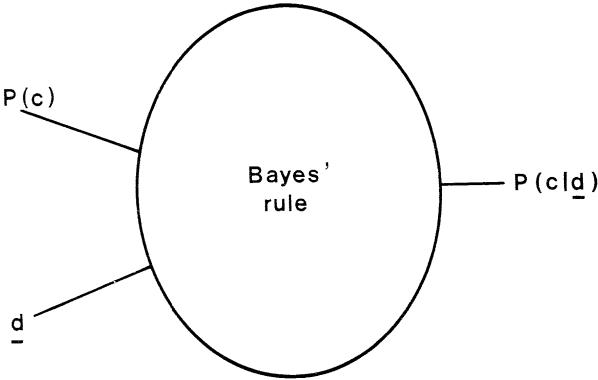


Figure 7:

It is important to define a measure of information which quantifies the usefulness of the data  $\mathbf{d}$  in generating the *a posteriori* PDF  $P(c|\mathbf{d})$  from the *a priori* PDF  $P(c)$ . An obvious measure to use is

$$I(c, \mathbf{d}) \equiv H(c) - H(c|\mathbf{d}) \quad (14)$$

where

$$H(c|\mathbf{d}) \equiv \sum_{\mathbf{d}} P(\mathbf{d}) \left[ - \sum_c P(c|\mathbf{d}) \log[P(c|\mathbf{d})] \right] \quad (15)$$

and the base of the logarithm used in  $H(c)$  and in  $H(c|\mathbf{d})$  must be the same.  $I(c, \mathbf{d})$  measures the reduction in uncertainty associated with the *a priori* PDF  $P(c)$  when data  $\mathbf{d}$  are observed, averaged over observations.  $I(c, \mathbf{d})$  is called transinformation or mutual information, and it provides an information theoretic measure of the usefulness of an observation scheme in collecting data that is suitable for drawing inferences about class membership.

#### V. GAUSSIAN PROBABILITY DENSITY FUNCTIONS

I shall now derive a closed form expression for  $I(c, \mathbf{d})$  by assuming that  $P(c)$  and  $P(c|\mathbf{d})$  are gaussian. The class label  $c$  must therefore be continuous, and I shall extend it further to be multi-dimensional (i.e.  $c \rightarrow \mathbf{c}$ ). Defining the *a priori* and *a posteriori* covariance matrices as  $C_{\text{PRIOR}}$  and  $C_{\text{POST}}$  respectively, after some algebra I obtain the result

$$I(\mathbf{c}, \mathbf{d}) = \log \left[ \frac{\det[C_{\text{PRIOR}}]}{\det[C_{\text{POST}}]} \right] \quad (16)$$

A determinant of a covariance matrix is the product of its eigenvalues, and so it measures the “volume of uncertainty” that is associated with the corresponding gaussian

distribution. The quantity  $\exp[I(\mathbf{c}, \mathbf{d})]$  (note the equality of logarithm base and exponentiating base) therefore measures the ratio of the effective number of classes associated with  $P(\mathbf{c})$  to the average of the effective number of classes associated with  $P(\mathbf{c}|\mathbf{d})$ . Clearly this corresponds to our intuitive notion of the amount of useful information contained in  $\mathbf{d}$  on average.

An important caveat is that only those class labels which are of interest should be included in  $P(\mathbf{c})$ . All labels contribute to the value of  $I(\mathbf{c}, \mathbf{d})$ , and so to obtain a measure of the relevant part of the mutual information the irrelevant class labels should be excluded. This process should be carried even further so that classes that are equivalent (for one's particular purposes) are combined into a single class. Clearly the value of  $I(\mathbf{c}, \mathbf{d})$  that is calculated will depend strongly on the particular interests and requirements of the observer, as expected.

#### VI. SAMPLING SCHEME OPTIMISATION

I shall assume that the classes have been regrouped as I discussed above. The value of  $I(\mathbf{c}, \mathbf{d})$  then indicates how much useful information a particular observation (or sampling) scheme provides on average for resolving the uncertainty about which class is present. For a particular problem the sampling scheme can be altered so as to explore variations in the value of  $I(\mathbf{c}, \mathbf{d})$ , and to optimise the sampling scheme by maximising  $I(\mathbf{c}, \mathbf{d})$ .

All the results that I have calculated [4, 6] agree with one's intuition about what a good sampling scheme should look like. When the number of sample points is fixed the samples should not be so close together that they measure essentially the same information, and they should not be so far apart that they undersample the data space. Mutual information is an information theoretic measure that allows one to trade off between these two extreme cases optimally. In particular I have derived Nyquist sampling of bandlimited functions, exponential sampling of Laplace transforms, and prior knowledge dependence of optimal sampling. Mutual information provides a rigorous means of quantifying the influence of prior knowledge on the design of sampling schemes, and it produces the expected results in simple cases.

#### VII. INDEPENDENT INFORMATION CHANNELS

The expression for  $I(\mathbf{c}, \mathbf{d})$  in Equation 16 is simplified by simultaneous diagonalisation of the two covariance matrices  $C_{\text{PRIOR}}$  and  $C_{\text{POST}}$ . This leads to

$$I(\mathbf{c}, \mathbf{d}) = \sum_j I_j \quad (17)$$

where

$$I_j \equiv \log \left[ \frac{\lambda_{\text{PRIOR},j}}{\lambda_{\text{POST},j}} \right] \quad (18)$$

and I use  $\lambda$  generically to denote an eigenvalue. The fact that in Equation 17 the total mutual information  $I(\mathbf{c}, \mathbf{d})$  is decomposed into a sum over components  $I_j$  means that the eigenvectors which simultaneously diagonalise Equation 16 must correspond to independent information channels, each carrying an independent piece of useful information about  $\mathbf{c}$ .

A simple interpretation of this decomposition is that the eigenvectors form a feature set for separating the classes  $\mathbf{c}$  optimally. They may also be identified with the Karhunen-Loëve expansion of a covariance matrix.

### VIII. SUPER-RESOLUTION

I have related the above decomposition to super-resolution [5, 7] of complex (coherent) images. I model a linear imaging system of this type by

$$\mathbf{d} = T \mathbf{f} + \mathbf{n} \quad (19)$$

Defining a gaussian *a priori* PDF over scattered fields  $\mathbf{f}$

$$P(\mathbf{f}) \propto \exp[-\mathbf{f}^\dagger W^{-1} \mathbf{f}] \quad (20)$$

where  $W$  is a covariance matrix that expresses the prior knowledge that  $\langle f_j \rangle = 0$  and that  $\langle f_j f_k^* \rangle = W_{j,k}$  ( $\langle \dots \rangle$  denotes ensemble average). There is no factor  $1/2$  in the argument of the exponential in Equation 20 because the field  $\mathbf{f}$  is complex (not real). The imaging operator  $T$  is linear and the (additive) noise  $\mathbf{n}$  is modelled also using a gaussian PDF

$$P(\mathbf{n}) \propto \exp[-\mathbf{n}^\dagger N \mathbf{n}] \quad (21)$$

where usually  $N$  is assumed to be proportional to the identity matrix (i.e. white noise).

Using the PDFs defined above a little algebra yields

$$P(\mathbf{f}|\mathbf{d}) \propto \exp[-(\mathbf{f} - \mathbf{f}_0)^\dagger M^{-1} (\mathbf{f} - \mathbf{f}_0)] \quad (22)$$

where

$$M \equiv T^\dagger N^{-1} T + W^{-1} \quad (23)$$

and

$$\begin{aligned} \mathbf{f}_0 &\equiv [T^\dagger N^{-1} T + W^{-1}] T^\dagger \mathbf{d} \\ &= W T^\dagger [T W T^\dagger + N]^{-1} \mathbf{d} \end{aligned} \quad (24)$$

Both the *a priori* PDF  $P(\mathbf{f})$  and the *a posteriori* PDF  $P(\mathbf{f}|\mathbf{d})$  are gaussian, so Equation 16 may be used with the replacement  $\mathbf{c} \rightarrow \mathbf{f}$ .

The simultaneous eigensystem of  $W$  and  $M$  provides the inputs of the set of independent information channels, which may be transformed using  $T$  to produce their outputs in data space; the details of this calculation are contained in [7]. These information channels can be used

to invert the data  $\mathbf{d}$ , and to obtain an estimate or reconstruction of the scattered field  $\mathbf{f}$ . The maximum *a posteriori* probability (MAP)  $\mathbf{f}$  is clearly  $\mathbf{f}_0$ , and I have shown in [7] that  $\mathbf{f}_0$  consists of components that are obtained by independent inversion in each information channel. Note that there is no problem of sensitivity to data noise because such effects have already been modelled in  $P(\mathbf{n})$ .

The effect of such an inversion is to reorganise and interpolate the data (by using information channels) in a manner that is dependent on the form of  $P(\mathbf{f})$ . In particular, if  $W$  expresses the prior knowledge that  $\mathbf{f}$  has large components concentrated only in a small region of order of the resolution area of  $T$ , then such an inversion can lead to super-resolution [5, 7].

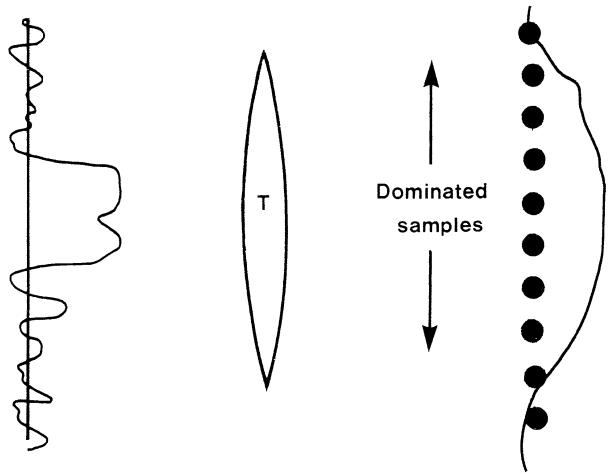


Figure 8:

Physically the explanation of super-resolution is simple. With reference to Figure 8 if a small region of the scattered field has larger components than elsewhere, then the fraction of the data space which is dominated by contributions from this region is larger than average; thus a greater than average number of samples contains information pertaining to the form of the large amplitude scattered field, and this leads to the possibility of super-resolution. By a simple extension of this argument the degree of super-resolution that is obtainable must decrease either as the size of the large amplitudes decreases and/or as the competition amongst the parts of the scattered field for regions of data space increases.

### IX. CONSTRUCTIVE DEFINITION OF NON-GAUSSIAN PDFS

I have presented an extensive analysis of the gaussian PDF case. However it is not always realistic to describe prior knowledge in terms of gaussian PDFs, so I shall now turn to the more difficult non-gaussian case.

Because analytic results are now difficult to obtain I shall adopt the constructive approach to defining PDFs.

Thus I will not write down an analytic expression for  $P(\mathbf{c})$ , rather I shall define a set of rules for generation of  $P(\mathbf{c})$ . I hope that this approach will permit me to define a richer structure of PDFs than is possible using the direct analytic approach, and furthermore that the constructive definition of the PDF will allow me to generate samples from  $P(\mathbf{c})$  with ease. Both of these advantages will turn out to be true. There are two essential components involved in constructively defining a PDF which were defined in the introduction; nodes and links form the skeletal model of a PDF.

A very simple example of such a model is a first order Markov chain as shown in Figure 9.

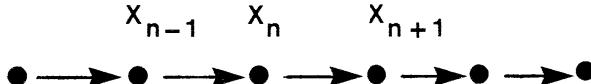


Figure 9:

The joint PDF  $P(x)$  over the variables  $x_j$  is generated by developing the chain using the update rule (or stochastic matrix)

$$x_{j+1} = a(x_j) x_j + b(x_j) \nu_j \quad (25)$$

where the  $\nu_j$  are usually generated by a white (gaussian) noise process. Clearly if  $b(x_j) = 0 \forall x_j$ , then the relationship between the  $x_j$  is deterministic. Usually both terms in Equation 25 contribute in which case the  $x_j$  have mutual statistical dependency (i.e. a PDF  $P(\mathbf{x})$  describes their properties).

Another example that is more closely related to the problem of modelling images is shown in Figure 10.

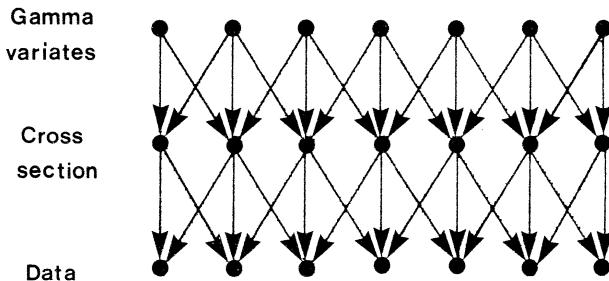


Figure 10:

This depicts (schematically) as nodes and links the clutter model that is discussed by Oliver in his contribution to these proceedings. The meaning of the diagram is self evident; it should be interpreted as an information flow diagram.

## X. GRAPHS, TREES AND TEXTURE MODELS

I shall now analyse more closely the type of information that is encoded into a particular structure of nodes and links. Such structures are called graphs, and their study forms the field of graph theory. It is not necessary to get involved with the rigours of graph theory here, because I intend to give a heuristic treatment of PDF generation models.

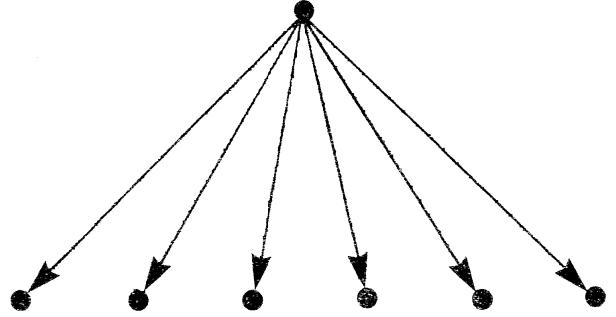


Figure 11:

I show a simple type of graph in Figure 11 where a parent node has several children; this is called a tree graph for obvious reasons. The children naturally have mutual dependencies because they have a common parent. A graph of this type can describe very long range mutual dependencies in images. In particular it can be used to describe a segment of an image, where the parent node carries information relating to the segment properties and the children each correspond to a single pixel value.

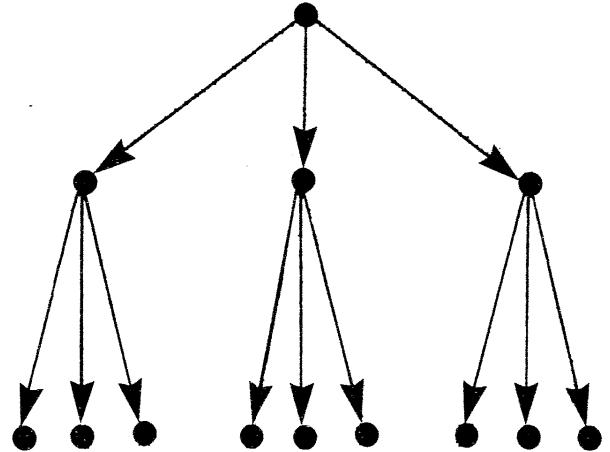


Figure 12:

I show a more sophisticated graph in Figure 12 which has grandchildren. As in Figure 11 the children have mutually dependent properties because of the common

parent, but now each set of grandchildren (of one particular child) have mutually dependent properties because of the common child. There are thus two levels of mutual dependency; one affects the mutual dependency amongst a single set of grandchildren (short range dependency), and the other affects the mutual dependency amongst different sets of grandchildren (long range dependency). Loosely speaking there are two correlation scales in the grandchildren which are naturally described by a tree graph. In terms of images such as a graph could not only describe a segment, but also clustering within a segment (sub-segments).

I show a type of graph that might be used to model texture in an image in Figure 13.

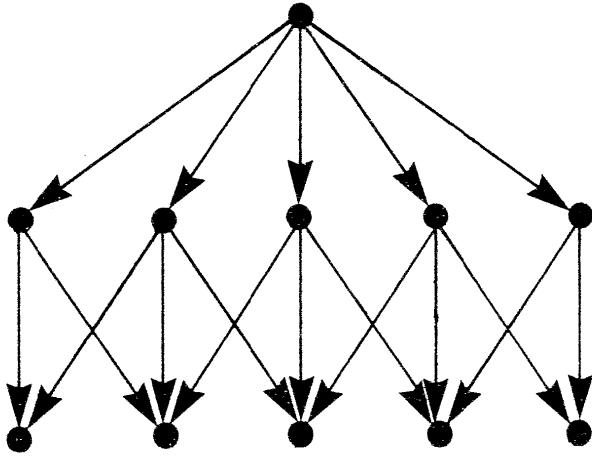


Figure 13:

This differs from Figure 12 only insofar as each grandchild belongs to more than one family! This is an important difference for image modelling because it means that the sub-segments of Figure 12 are now permitted to overlap. This is necessary if a translation invariant (or homogeneous) texture model is to be built. Because of this overlap the graph of Figure 13 is not treelike. Its structure gives rise to two length scales; segment size and texture coarseness size.

The graphs in Figure 11, Figure 12, and Figure 13 are causal graphs because the flow of dependency is in one direction only. The overall dependencies can be written down as nested conditional PDFs by using the arrows on the links of the graph to identify where there is dependency. Causal graphs are very simple because there is no feedback, and therefore there is no danger of any contradictions arising between the states of the variables at the nodes. Monte Carlo methods may be used to generate samples from the PDF that is constructively generated by each causal graph. This solves the direct problem of generating data from an underlying model.

## XI. BAYESIAN INFERENCE USING GRAPHS

The inverse problem (or inference process) of regenerating the underlying variables of a model from data can also be solved within the framework of graphs. This corresponds to clamping the data variables (e.g. grandchildren) with their observed values, and then deducing the distribution of states that the remaining variables (e.g. parent and children) should take. However the general causal graph model above does not provide a simple prescription for inferring the *a posteriori* PDF.

A solution to this problem is obtained if a bidirectional flow of dependency is permitted in the graph model. A simple way of achieving this is to define the joint PDF over all the variables of the graph as a Gibbs distribution [3]. Conventionally this is written in the form

$$P(\mathbf{x}) \propto \exp[-U(\mathbf{x})] \quad (26)$$

where  $U(\mathbf{x})$  is a sum of terms each of which involves only a few components of  $\mathbf{x}$ . The form of  $P(\mathbf{x})$  is therefore a product of component factors, which I prefer to emphasise by writing

$$P(\mathbf{x}) \propto p_1(\mathbf{x}) \cdots p_R(\mathbf{x}) \quad (27)$$

where each factor depends on only a few components of  $\mathbf{x}$ . The model that is so constructed is difficult to make exactly equivalent to a causal graph model, because it is necessary to balance delicately the reverse flow of dependency if the net forward flow is to be the same as in the causal graph model. However the Gibbs distribution form of  $P(\mathbf{x})$  (and Equation 27) is very useful because it guarantees that the model has no internal inconsistencies [1].

The Metropolis algorithm [8] may be used to generate samples from  $P(\mathbf{x})$ , and in particular both the direct and inverse problems may be solved. The class of the system  $S$  is  $\mathbf{c}$ , the observed data is  $\mathbf{d}$ , and any other relevant variables are  $\mathbf{h}$ , so  $P(\mathbf{x})$  is replaced by  $P(\mathbf{c}, \mathbf{h}, \mathbf{d})$ . The direct problem involves an unconstrained use of the Metropolis algorithm to generate samples  $(\mathbf{c}, \mathbf{h}, \mathbf{d})$ , where only  $\mathbf{d}$  is of interest for unsupervised training, and where  $(\mathbf{c}, \mathbf{d})$  is of interest for supervised training. The inverse problem involves clamping  $\mathbf{d}$  with some data, and then using the Metropolis algorithm to generate samples  $(\mathbf{c}, \mathbf{h})$ ; this generates  $P(\mathbf{c}, \mathbf{h}|\mathbf{d})$  and hence  $P(\mathbf{c}|\mathbf{d})$ .

The above means of generating samples from the *a posteriori* PDF can be extended so that the MAP solution is obtained; this is called optimisation by simulated annealing (OSA). The essence of the method is that as samples from the *a posteriori* PDF are generated by the Metropolis algorithm, the form of the Gibbs distribution should be slowly altered so as to accentuate the differences in probability between different states. In practice this is achieved by the transformation  $U(\mathbf{x}) \rightarrow \frac{U(\mathbf{x})}{T}$  with  $T \rightarrow 0$  in Equation 26. The parameter  $T$  is analogous to a temperature in statistical thermodynamics, and the way in which it approaches zero is called the

annealing schedule. Clearly the MAP solution has the smallest value of  $U(\mathbf{x})$  and so becomes overwhelmingly likely to be sampled as  $T \rightarrow 0$ . An interesting example of this method is contained in [2] and I recommend that that paper be read by anyone who is interested in OSA in image processing.

There is a pitfall in the above OSA method; the annealing schedule has to be exceedingly slow in order to guarantee convergence to the MAP solution in some cases. Also it can lead to very long computer run times as the Metropolis algorithm has to generate many samples at each stage of the annealing schedule in order to guarantee that the graph has equilibrated. However when OSA is used carefully on a parallel computer architecture it is a powerful tool.

## XII. THE WAY AHEAD

In my opinion there is a trade-off between the computational expense (in time units) of an inference method and the quality of the inference, where I assume throughout that the degree of parallelism in the method is held constant. The Gibbs distribution method (with or without OSA) has highly desirable properties, but it consumes large resources. I believe that we could speed up

the inference process by incorporating heuristics or non-Bayesian methods, although we would probably sacrifice the ability to identify an *a posteriori* PDF.

A powerful argument against the Bayesian approach is that it relies upon the existence of PDFs. However I introduced these only as a means of representing the form of a scattergram after accumulating a large number (infinite number, strictly) of training sets. In practice the number of training sets available is usually insufficient to produce a densely populated scattergram, and so it is not obvious that a PDF is the best representation.

It seems therefore that the resolution of the computational cost of the inference problem by sacrificing the Bayesian approach and adopting heuristics is intimately bound up with the choice of representation that was chosen for recording the observations during the original training process. In short, training and inference are not really separate processes at all!

Of course the artificial intelligence (AI) community has adopted a heuristic strategy all along. However AI has concerned itself principally with the higher level variables and relationships within a graph. There is no analysis which deals with the whole of a graph including its low level (possibly highly stochastic) variables. There is a clear opportunity for much original research in this area.

- [1] J Besag, *Spatial interaction and the statistical analysis of lattice systems*, Journal of the Royal Statistical Society: Series B **36** (1974), no. 2, 192–236.
- [2] S Geman and D Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Machine Intelligence **6** (1984), no. 6, 721–741.
- [3] R Kindermann and J L Snell, *Markov random fields and their applications*, Contemporary Mathematics, vol. 1, American Mathematical Society, Providence, 1980.
- [4] S P Luttrell, *A new method of sample optimisation*, Optica Acta **32** (1985), no. 3, 255–257.
- [5] ———, *Prior knowledge and object reconstruction using the best linear estimate technique*, Optica Acta **32** (1985), no. 6, 703–716.
- [6] ———, *The use of transinformation in the design of data sampling schemes for inverse problems*, Inverse Problems **1** (1985), no. 3, 199–218.
- [7] S P Luttrell and C J Oliver, *Prior knowledge in synthetic aperture radar processing*, Journal of Physics D: Applied Physics **19** (1986), no. 3, 333–356.
- [8] N Metropolis, A W Rosenbluth, M N Rosenbluth, A H Teller, and E Teller, *Equations of state calculations by fast computing machines*, Journal of Chemical Physics **21** (1953), no. 6, 1087–1092.

# Self-Organisation: A Derivation from First Principles of a Class of Learning Algorithms \*

S P Luttrell

*Royal Signals and Radar Establishment, St Andrews Road, Malvern, WORCS, WR14 3PS, UK*

We present a novel derivation of Kohonen's topographic mapping learning algorithm. Thus we prescribe a vector quantiser by minimising an a reconstruction distortion measure. We include in this distortion a contribution from the code noise which corrupts the output of the vector quantiser. Such code noise models the expected distorting effect of later stages of processing, and thus provides a convenient way of ensuring that the vector quantiser acquires a useful coding scheme. The neighbourhood updating scheme of Kohonen's self-organising neural network emerges as a special case of this code noise model. This reformulation of Kohonen's algorithm provides a simple interpretation of the role of the neighbourhood update scheme which is used.

## I. INTRODUCTION

Vector quantisation theory is a generalisation of scalar quantisation theory as expressed by the Lloyd-Max equations [3, 8]. The generalisation of the Lloyd-Max equations to vector quantisation is straightforward, and the LBG, or k-means, algorithm, [2] provides a means of adjusting the code vectors to locate a local minimum of a distortion measure. The LBG algorithm with an  $L_2$  distortion measure is the batch training version of Kohonen's vector quantisation algorithm with zero neighbourhood size [1].

## II. THEORY

### A. The LBG algorithm

Although it is not strictly necessary, we shall restrict our attention to an  $L_2$  distortion measure  $d(\mathbf{x}, \mathbf{x}')$  defined as

$$d(\mathbf{x}, \mathbf{x}') \equiv \|\mathbf{x}' - \mathbf{x}\|^2 \quad (2.1)$$

which measures the Euclidean distance between a vector  $\mathbf{x}$  and its reconstruction after vector quantisation  $\mathbf{x}'$ . We shall be concerned with the average of  $d(\mathbf{x}, \mathbf{x}')$  over a training set of vectors  $\mathbf{x}$ , so we shall introduce the probability density function (PDF)  $P(\mathbf{x})$  over samples  $\mathbf{x}$  selected at random from the training set.

Denote the encoding operation as  $y(\mathbf{x})$ , and the corresponding decoding operation as  $\mathbf{x}'(y)$ . The average  $L_2$  distortion  $D$  is then given by the average of  $d(\mathbf{x}, \mathbf{x}'(y(\mathbf{x})))$  over  $\mathbf{x}$

$$D \equiv \int d\mathbf{x} P(\mathbf{x}) \|\mathbf{x}'(y(\mathbf{x})) - \mathbf{x}\|^2 \quad (2.2)$$

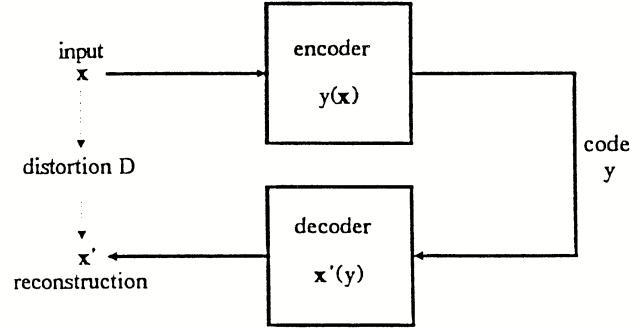


Figure 1: Representation of Equation 2.2 describing the  $L_2$  distortion  $D$  after encoding using  $y(\mathbf{x})$  and then decoding using  $\mathbf{x}'(y)$ . This is a single stage encoder.

Equation 2.2 is depicted schematically in Figure 1, where  $\mathbf{x}'(y)$  acts as a pseudo-inverse for  $y(\mathbf{x})$ . The optimum encoding/decoding scheme is found by appropriately varying the functions  $y(\mathbf{x})$  and  $\mathbf{x}'(y)$  so as to minimise  $D$ . Necessary conditions for minimisation of  $D$  are [2]:

A1. Given  $\mathbf{x}$ , the appropriate code  $y = y(\mathbf{x})$  minimises the distortion  $\|\mathbf{x}'(y(\mathbf{x})) - \mathbf{x}\|^2$ .

A2. Given  $y$ , the appropriate reconstruction  $\mathbf{x}' = \mathbf{x}'(y)$  is the centroid of those  $\mathbf{x}$  which satisfy  $y = y(\mathbf{x})$ .

Note that A1 is a nearest neighbour encoding rule. A1 and A2 imply that  $D$  is stationary (and locally minimum) with respect to variations of  $y(\mathbf{x})$  and  $\mathbf{x}'(y)$ , respectively. Such conditions are therefore sufficient for a local minimum of  $D$ , but are only necessary for a global minimum. The LBG algorithm runs in batch training mode and consists of alternately adjusting  $y(\mathbf{x})$  according to A1, and then adjusting  $\mathbf{x}'(y)$  according to A2.

### B. Two stage quantisation

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 11, 2019.

This appeared in Proceedings of the 3<sup>rd</sup> IEEE International Conference on Neural Networks, Washington DC, Vol. 2, 495-498, 1989.

Now consider a scheme in which the encoding operation is broken down into two stages,  $\mathbf{h}(\mathbf{x})$  followed by  $y(\mathbf{h})$ , where  $\mathbf{h}$  is an intermediate code. The  $L_2$  distortion

is then given by

$$D \equiv \int d\mathbf{x} P(\mathbf{x}) \|\mathbf{x}'(y(\mathbf{h}(\mathbf{x}))) - \mathbf{x}\|^2 \quad (2.3)$$

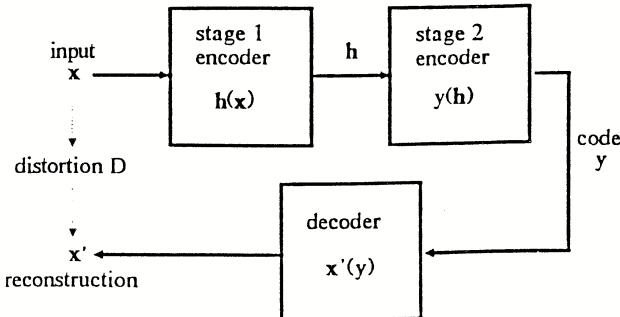


Figure 2: Representation of Equation 2.3 describing the  $L_2$  distortion  $D$  after encoding using  $\mathbf{h}(\mathbf{x})$  followed by  $y(\mathbf{h})$ , and then decoding using  $\mathbf{x}'(y)$ . This is a 2-stage encoder.

Equation 2.3 is depicted schematically in Figure 2, which should be compared with Figure 1. Equation 2.3 is more clumsy than Equation 2.2 because there are many ways of achieving the same overall encoding operation  $y(\mathbf{h}(\mathbf{x}))$  by delicately balancing the functional forms of  $\mathbf{h}(\mathbf{x})$  and  $y(\mathbf{h})$ . In general, the functional forms of  $\mathbf{h}(\mathbf{x})$  and  $y(\mathbf{h})$  are not simple, although  $y(\mathbf{h}(\mathbf{x}))$  itself is a vector quantiser.

The essential step is to model the average distortion due to the second stage of encoding  $y(\mathbf{h})$  as a noise process acting on the output of the first stage  $\mathbf{h}(\mathbf{x})$ . Thus consider a modified distortion defined as

$$D_1 \equiv \int d\mathbf{x} P(\mathbf{x}) \int d\mathbf{n} \pi(\mathbf{n}) \|\mathbf{x}'(\mathbf{h}(\mathbf{x}) + \mathbf{n}) - \mathbf{x}\|^2 \quad (2.4)$$

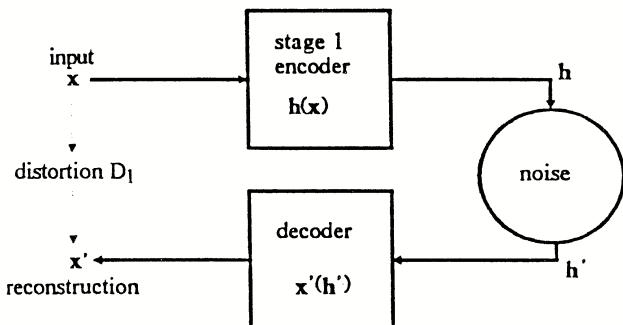


Figure 3: Representation of Equation 2.4 describing the  $L_2$  distortion  $D_1$  after encoding using  $\mathbf{h}(\mathbf{x})$  and modelling the distorting effect of  $y(\mathbf{h})$  (see Figure 2) as a noise process which corrupts  $\mathbf{h}$  into  $\mathbf{h}'$ , and then decoding using  $\mathbf{x}'(\mathbf{h}')$ .

Equation 2.4 is depicted in Figure 3, which is to be compared with Figure 2. This is the  $L_2$  distortion which would arise from encoding  $\mathbf{x}$  using  $\mathbf{h}(\mathbf{x})$ , followed by addition of noise  $\mathbf{n}$  with PDF  $\pi(\mathbf{n})$  (to produce  $\mathbf{h}'$ ), followed

by decoding using  $\mathbf{x}'(\mathbf{h}')$ . By a suitable choice of  $\pi(\mathbf{n})$  we can model the *average* distortion due to  $y(\mathbf{h})$ . Note that we use a somewhat cavalier notation by using  $\mathbf{x}'(\mathbf{h}')$  here and  $\mathbf{x}'(y)$  elsewhere: this does not imply that the functions are the same.

The noise model which we have used in Equation 2.4 is signal independent. More generally we could use a signal dependent noise model to take into account the different distortions introduced by  $y(\mathbf{h})$  for different  $\mathbf{h}$ , in which case  $\pi(\mathbf{n})$  would become  $\pi(\mathbf{n}|\mathbf{h})$ . We do not study such an extension of the model in this paper.

In Figure 4 we show the set theoretic version of Figure 2 (and Figure 3). Note how the distortion is intuitively and consistently represented by the sizes of the subsets depicted.

Neither  $\mathbf{h}(\mathbf{x})$  nor  $y(\mathbf{h})$  is guaranteed to have a simple form (only  $y(\mathbf{h}(\mathbf{x}))$  is a complete vector quantiser), but we have nevertheless represented them as if they mapped clusters of nearby points to the same output: this is purely for diagrammatic convenience. The equivalent noise which  $y(\mathbf{h})$  induces on a particular  $\mathbf{h}$  ( $\mathbf{h}_0$ , say) is then represented by the set of  $\mathbf{h}$  which satisfy  $y(\mathbf{h}) = y(\mathbf{h}_0)$ . In Figure 4 we show how four separate  $\mathbf{h}$  values are mapped to the same  $y$  value, and we show how each of those four  $\mathbf{h}$  values derives from a small cluster of  $\mathbf{x}$  values, and how the four clusters themselves are clustered, thus guaranteeing a small overall  $L_2$  distortion.

Note that we have not yet said anything about the detailed form of the noise model: not even whether it is a continuous function. It is determined entirely by the particular form of the distortion introduced by  $y(\mathbf{h})$ .

### C. A modified learning algorithm

Now let us minimise  $D_1$  with respect to  $\mathbf{h}(\mathbf{x})$  (and, incidentally,  $\mathbf{x}'(\mathbf{h})$ ). The functional derivatives of  $D_1$  with respect to  $\mathbf{h}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{h})$  are given by

$$\frac{\delta D_1}{\delta \mathbf{h}(\mathbf{x})} = P(\mathbf{x}) \int d\mathbf{n} \pi(\mathbf{n}) \left. \frac{\partial \|\mathbf{x}'(\mathbf{h}) - \mathbf{x}\|^2}{\partial \mathbf{h}} \right|_{\mathbf{h}=\mathbf{h}(\mathbf{x})+\mathbf{n}} \quad (2.5)$$

$$\frac{\delta D_1}{\delta \mathbf{x}'(\mathbf{h})} = 2 \int d\mathbf{x} P(\mathbf{x}) \pi(\mathbf{h} - \mathbf{h}(\mathbf{x})) (\mathbf{x}'(\mathbf{h}) - \mathbf{x}) \quad (2.6)$$

These two derivatives imply that the two necessary conditions A1 and A2 for minimising  $D_1$  must be modified to become:

B1. Given  $\mathbf{x}$ , choose  $\mathbf{h}(\mathbf{x})$  to minimise  $\int d\mathbf{n} \pi(\mathbf{n}) \|\mathbf{x}'(\mathbf{h}(\mathbf{x}) + \mathbf{n}) - \mathbf{x}\|^2$ .

B2. Given  $\mathbf{h}$ , choose  $\mathbf{x}'(\mathbf{h})$  to satisfy

$$\mathbf{x}'(\mathbf{h}) = \frac{\int d\mathbf{x} P(\mathbf{x}) \pi(\mathbf{h} - \mathbf{h}(\mathbf{x})) \mathbf{x}}{\int d\mathbf{x} P(\mathbf{x}) \pi(\mathbf{h} - \mathbf{h}(\mathbf{x}))} \quad (2.7)$$

As a consistency check, if  $\pi(\mathbf{n}) = \delta(\mathbf{n})$ , where  $\delta(\mathbf{n})$  is a Dirac delta function (or the analogous Kronecker

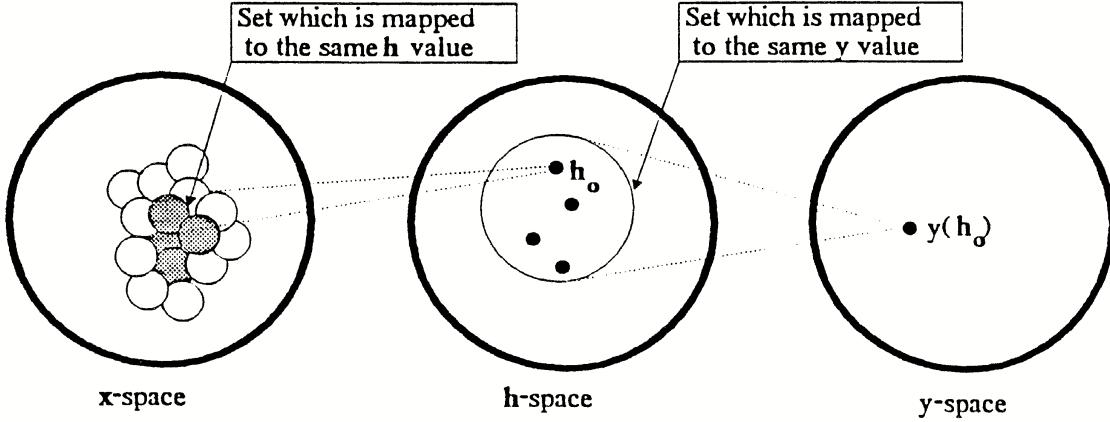


Figure 4: Set theoretic representation of Figure 2.

delta if we consider the discrete case), then the model reduces to the encoding/decoding depicted in Figure 1 (with  $y$  replaced by  $\mathbf{h}$ ), and the above conditions reduce to the standard LBG vector quantisation conditions, as expected.

We shall now break the degeneracy of the set of pairs of functions  $\mathbf{h}(\mathbf{x})$  and  $y(\mathbf{h})$  which minimise the distortion  $D_1$  by assuming that  $y(\mathbf{h})$  is a minimum  $L_2$  distortion vector quantiser for  $\mathbf{h}$ . Thus we shall minimise  $D_2$  which is defined as

$$D_2 \equiv \int d\mathbf{h} P(\mathbf{h}) \|\mathbf{h}'(y(\mathbf{h})) - \mathbf{h}\|^2 \quad (2.8)$$

with respect to  $y(\mathbf{h})$  and  $\mathbf{h}'(y)$ . Given the value of  $D_2$ , and no further information about the distorting effect of  $y(\mathbf{h})$ , a zero mean Gaussian noise model  $\pi(\mathbf{n})$  with variance  $D_2$  is the appropriate maximum entropy choice.

We may simplify B1 by assuming that  $\pi(\mathbf{n})$  is a smooth function of  $\mathbf{n}$ . Thus we Taylor expand in  $\mathbf{n}$  using

$$\mathbf{x}'(\mathbf{h}(\mathbf{x}) + \mathbf{n}) = \exp(\mathbf{n} \cdot \nabla_{\mathbf{h}}) \mathbf{x}'(\mathbf{h})|_{\mathbf{h}=\mathbf{h}(\mathbf{x})} \quad (2.9)$$

where  $\nabla_{\mathbf{h}}$  is the vector derivative operator with respect to  $\mathbf{h}$ . This yields to second order

$$\int d\mathbf{n} \pi(\mathbf{n}) \|\mathbf{x}'(\mathbf{h}(\mathbf{x}) + \mathbf{n}) - \mathbf{x}\|^2 \simeq \left(1 + \frac{D_2}{2} \nabla_{\mathbf{h}}^2\right) (\mathbf{x}'(\mathbf{h}) - \mathbf{x}) \Big|_{\mathbf{h}=\mathbf{h}(\mathbf{x})} \quad (2.10)$$

where we have used

$$\begin{aligned} \int d\mathbf{n} \pi(\mathbf{n}) &= 1 \\ \int d\mathbf{n} \pi(\mathbf{n}) n_i &= 1 \\ \int d\mathbf{n} \pi(\mathbf{n}) n_i n_j &= D_2 \delta_{i,j} \end{aligned} \quad (2.11)$$

The first term on the right hand side of Equation 2.10 is the conventional distortion, and the second (curvature) term arises from the output noise model  $\pi(\mathbf{n})$ . We shall henceforth assume that the curvature term is small, so that B1 may be approximated by A1 (with  $y$  replaced by  $\mathbf{h}(\mathbf{x})$ ). This reduces B1 to a nearest neighbour encoding rule, as before.

B2 cannot be simplified in an analogous fashion to B1. However, from Equation 2.6 we may derive a stochastic gradient descent algorithm for realising B2. Thus we

draw  $\mathbf{x}$  samples at random from the training set using the  $\int d\mathbf{x} P(\mathbf{x})$  factor, and gradient descent of  $D_1$  is then achieved by updates of the form

$$\mathbf{x}'(\mathbf{h}') \longrightarrow \mathbf{x}'(\mathbf{h}') + \epsilon \pi(\mathbf{h}' - \mathbf{h}(\mathbf{x})) (\mathbf{x} - \mathbf{x}'(\mathbf{h}')) \quad (2.12)$$

where  $\epsilon > 0$ ,  $\mathbf{x}$  is drawn from the PDF  $P(\mathbf{x})$ , and  $\mathbf{h}(\mathbf{x})$  is the nearest neighbour encoding approximation to B1. This update prescription is applied to all  $\mathbf{h}'$  for which  $\pi(\mathbf{h}' - \mathbf{h}) > 0$ . This scheme is identical to the one that Kohonen proposed for obtaining topographic mappings [1], and it leads to a decoding operation  $\mathbf{x}'(\mathbf{h})$  which is a smoothly varying function of  $\mathbf{h}$ , although the encoding operation  $\mathbf{h}(\mathbf{x})$  is not guaranteed to be a smooth function of  $\mathbf{x}$  when  $\dim(\mathbf{h}) < \dim(\mathbf{x})$ . Note that we have had to make an approximation (where we retain only the leading term in Equation 2.10) in order to obtain this correspondence between a minimum  $L_2$  distortion and Kohonen's scheme.

Our noise model does not make specific predictions about the dependence of  $\pi(\mathbf{n})$  on the number of updates which have elapsed. However,  $D_1$  has multiple local minima which makes it difficult to locate the global minimum by using the gradient descent algorithm specified in Equation 2.12. The randomness of sampling from the training set makes unpredictable the order in which updates are performed according to Equation 2.12, which alleviates the local minimum problem somewhat. However, a broader  $\pi(\mathbf{n})$  gives rise to a  $D_1$  which has a softer dependence on  $\mathbf{h}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{h})$ , so better convergence to the global minimum of  $D_1$  is obtained by starting with a broad  $\pi(\mathbf{n})$ , and then progressively reducing the width of  $\pi(\mathbf{n})$  as the updates proceed until the required form is obtained [1]. This is equivalent to a renormalised scheme in which  $\pi(\mathbf{n})$  is fixed whilst the underlying space is progressively dilated as the updates proceed [4]. The renormalised scheme gives much faster convergence than Kohonen's original scheme.

A similar approach to that presented in this paper has been presented elsewhere [9]. They minimise a quadratic potential which is analogous to our distortion measure  $D_1$  in Equation 2.4. However, our approach provides a more detailed interpretation of the neighbourhood update factor in the learning algorithm, and it also generalises to multi-stage vector quantisers with no further assumptions. This property is particularly convenient for designing adaptive hierarchical pyramid processors for signal and image analysis.

#### D. Some extensions

All of the results in this paper can be re-expressed using an arbitrary (non- $L_2$ ) distortion measure. The choice

of a preferred form for  $d(\mathbf{x}, \mathbf{x}')$  depends on whatever structure in  $\mathbf{x}$  one deems to be important.

The LBG algorithm is the batch training version of Kohonen's vector quantisation algorithm with zero neighbourhood size. It is also possible to use Equation 2.5 and Equation 2.6 to derive a batch training version of Kohonen's topographic mapping algorithm.

We have presented elsewhere some applications of 2-stage and multi-stage vector quantisation to low-level image processing [4], cluster decomposition of PDFs [7], image compression [6], and time series analysis [5].

### III. SUMMARY AND CONCLUSIONS

We have shown how topographic mappings arise naturally in a 2-stage vector quantiser. The choice of neighbourhood used at the output of the first stage is determined solely by the additional distortion introduced by the second stage of the quantiser. This principle generalises directly to multi-stage quantisers. The advantage of our approach is that (variants of) Kohonen's self-organising neural network may be derived from first principles.

A hierarchical multi-stage quantiser constructed along these lines is an adaptive pyramid processor in which successively cruder quantisations (or codings) of the input are extracted. Current research indicates that such neural networks show much promise [5].

- 
- [1] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
  - [2] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
  - [3] S P Lloyd, *Least squares quantisation in PCM*, IEEE Transactions on Information Theory **28** (1982), no. 2, 129–137.
  - [4] S P Luttrell, *Self organising multilayer topographic mappings*, Proceedings of IEEE conference on neural networks (San Diego), IEEE, 1988, pp. 93–100.
  - [5] ———, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
  - [6] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
  - [7] ———, *Maximum entropy and Bayesian methods*, ch. The use of Bayesian and entropic methods in neural network theory, pp. 363–370, Kluwer, Dordrecht, 1989.
  - [8] J Max, *Quantising for minimum distortion*, IEEE Transactions on Information Theory **6** (1960), no. 1, 7–12.
  - [9] H Ritter and K Schulten, *Kohonen's self-organising maps: Exploring their computational capabilities*, Proceedings of IEEE conference on neural networks (San Diego), IEEE, 1988, pp. 109–116.

# Hierarchical Self-Organising Networks \*

S P Luttrell  
*Royal Signals and Radar Establishment, UK*

## I. INTRODUCTION

Most neural networks are *parametric* models, the parameter values of which are chosen (by some training scheme) to optimise some appropriately chosen cost function.

We shall derive a training scheme for a *non-parametric* neural network, which leads to the vector quantiser. Then we shall introduce a new principle - the robust hidden layer principle - in order to relate the vector quantiser to self-organising neural networks.

Finally we shall demonstrate how hierarchical self-organising neural networks may be constructed by further application of the robust hidden layer principle.

## II. SELF-ORGANISING NETWORKS

We shall first of all clarify some nomenclature concerning supervised and unsupervised networks. We shall compare various networks to show that the vector quantiser emerges as the most general (i.e. non-parametric) unsupervised network.

### A. Supervised and unsupervised networks

The simplest way of comparing and contrasting supervised and unsupervised networks is to consider a single hidden layer network. Define a 2-stage mapping  $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}'$  as follows

$$\begin{aligned}\mathbf{y} &\equiv \mathbf{y}(\mathbf{x}) \\ \mathbf{x}' &\equiv \mathbf{x}'(\mathbf{y})\end{aligned}\quad (2.1)$$

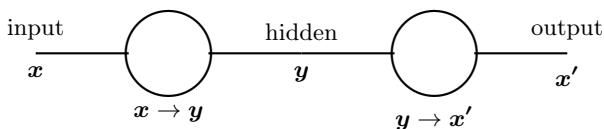


Figure 1: Single hidden layer network.

Table I: Comparison of various networks.

net	$\mathbf{x} \rightarrow \mathbf{y}$	$\mathbf{y} \rightarrow \mathbf{x}'$	target	training
MLP	$\sigma(\sum_j w_{i,j}^1 x_j)$	$\sigma(\sum_j w_{i,j}^2 y_j)$	$\mathbf{t}(\mathbf{x})$	backpropagation
RBF	$\phi( \mathbf{x} - \mathbf{x}^{(i)} )$	$\sum_j w_{i,j} y_j$	$\mathbf{t}(\mathbf{x})$	linear least squares
VQ	$y_i(\mathbf{x})$	$x'_i(\mathbf{y})$	$\mathbf{x}$	LBG, k-means

where  $\mathbf{x}$  is the *input* vector,  $\mathbf{y}$  is the *hidden layer* vector,  $\mathbf{x}'$  is the *output* vector. The values of the vector components may be thought of as the firing rates of neurons in the corresponding network layer. This network is depicted in Figure 1.

If one trains the network to produce a *target* output  $\mathbf{t}(\mathbf{x})$  then the training programme is *supervised*, whereas if one trains the network to reproduce its input ( $\mathbf{t}(\mathbf{x}) = \mathbf{x}$ ) the training programme is termed *unsupervised*. When  $\mathbf{t}(\mathbf{x}) = \mathbf{x}$  the hidden vector  $\mathbf{y}$  should be thought of as the output (it is a reduced representation of  $\mathbf{x}$ ), and the output vector  $\mathbf{x}'$  should be thought of as the reconstruction (of  $\mathbf{x}$ ). We shall emphasise this later on by folding Figure 1 to bring  $\mathbf{x}$  back next to  $\mathbf{x}'$ .

The dimensionalities of the various vectors are constrained only by  $\dim \mathbf{x} = \dim \mathbf{x}'$  when  $\mathbf{t}(\mathbf{x}) = \mathbf{x}$ . The term *self-organisation* is used to describe the process of adaptation of the mappings  $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}'$  during *unsupervised* training. Clearly, the use of the three terms supervised, unsupervised, and self-organisation is somewhat arbitrary (for historical reasons), because in a sense all networks learn under supervised training conditions and consequently become self-organised.

The functions  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  may be parameterised in whatever fashion one chooses. In Table I we compare three different networks: the multilayer perceptron (MLP), the radial basis function (RBF) network, and what will prove later to be the vector quantiser (VQ). The MLP and the RBF networks are parametric (they use explicit weights) whereas the VQ is non-parametric (no explicit weights). The first mapping  $\mathbf{x} \rightarrow \mathbf{y}$  can be thought of as measuring a *feature set*: the MLP projects onto weight vectors, the RBF network measures distance from weight vectors, and the VQ has yet to be decided. The second mapping  $\mathbf{y} \rightarrow \mathbf{x}'$  combines the measured features to approximate the desired output: both the MLP and the RBF networks project the measurements onto weight vectors, and the VQ has yet to be decided.

In all cases a suitable training scheme will cause the network to optimise some appropriately chosen cost function. The MLP uses backpropagation to adjust both layers of weights according to a “chain rule” which prop-

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This appeared in Proceedings of the 1<sup>st</sup> IEE International Conference on Artificial Neural Networks, London, 2-6, 1989. © Controller, Her Majesty's Stationery Office, 1989.

agates the required change in the output (to best match the target) backwards through the network. The RBF network usually fixes the  $\mathbf{x}^{(i)}$  (see Table I) initially (e.g. a subset of the training set is used as the  $\mathbf{x}^{(i)}$ ), and the weight vectors in the second mapping are optimised by a linear least squares technique (note that there is no sigmoidal nonlinearity at the output of the RBF network). The VQ is non-parametric, so functional differentiation will be used to explore function space and to derive a training scheme.

The unsupervised ( $\mathbf{t}(\mathbf{x}) = \mathbf{x}$ ) versions of the MLP and RBF are special cases of the VQ in which  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  are tightly constrained by specifying them in terms of a finite number of parameters. However, it turns out that remarkably simple functional forms emerge automatically during training of a VQ, despite the fact that the VQ has access to the whole space of functions  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  should it need to use them. The emergence of such simplicity without introducing it explicitly (as in the unsupervised MLP or RBF networks) is pleasing. The same cannot be said of the hypothetical supervised VQ, where no useful results emerge without introducing further constraints.

### B. Training a vector quantiser

To train a VQ we must first choose a cost function  $C_1$ . This choice will impose our preference for a particular type of distortion in the overall mapping  $\mathbf{x} \rightarrow \mathbf{x}'$  and will thus colour the information that is available in the hidden layer vector  $\mathbf{y}$ . The problem of choice of  $C_1$  is a subject in its own right which we shall not dwell on here.

For simplicity we shall use an  $L_2$  (i.e. squared) distortion, but the results that we shall derive can be generalised to other types of distortion. Thus

$$C_1 \equiv \int d\mathbf{x} P_{\mathbf{x}}(\mathbf{x}) |\mathbf{x}'(\mathbf{y}(\mathbf{x})) - \mathbf{x}|^2 \quad (2.2)$$

where  $P_{\mathbf{x}}(\mathbf{x})$  is a probability measure induced on the input space by the training set. The equivalence between integration and summation over the training set is given by

$$\int d\mathbf{x} P_{\mathbf{x}}(\mathbf{x}) (\dots) \longrightarrow \sum_{\mathbf{x}} (\dots) \quad (2.3)$$

where  $(\dots)$  is the argument of the integration or summation, and the denominator  $\sum_{\mathbf{x}}$  is a normalising factor - the number of members in the training set. The information in the training set (ignoring the order of presentation) may be expressed as

$$P_{\mathbf{x}}(\mathbf{x}) = \frac{\sum_{\mathbf{x}} \delta(\mathbf{x} - \mathbf{x}^{(i)})}{\sum_{\mathbf{x}}} \quad (2.4)$$

where  $\delta(\dots)$  is the familiar Dirac delta function. Note that Equation 2.4 is consistent with Equation 2.3. Mathematical derivations are easier when continuous notation (i.e. integrations) is used, whereas numerical algorithms use the equivalent discrete notation. Conversion between the two notations is straightforward.

Because the functions  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  are non-parametric we must functionally differentiate  $C_1$  in order to derive an appropriate gradient descent training procedure. The functional derivatives of  $C_1$  are derived as follows:

$$\begin{aligned} \frac{\delta C_1}{\delta \mathbf{y}(\mathbf{x})} &= \frac{\delta}{\delta \mathbf{y}(\mathbf{x})} \int d\xi P_{\mathbf{x}}(\xi) \int dy \delta(\mathbf{y} - \mathbf{y}(\xi)) |\mathbf{x}'(\mathbf{y}) - \xi|^2 \\ &= -P_{\mathbf{x}}(\mathbf{x}) \int dy \frac{\partial \delta(\mathbf{y} - \mathbf{y}(\mathbf{x}))}{\partial \mathbf{y}} |\mathbf{x}'(\mathbf{y}) - \mathbf{x}|^2 \\ &= +P_{\mathbf{x}}(\mathbf{x}) \int dy \delta(\mathbf{y} - \mathbf{y}(\mathbf{x})) \frac{\partial |\mathbf{x}'(\mathbf{y}) - \mathbf{x}|^2}{\partial \mathbf{y}} \\ &= P_{\mathbf{x}}(\mathbf{x}) \frac{\partial |\mathbf{x}'(\mathbf{y}) - \mathbf{x}|^2}{\partial \mathbf{y}} \Big|_{\mathbf{y}=\mathbf{y}(\mathbf{x})} \end{aligned} \quad (2.5)$$

$$\begin{aligned} \frac{\delta C_1}{\delta \mathbf{x}'(\mathbf{y})} &= \frac{\delta}{\delta \mathbf{x}'(\mathbf{y})} \int dx P_{\mathbf{x}}(\mathbf{x}) |\mathbf{x}'(\mathbf{y}(\mathbf{x})) - \mathbf{x}|^2 \\ &= 2 \int dx P_{\mathbf{x}}(\mathbf{x}) \delta(\mathbf{y} - \mathbf{y}(\mathbf{x})) (\mathbf{x}'(\mathbf{y}) - \mathbf{x}) \end{aligned} \quad (2.6)$$

We have listed the steps of the derivation because they illustrate a number of important and useful points about functional differentiation.

In the first line of Equation 2.5 we have changed the dummy  $\mathbf{x}$  integration variable to a  $\xi$  to avoid a notation clash, and we have used the identity  $\int dy \delta(\mathbf{y} - \mathbf{y}(\xi)) \equiv 1$  to simplify the  $\mathbf{y}(\xi)$  dependence of the integrand ready for functional differentiation. The second line is obtained by using

$$\frac{\delta}{\delta \mathbf{y}(\mathbf{x})} \delta(\mathbf{y} - \mathbf{y}(\xi)) = -\delta(\mathbf{x} - \xi) \frac{\partial}{\partial \mathbf{y}} \delta(\mathbf{y} - \mathbf{y}(\mathbf{x})) \quad (2.7)$$

and integrating out the  $\xi$  variable. Note that the functional derivative has now become a partial derivative. The third line is obtained by integration by parts (surface terms are zero), and finally the fourth line is obtained by integrating out the  $\mathbf{y}$  variable. Equation 2.6 is much simpler to derive, because it uses only the result

$$\frac{\delta \mathbf{x}'(\mathbf{y}(\mathbf{x}))}{\delta \mathbf{x}'(\mathbf{y})} = \delta(\mathbf{y} - \mathbf{y}(\mathbf{x})) \quad (2.8)$$

Note that we have not yet assumed anything about the functional form of  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  (except the rather weak assumption that we can perform functional differentiation). A gradient descent algorithm which optimises (minimises)  $C_1$  can now be written down by inspection of Equation 2.5 and Equation 2.6.

Firstly, we shall optimise  $\mathbf{y}(\mathbf{x})$ . Assuming that  $P_{\mathbf{x}}(\mathbf{x}) \neq 0$  then  $\frac{\delta C_1}{\delta \mathbf{y}(\mathbf{x})} = 0$  implies

$$\mathbf{y}(\mathbf{x}) = \arg \min_{\mathbf{y}} |\mathbf{x}'(\mathbf{y}) - \mathbf{x}|^2 \quad (2.9)$$

Thus the optimum function  $\mathbf{y}(\mathbf{x})$  is completely determined by the current  $\mathbf{x}'(\mathbf{y})$  which is itself optimised by using Equation 2.6. Inspection of Equation 2.9 reveals that the optimum  $\mathbf{y}(\mathbf{x})$  is a *nearest neighbour classifier* which seeks that  $\mathbf{x}'(\mathbf{y})$  (for all  $\mathbf{y}$ ) that is closest to  $\mathbf{x}$ .

The  $P_{\mathbf{x}}(\mathbf{x}) \neq 0$  assumption is violated in Equation 2.4, so we cannot determine  $\mathbf{y}(\mathbf{x})$  for those  $\mathbf{x}$  which are not in the training set. This result should be expected because such  $\mathbf{x}$  do not contribute to the cost function anyway. Strictly speaking, the network fails to generalise to novel inputs. This is a direct consequence of the non-parametric nature of the model that we have used.

However, we shall amend the model so that novel inputs are also mapped according to Equation 2.9. In order for this to be a valid procedure, the optimum solution  $\mathbf{y}(\mathbf{x})$  that would be derived from a fully populated training set (i.e. examples of all possible inputs) must be very similar to the solution that we have discovered from the limited training set. This is a delicate problem involving a trade-off between knowledge encoded in the chosen distortion measure (and network structure) and knowledge encoded in the training set. We shall not investigate this problem further here.

Secondly, we shall optimise  $\mathbf{x}'(\mathbf{y})$ . There are two approaches to achieving  $\frac{\delta C_1}{\delta \mathbf{x}'(\mathbf{y})} = 0$  - *batch* and *continuous* mode training. Thus

$$\mathbf{x}'(\mathbf{y}) = \frac{\int d\mathbf{x} P_{\mathbf{x}}(\mathbf{x}) \delta(\mathbf{y} - \mathbf{y}(\mathbf{x})) \mathbf{x}}{\int d\mathbf{x} P_{\mathbf{x}}(\mathbf{x}) \delta(\mathbf{y} - \mathbf{y}(\mathbf{x}))} \quad \text{batch} \quad (2.10)$$

$$\delta \mathbf{x}'(\mathbf{y}) = \epsilon \delta(\mathbf{y} - \mathbf{y}(\mathbf{x})) (\mathbf{x} - \mathbf{x}'(\mathbf{y})) \quad \text{continuous}$$

Batch training makes a sequence of passes through the training set and updates the network after each pass, whereas continuous training selects samples at random from the training set and updates the network after each sample.

Batch training updates  $\mathbf{x}'(\mathbf{y})$  to be the mean of all input vectors  $\mathbf{x}$  that map to  $\mathbf{y}$ , and thus achieves  $\frac{\delta C_1}{\delta \mathbf{x}'(\mathbf{y})} = 0$  exactly. However, in performing this update the nearest neighbour solution to  $\frac{\delta C_1}{\delta \mathbf{y}(\mathbf{x})} = 0$  (see Equation 2.9) is disrupted. The full batch training algorithm must therefore iterate by alternately solving  $\frac{\delta C_1}{\delta \mathbf{y}(\mathbf{x})} = 0$  and solving  $\frac{\delta C_1}{\delta \mathbf{x}'(\mathbf{y})} = 0$ . Thus a local minimum of  $C_1$  is found by following a zigzag path in the product function space  $\mathbf{y}(\mathbf{x}) \otimes \mathbf{x}'(\mathbf{y})$ .

Continuous training performs gradient descent by incrementing  $\mathbf{x}'(\mathbf{y})$  directly towards  $\mathbf{x}$  ( $0 < \epsilon < 1$ ). Note that  $\delta(\mathbf{y} - \mathbf{y}(\mathbf{x}))$  ensures that  $\delta \mathbf{x}'(\mathbf{y})$  is incremented only at  $\mathbf{y} = \mathbf{y}(\mathbf{x})$ . The training set sampling scheme used in continuous training *must* generate the measure  $d\mathbf{x} P_{\mathbf{x}}(\mathbf{x})$  used in the integration in Equation 2.6, as random sampling indeed does. Continuous training does not achieve  $\frac{\delta C_1}{\delta \mathbf{x}'(\mathbf{y})} = 0$  exactly - it can only approach it by gradually reducing the update parameter  $\epsilon$ .

Batch training is more commonly known as the LBG algorithm, Linde, Buzo and Gray [2], which is equivalent to the *k*-means algorithm of cluster analysis. Continuous training is the approach used by Kohonen [1]. Batch

training can get trapped in local minima of the cost function, whereas the random sampling order used in continuous training alleviates this problem somewhat.

In summary, a VQ is an unsupervised non-parametric 2-layer network  $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}'$ . When optimised according to an  $L_2$  distortion measure the mapping  $\mathbf{x} \rightarrow \mathbf{y}$  becomes a nearest neighbour classifier as specified by Equation 2.9. The mapping  $\mathbf{y} \rightarrow \mathbf{x}'$  then reproduces the appropriate class centre  $\mathbf{x}'(\mathbf{y})$ . The batch version of the training algorithm is familiar as the LBG or k-means algorithm.

### III. ROBUSTNESS OF THE HIDDEN LAYER

We shall now attempt to stabilise the information in the hidden layer of the VQ, so that it is not easily degraded by noise processes of various types. This is a highly desirable property for any network to have. Usually stability is guaranteed by redundancy in the network's internal representation, but we shall seek a similar effect by intelligent training of the network's internal representation.

#### A. Training a noisy vector quantiser

The following argument paraphrases the one presented in Luttrell [7]. Consider a slight generalisation of Equation 2.2

$$C_1' \equiv \int d\mathbf{x} P_{\mathbf{x}}(\mathbf{x}) \int d\mathbf{n} P_{\mathbf{n}}(\mathbf{n}) |\mathbf{x}'(\mathbf{y}(\mathbf{x}) + \mathbf{n}) - \mathbf{x}|^2 \quad (3.1)$$

where  $\mathbf{n}$  is a noise vector added to the hidden layer vector, and  $P_{\mathbf{n}}(\mathbf{n})$  is its probability density. If  $P_{\mathbf{n}}(\mathbf{n}) = \delta(\mathbf{n})$  then  $C_1'$  reduces to  $C_1$ , as expected. The effect of non-zero noise is to increase the distortion. This degradation can be minimised if the VQ is optimised specifically to take account of the presence of the noise - i.e.  $C_1'$  and not  $C_1$  is optimised. The way in which the VQ chooses to represent information in its hidden layer must therefore become stabilised (or robust) with respect to the damaging effects of the noise process. We shall call this the *robust hidden layer principle*.

Functional differentiation of  $C_1'$  yields

$$\frac{\delta C_1'}{\delta \mathbf{y}(\mathbf{x})} = P_{\mathbf{x}}(\mathbf{x}) \int d\mathbf{n} P_{\mathbf{n}}(\mathbf{n}) \left. \frac{\partial |\mathbf{x}'(\mathbf{y}) - \mathbf{x}|^2}{\partial \mathbf{y}} \right|_{\mathbf{y}=\mathbf{y}(\mathbf{x})+\mathbf{n}} \quad (3.2)$$

$$\frac{\delta C_1'}{\delta \mathbf{x}'(\mathbf{y})} = 2 \int d\mathbf{x} P_{\mathbf{x}}(\mathbf{x}) P_{\mathbf{n}}(\mathbf{y} - \mathbf{y}(\mathbf{x})) (\mathbf{x}'(\mathbf{y}) - \mathbf{x}) \quad (3.3)$$

where the manipulations are analogous to those used in Equation 2.5 and Equation 2.6, and the results amount to the replacement  $\delta(\mathbf{n}) \rightarrow P_{\mathbf{n}}(\mathbf{n})$  in those equations.

Firstly we shall optimise  $\mathbf{y}(\mathbf{x})$ . Assume that the low order moments of  $P_{\mathbf{n}}(\mathbf{n})$  are given by

$$\begin{aligned}\int d\mathbf{n} P_{\mathbf{n}}(\mathbf{n}) &= 1 \\ \int d\mathbf{n} P_{\mathbf{n}}(\mathbf{n}) n_i &= 0 \\ \int d\mathbf{n} P_{\mathbf{n}}(\mathbf{n}) n_i n_j &= D \delta_{i,j}\end{aligned}\quad (3.4)$$

and then Taylor expand  $\frac{\delta C_1'}{\delta \mathbf{y}(\mathbf{x})}$  to obtain

$$\frac{\delta C_1'}{\delta \mathbf{y}(\mathbf{x})} \approx P_{\mathbf{x}}(\mathbf{x}) \frac{\partial}{\partial \mathbf{y}} \left( 1 + \frac{D}{2} \frac{\partial^2}{\partial \mathbf{y}^2} \right) |\mathbf{x}'(\mathbf{y}) - \mathbf{x}|^2 \Big|_{\mathbf{y}=\mathbf{y}(\mathbf{x})+\mathbf{n}} \quad (3.5)$$

Assuming that the noise process is zero mean (if not, it can be made so by a systematic opposing bias) then it affects the solution of  $\frac{\delta C_1'}{\delta \mathbf{y}(\mathbf{x})} = 0$  only in second order. Thus for sufficiently weak noise, and assuming that  $P_{\mathbf{x}}(\mathbf{x}) \neq 0$ , we may use Equation 2.9 for  $\mathbf{y}(\mathbf{x})$ .

Secondly, we shall optimise  $\mathbf{x}'(\mathbf{y})$ . As before, there are batch and continuous modes of training. These are obtained by replacing the  $\delta(\mathbf{y} - \mathbf{y}(\mathbf{x}))$  in Equation 2.10 by  $P_{\mathbf{n}}(\mathbf{y} - \mathbf{y}(\mathbf{x}))$  to yield

$$\mathbf{x}'(\mathbf{y}) = \frac{\int d\mathbf{x} P_{\mathbf{x}}(\mathbf{x}) P_{\mathbf{n}}(\mathbf{y} - \mathbf{y}(\mathbf{x})) \mathbf{x}}{\int d\mathbf{x} P_{\mathbf{x}}(\mathbf{x}) P_{\mathbf{n}}(\mathbf{y} - \mathbf{y}(\mathbf{x}))} \quad \text{batch} \quad (3.6)$$

$$\delta \mathbf{x}'(\mathbf{y}) = \epsilon P_{\mathbf{n}}(\mathbf{y} - \mathbf{y}(\mathbf{x})) (\mathbf{x} - \mathbf{x}'(\mathbf{y})) \quad \text{continuous}$$

The  $P_{\mathbf{n}}(\mathbf{y} - \mathbf{y}(\mathbf{x}))$  factor ensures that  $\mathbf{x}'(\mathbf{y})$  is updated not only at  $\mathbf{y} = \mathbf{y}(\mathbf{x})$  (as it was in Equation 2.10), but also at other  $\mathbf{y}$  in the neighbourhood of  $\mathbf{y} = \mathbf{y}(\mathbf{x})$ . The relative size of the *neighbourhood updates* is completely determined by  $P_{\mathbf{n}}(\mathbf{y} - \mathbf{y}(\mathbf{x}))$  - i.e. the noise process against which we are attempting to stabilise the network.

The continuous training algorithm that we derived in Equation 3.6 is almost the same as Kohonen's topographic mapping training algorithm [1]. The only missing element is the shrinking neighbourhood size used in [1] - this would correspond to starting with a high noise level and gradually reducing its level during training. This is an algorithmic trick which is introduced to accelerate convergence of the algorithm by concentrating on coarse structure before fine structure.

Another algorithmic trick is presented in Luttrell [4] which provides a further increase in the convergence rate. In [4] we noted that shrinking the neighbourhood on a fixed total number of neurons is roughly equivalent to using a fixed neighbourhood on an increasing number of neurons, and then used this fact to formulate a renormalised version of the training algorithm.

#### IV. MULTILAYER VECTOR QUANTISER

We shall demonstrate an advantage of factorising a VQ into a hierarchy of operations, and we shall use the robust hidden layer approach of the previous section to derive an appropriate training scheme.

##### A. Many hidden layers

A VQ forms its hidden layer vector  $\mathbf{y}$  directly from its input vector  $\mathbf{x}$ . By analogy with multilayer networks an extension of this is the *multilayer vector quantiser*, in which a sequence of mappings, for instance  $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{z} \rightarrow \mathbf{y}' \rightarrow \mathbf{x}'$ , takes us from the input  $\mathbf{x}$  to the reconstruction  $\mathbf{x}'$  via several hidden layer vectors  $\mathbf{y}$ ,  $\mathbf{z}$  and  $\mathbf{y}'$ . The functions used in this example would then be

$$\begin{aligned}\mathbf{y} &\equiv \mathbf{y}(\mathbf{x}) \\ \mathbf{z} &\equiv \mathbf{z}(\mathbf{y}) \\ \mathbf{y}' &\equiv \mathbf{y}'(\mathbf{z}) \\ \mathbf{x}' &\equiv \mathbf{x}'(\mathbf{y}')\end{aligned}\quad (4.1)$$

This network is depicted in Figure 2. Equation 4.1 and Figure 2 should be compared with their counterparts in Equation 2.1 and Figure 1 for a single hidden layer network. Our notation has been deliberately chosen to pair up network layers which perform a similar function in the unsupervised training mode  $\mathbf{t}(\mathbf{x}) = \mathbf{x}$  (as we shall see later).

The VQ is non-parametric, so a single hidden layer VQ explores all of function space in its attempt to find the optimum  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  to use in Equation 2.1. The cost function  $C_1$  in Equation 2.2 thus attains its lowest possible value, and inserting extra layers of transformation as in Equation 2.2 cannot improve on this. Extra layers can cause further damage to the information carried by the hidden layers, and so the reconstruction distortion can be larger than in the single hidden layer case.

However, if a multilayer VQ has a *hierarchical* structure, then it can be preferable to a single layer VQ, despite the fact that it can (and usually does) yield a higher distortion. In order to appreciate this let us define a measure of the complexity  $\mathcal{C}(\mathbf{y}(\mathbf{x}))$  of  $\mathbf{y}(\mathbf{x})$ . If  $Q$  is the number of quantisation levels available to each vector component, then

$$\mathcal{C}(\mathbf{y}(\mathbf{x})) \equiv \dim \mathbf{y} Q^{\dim \mathbf{x}} \quad (4.2)$$

is the number of entries which would be needed in order to implement  $\mathbf{y}(\mathbf{x})$  directly in table look-up form. Now consider the following factorisation of  $\mathbf{y}(\mathbf{x})$

$$\begin{aligned}\mathbf{x} &\equiv (\mathbf{x}_1, \mathbf{x}_2) \\ \mathbf{y} &\equiv (\mathbf{y}_1, \mathbf{y}_2) \\ y_i &\equiv y_i(x_i), \quad i = 1, 2\end{aligned}\quad (4.3)$$

which is depicted in Figure 3.  $\mathcal{C}(y_1(x_1), y_2(x_2))$  is given by

$$\begin{aligned}\mathcal{C}(y_1(x_1), y_2(x_2)) &\equiv \dim \mathbf{y} (Q^{\dim \mathbf{x}_1} + Q^{\dim \mathbf{x}_2}) \\ &\ll \mathcal{C}(\mathbf{y}(\mathbf{x}))\end{aligned}\quad (4.4)$$

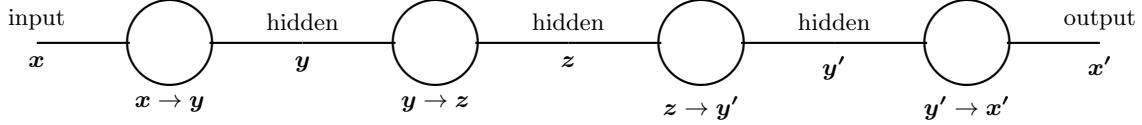


Figure 2: Three hidden layer network.

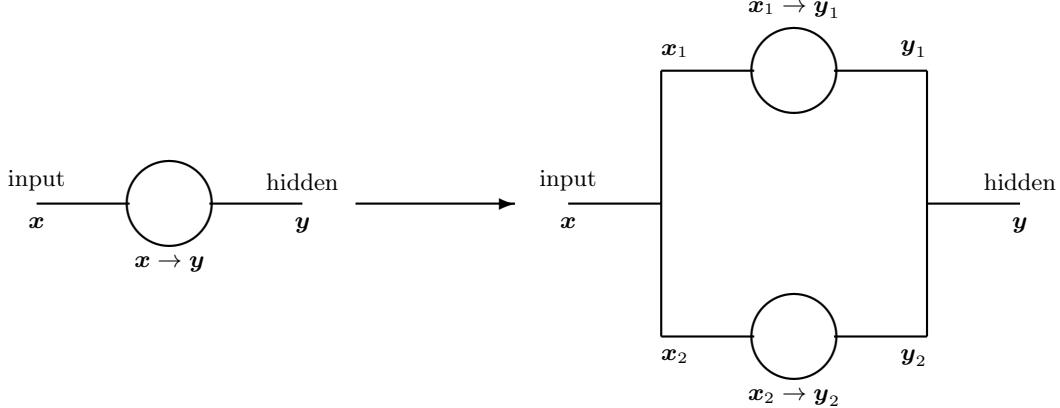


Figure 3: Factorisation of mapping.

The price to be paid for this reduction in complexity is an increased reconstruction distortion. An advantage is that we can represent the mappings in table look-up form provided that we factorise the mappings to a sufficient degree.

Another advantage of a hierarchical VQ is that various representations of the input vector are available at different levels of the hierarchy. Thus it is analogous to a multiresolution (pyramid) image processor for which the word “resolution” means “size of quantisation cell in function space” rather than (say) “spatial resolution”.

### B. Training a two stage vector quantiser

Now consider a two stage VQ - i.e. three hidden layers between input and reconstruction. The appropriate cost function is

$$C_2 \equiv \int d\mathbf{x} P_{\mathbf{x}}(\mathbf{x}) |\mathbf{x}'(\mathbf{y}'(\mathbf{z}(\mathbf{y}(\mathbf{x})))) - \mathbf{x}|^2 \quad (4.5)$$

This is depicted in Figure 4 where we have now chosen a notation and folded the diagram (as promised in the discussion following Equation 2.1) to make clear which stages of the VQ perform similar functions. The hidden vector  $\mathbf{z}$  should be thought of as the output (it is a reduced representation of  $\mathbf{x}$ ), and the output vector  $\mathbf{x}'$  should be thought of as the reconstruction (of  $\mathbf{x}$ ).

Optimisation of  $C_2$  involves four separate functions -  $\mathbf{y}(\mathbf{x})$ ,  $\mathbf{z}(\mathbf{y})$ ,  $\mathbf{y}'(\mathbf{z})$ , and  $\mathbf{x}'(\mathbf{y}')$ . These combine into composite functions  $\mathbf{z}(\mathbf{y}(\mathbf{x}))$  in the forward direction, and  $\mathbf{x}'(\mathbf{y}'(\mathbf{z}))$  in the reverse direction. Thus the above two

stage VQ using *simple* functions is equivalent to the single stage VQ  $\mathbf{x} \rightarrow \mathbf{z} \rightarrow \mathbf{x}'$  using *composite* functions. Unfortunately, by delicately balancing functional forms there is an infinite number of ways in which  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{z}(\mathbf{y})$  can combine to yield a given  $\mathbf{z}(\mathbf{y}(\mathbf{x}))$ , and similarly for  $\mathbf{y}'(\mathbf{z})$ ,  $\mathbf{x}'(\mathbf{y}')$ , and  $\mathbf{x}'(\mathbf{y}'(\mathbf{z}))$ . Optimisation of  $C_2$  alone does not give a simple solution to the two stage VQ problem (in the way that  $C_1$  did for the single stage VQ), so we need to introduce some further constraints to bound the optimisation problem more tightly.

There are many feasible solutions to the problem of additional constraints. We will constrain the allowed forms for the functions by imposing the extra condition that  $\mathbf{y} \rightarrow \mathbf{z} \rightarrow \mathbf{y}'$  itself be a VQ. This is not a necessary choice, but it is sufficient to yield some very useful results. We must therefore optimise the boxed section of Figure 4.

An implementation problem now arises, because optimising  $\mathbf{y} \rightarrow \mathbf{z} \rightarrow \mathbf{y}'$  requires knowledge of  $\mathbf{y}(\mathbf{x})$  in order to generate the  $\mathbf{y}$  training set from the  $\mathbf{x}$  training set. In principle, we could optimise  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  as an outer “optimisation loop”, and  $\mathbf{z}(\mathbf{y})$  and  $\mathbf{y}'(\mathbf{z})$  as an inner “optimisation loop”, but this would be very time consuming.

We shall therefore use an approximate optimisation scheme as shown in the right hand half of Figure 4, where we have performed the transformation shown in Figure 5. The left hand half of Figure 5 is a (deterministic) single stage VQ which maps  $\mathbf{y} \rightarrow \mathbf{y}'$  overall, whereas the right hand half of Figure 5 is a (stochastic) approximation to this mapping, which we shall derive below. Thus we optimise the first stage of the VQ in Figure 4 in the manner

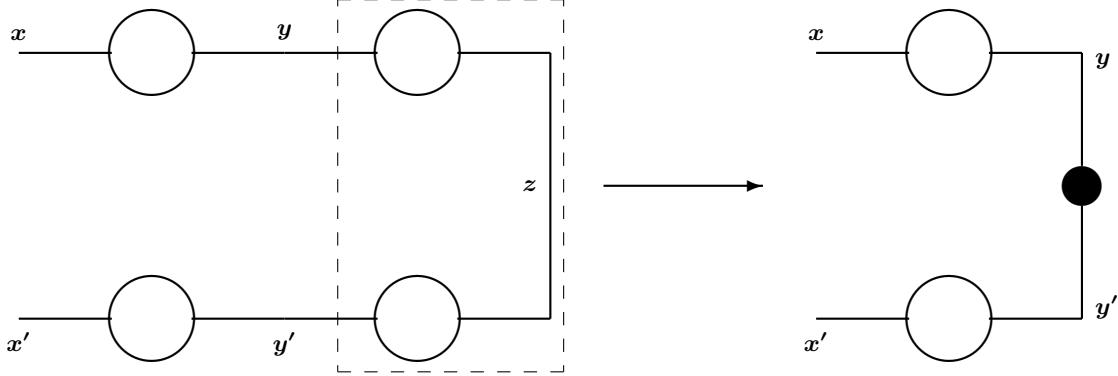


Figure 4: Training two stage vector quantiser.

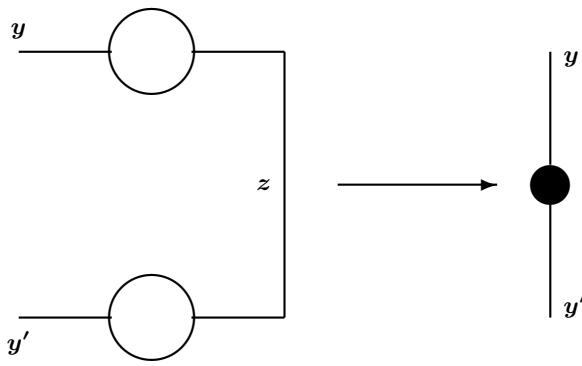


Figure 5: Stochastic model of vector quantiser.

of a noisy VQ as in Equation 3.1. The outer and inner “optimisation loops” have been decoupled by representing the effects of  $\mathbf{y} \rightarrow \mathbf{z} \rightarrow \mathbf{y}'$  (inner loop) as a stochastic process. When we are satisfied that  $\mathbf{y}(x)$  has been suitably optimised (outer loop), we can use it to generate the  $\mathbf{y}$  training set required for optimising  $\mathbf{y} \rightarrow \mathbf{z} \rightarrow \mathbf{y}'$  (inner loop).

We shall model the stochastic mapping as an additive noise process. Thus  $\mathbf{y} \rightarrow \mathbf{z} \rightarrow \mathbf{y}' = \mathbf{y} + \mathbf{n}$ , where  $\mathbf{n}$  is described by  $P_{\mathbf{n}}(\mathbf{n})$ . We have used the same notation as in Equation 3.1 because they are describing exactly the same process. We may also determine the  $P_{\mathbf{n}}(\mathbf{n})$  which guarantees that in Figure 5 the stochastic mapping approximates the deterministic mapping. Thus

$$\begin{aligned} \mathbf{n}(\mathbf{y}) &\equiv \mathbf{y}'(\mathbf{z}(\mathbf{y})) - \mathbf{y} \\ \int d\mathbf{y} P_{\mathbf{y}}(\mathbf{y}) \mathbf{n}(\mathbf{y}) &= \mathbf{0} \\ \int d\mathbf{y} P_{\mathbf{y}}(\mathbf{y}) \text{tr}(\mathbf{n}(\mathbf{y}) \mathbf{n}(\mathbf{y})^T) &= D \dim \mathbf{y} \end{aligned} \quad (4.6)$$

where  $P_{\mathbf{y}}(\mathbf{y})$  is the probability measure on  $\mathbf{y}$  given by

$$P_{\mathbf{y}}(\mathbf{y}) = \int dx P_x(x) \delta(\mathbf{y} - \mathbf{y}(x)) \quad (4.7)$$

The zero mean property is guaranteed by the minimum  $L_2$  distortion property of the  $\mathbf{y} \rightarrow \mathbf{z} \rightarrow \mathbf{y}'$  VQ, and the trace of the covariance matrix  $\text{tr}(\mathbf{n}(\mathbf{y}) \mathbf{n}(\mathbf{y})^T)$  is just the  $L_2$  norm of the distortion  $\mathbf{n}(\mathbf{y})$ .

Given only the zero mean and the trace of covariance of the noise process the appropriate maximum entropy choice for  $P_{\mathbf{n}}(\mathbf{n})$  is

$$P_{\mathbf{n}}(\mathbf{n}) = \frac{1}{(2\pi D)^{\frac{\dim \mathbf{y}}{2}}} \exp\left(-\frac{|\mathbf{n}|^2}{2D}\right) \quad (4.8)$$

The maximum entropy prescription is the only consistent way of assigning a probability distribution given limited information, Shore and Johnson [8, 9]. In principle we could measure other properties of  $\mathbf{n}(\mathbf{y})$  in order to characterise  $P_{\mathbf{n}}(\mathbf{n})$  more precisely. However, we have found Equation 4.8, or approximations thereof, to be adequate.

A criticism of this stochastic approach is that it requires knowledge of the variance of the noise process corresponding to the boxed section of Figure 4 *in advance of* optimising the boxed section. However, there is only one parameter to estimate (i.e.  $D$  in Equation 4.6) to describe the noise model of the boxed section. The precise value of  $D$  is not critical, so it can be estimated by splitting up  $\mathbf{y}$ -space into equal volume cells, the number of which is equal to the number of states in  $\mathbf{z}$ -space. The squared radius of each cell is then a good estimate (of an upper bound for) of the variance  $D$  of the distortion. In practice the input space is not uniformly populated by training set samples, so the actual distortion will be somewhat less than we estimate. In our numerical experiments we have found that errors in estimating this distortion do not upset the optimisation process.

### C. Further considerations

This procedure can be extended in a straightforward manner to a multilayer VQ. Progressing from the first to the last stage of the folded VQ (see Figure 4 for the two stage VQ), each stage is optimised as a noisy vector quantiser (except for the last stage which is noiseless). The noise variance used in the cost function of each stage is the estimated variance of the distortion produced by *all* subsequent stages of the VQ.

In practice a finite number of bits is used for each hidden layer vector component. Although we expressed our derivations in terms of functional differentiation, the extension to discrete spaces causes no problems. In a multilayer VQ, the greater the number of bits that can be carried through the intermediate stages the better. In the limiting case of infinite accuracy, the overall distortion will be the same as for a single stage VQ, because the information will not be damaged by the extra layers.

The simplest hierarchical VQ is a binary hierarchical tree in which 2-component vector inputs are mapped to 1-component scalar outputs at each junction of the tree. In the language of Equation 4.3 and Figure 3,  $\dim \mathbf{x} = 4$ ,  $\dim \mathbf{y} = 2$ ,  $\dim \mathbf{y}_i = 1, i = 1, 2$ . A table look-up implementation may then easily be implemented using 8 bits per vector component.  $Q = 2^8 = 256$  in Equation 4.2, so each look-up table must have  $\mathcal{C}(\dots) = 256^2 = 65536$  entries, and each entry must be 8 bits (i.e. a byte)

in size to permit 256 different possible output states to be represented.

We have presented elsewhere some applications of hierarchical vector quantisers (or self-organising networks). Basic algorithmic tricks and a demonstration of the use of topological prior knowledge may be found in [4], compression of synthetic aperture radar images may be found in Luttrell [3, 6], compression of assorted synthetic time series can be found in Luttrell [5].

### V. CONCLUSIONS

We have shown how to derive a single hidden layer self-organising neural network (as a vector quantiser) from two main assumptions - minimum  $L_2$  distortion, and the robust hidden layer principle. We have extended this to any (odd) number of hidden layers by further assuming that each layer must be a vector quantiser for the previous layer's output.

A special case of the multilayer case, the hierarchical neural network, may be used to realise vector quantisers entirely in table look-up form. It may also be used to preprocess high dimensional input data into a hierarchy of reduced representations, which may then be subsequently processed in the manner of multiresolution image processing.

- 
- [1] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
  - [2] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
  - [3] S P Luttrell, *Image compression using a neural network*, Proceedings of IGARSS conference on remote sensing (Edinburgh), ESA, 1988, pp. 1231–1238.
  - [4] ———, *Self organising multilayer topographic mappings*, Proceedings of IEEE conference on neural networks (San Diego), IEEE, 1988, pp. 93–100.
  - [5] ———, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
  - [6] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
  - [7] ———, *Self-organisation: a derivation from first principles of a class of learning algorithms*, Proceedings of IJCNN international joint conference on neural networks (Washington DC), IEEE, 1989, pp. 495–498.
  - [8] J E Shore and R W Johnson, *Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross entropy*, IEEE Transactions on Information Theory **26** (1980), no. 1, 26–37.
  - [9] ———, *Comments on and correction to ‘axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy’ (Jan 80 26–37) (Correspondence)*, IEEE Transactions on Information Theory **29** (1983), no. 6, 942–943.



# A Hierarchical Network for Clutter and Texture Modelling \*

Stephen P Luttrell<sup>†</sup>

Defence Research Agency: Electronics Division, St Andrews Rd,  
Malvern, WORCS, WR14 3PS, United Kingdom

The presence of clutter complicates the location of targets in time series and images. Various types of adaptive clutter model have been proposed to deal with this problem. In this paper we treat clutter as a type of texture, and we propose a novel type of hierarchical Gibbs distribution texture model. To optimise this type of model, we define a relative entropy cost function which we decompose into a sum over a number of terms, each of which can be interpreted as the mutual information between clusters of samples of the data. Furthermore we show how the various terms of this cost function can be used to construct an image-like representation of the relative entropy. Finally, using a Brodatz texture image, we present an example of this type of decomposition, and demonstrate that a statistical anomaly in the Brodatz texture image can be easily located.

## I. INTRODUCTION

This paper addresses the problem of building a Gibbs distribution (i.e. maximum entropy) model approximating the joint probability density function (PDF) of the pixels of a textured image. Such a model can be used in Bayesian inference [3, 4] to determine the probability that each area of the image is a piece of texture or something entirely different (e.g. an anomaly). There are many approaches to constructing Gibbs distribution models, such as the Boltzmann machine [1] or the Gibbs machine [5, 6]. In this paper we use a Gibbs distribution model [7], which is built up from the statistical properties measured at each layer of a hierarchy of transformations of the input data. The goal of this paper is not the training of, but rather the interpretation of, this type of hierarchical Gibbs distribution model.

In Section II we derive the basic maximum entropy expressions for a Gibbs distribution, constrained so that various of its marginal PDFs adopt required values. In Section III we introduce a relative entropy cost function in order to quantify the performance of the Gibbs distribution in Section II, and we show how this relative entropy may be expressed as a sum of mutual informations. In Section IV we present an application of the relative entropy results in Section III to a Brodatz texture image [2], which we display as a relative entropy image.

## II. CLUSTER DECOMPOSITION

In this section we derive a Gibbs distribution model of the PDF  $P(\mathbf{X})$  of a multidimensional data vector  $\mathbf{X}$ . The particular Gibbs distribution is determined by the

set of constraints that we impose: we choose to constrain some of its marginal PDFs.

We present a simplified derivation in which  $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2)$ , which we then generalise to the case  $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ . Finally, we further generalise this to accommodate a hierarchical decomposition of the joint PDF.

### A. Two clusters

Let  $N = 2$ , whence  $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2)$  and  $P(\mathbf{X}) = P(x_1, \mathbf{x}_2)$ . Define two marginals as

$$\begin{aligned} P_1(\mathbf{x}_1) &\equiv \int d\mathbf{x}_2 P_{12}(\mathbf{x}_1, \mathbf{x}_2) \\ P_2(\mathbf{x}_2) &\equiv \int d\mathbf{x}_1 P_{12}(\mathbf{x}_1, \mathbf{x}_2) \end{aligned} \quad (2.1)$$

and derive transformed versions of the  $\mathbf{x}_i$  as

$$\begin{aligned} y_1 &= y_1(\mathbf{x}_1) \\ y_2 &= y_2(\mathbf{x}_2) \end{aligned} \quad (2.2)$$

and use Equation 2.2 to obtain the joint PDF  $p_{12}(y_1, y_2)$  of the  $y_i$  as

$$p_{12}(y_1, y_2) = \int d\mathbf{x}_1 d\mathbf{x}_2 P_{12}(\mathbf{x}_1, \mathbf{x}_2) \delta(y_1 - y_1(\mathbf{x}_1)) \delta(y_2 - y_2(\mathbf{x}_2)) \quad (2.3)$$

where the Dirac delta functions constrain the integration to the surface  $(y_1, y_2) = (y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))$ . From Equation 2.3 we define two further marginals as

$$\begin{aligned} p_1(y_1) &\equiv \int dy_2 p_{12}(y_1, y_2) \\ p_2(y_2) &\equiv \int dy_1 p_{12}(y_1, y_2) \end{aligned} \quad (2.4)$$

We use  $P_1(\mathbf{x}_1)$ ,  $P_2(\mathbf{x}_2)$  and  $p_{12}(y_1, y_2)$  as constraints on a maximum entropy estimate  $Q_{mem,12}(\mathbf{x}_1, \mathbf{x}_2)$  of

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 22, 2019.

This appeared in Proceedings of the SPIE Conference on Adaptive Signal Processing, San Diego, 518-528, 1991.

<sup>†</sup>Electronic address: luttrell%hermes.mod.uk@relay.mod.uk

$P(x_1, \mathbf{x}_2)$ . We present this derivation in detail in Appendix A, and the final result is

$$Q_{mem,12}(\mathbf{x}_1, \mathbf{x}_2) = P_1(\mathbf{x}_1) P_2(\mathbf{x}_2) \frac{p_{12}(y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))}{p_1(y_1(\mathbf{x}_1)) p_2(y_2(\mathbf{x}_2))} \quad (2.5)$$

This result has an intuitively obvious form. The  $P_1(\mathbf{x}_1) P_2(\mathbf{x}_2)$  factor is the solution that we would ex-

pect if the constraints on  $p_{12}(y_1, y_2)$  were omitted. This is because there would then be no information about correlations between  $\mathbf{x}_1$  and  $\mathbf{x}_2$ . The dimensionless factor  $\frac{p_{12}(y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))}{p_1(y_1(\mathbf{x}_1)) p_2(y_2(\mathbf{x}_2))}$  introduces knowledge of correlations between  $\mathbf{x}_1$  and  $\mathbf{x}_2$ . Note that this factor is unity when  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are statistically independent variables.

## B. $N$ clusters

We now extend the results of Section II A to  $N$  clusters. Thus

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \quad (2.6)$$

No further derivations are needed because the details are almost identical to those in Section II A. The results which correspond to Equation 2.1, Equation 2.2, Equation 2.3 and Equation 2.4 are respectively (for  $i = 1, 2, \dots, N$ )

$$P_i(\mathbf{x}_i) = \int d\mathbf{x}_1 d\mathbf{x}_2 \cdots d\hat{\mathbf{x}}_i \cdots d\mathbf{x}_{N-1} d\mathbf{x}_N P(\mathbf{X}) \quad (2.7)$$

$$y_i = y_i(\mathbf{x}_i) \quad (2.8)$$

$$p_{12\dots N}(y_1, y_2, \dots, y_N) = \int d\mathbf{x}_1 d\mathbf{x}_2 \cdots d\mathbf{x}_N \delta(y_1 - y_1(\mathbf{x}_1)) \delta(y_2 - y_2(\mathbf{x}_2)) \cdots \delta(y_N - y_N(\mathbf{x}_N)) P(\mathbf{X}) \quad (2.9)$$

$$p_i(y_i) = \int dy_1 dy_2 \cdots \hat{dy}_i \cdots dy_{N-1} dy_N p_{12\dots N}(y_1, y_2, \dots, y_N) \quad (2.10)$$

where the hats in Equation 2.7 and Equation 2.10 denote an omitted integral. This leads to a generalisation of Equation 2.5 as

$$Q_{mem}(\mathbf{X}) = P_1(\mathbf{x}_1) P_2(\mathbf{x}_2) \cdots P_N(\mathbf{x}_N) \frac{p_{12\dots N}(y_1(\mathbf{x}_1), y_2(\mathbf{x}_2), \dots, y_N(\mathbf{x}_N))}{p_1(y_1(\mathbf{x}_1)) p_2(y_2(\mathbf{x}_2)) \cdots p_N(y_N(\mathbf{x}_N))} \quad (2.11)$$

The interpretation of Equation 2.11 is analogous that of Equation 2.5.

## C. Hierarchical clusters

First of all, we extend our notation in Equation 2.2 to accommodate an arbitrary number of stages of transformation. Thus we write

$$\begin{aligned} \mathbf{X}^{(l)} &= (\mathbf{x}_1^{(l)}, \mathbf{x}_2^{(l)}, \dots, \mathbf{x}_{N_l}^{(l)}) ; \quad l = 0, 1, \dots, L-1 \\ y_i^{(l)} &= y_i^{(l)}(\mathbf{x}_i^{(l-1)}) ; \quad l = 1, 2, \dots, L; \quad i = 1, 2, \dots, N_{l-1} \\ \mathbf{X}^{(l)} &= (y_1^{(l)}, y_2^{(l)}, \dots, y_{N_{l-1}}^{(l)}) ; \quad l = 1, 2, \dots, L-1 \end{aligned} \quad (2.12)$$

where  $l$  denotes the layer of the hierarchy (numbering from 0 upwards), and  $i$  denotes the position of a node in the layer. Note that for notational convenience, after we transform from layer  $l-1$  to layer  $l$ , we gather together the  $y_i^{(l)}$  to form a vector  $\mathbf{X}^{(l)}$  representing the state of the entire layer  $l$  of the hierarchy.

We then obtain the maximum entropy estimate  $Q_{mem}(\mathbf{X})$  as

$$\begin{aligned}
Q_{mem}(\mathbf{X}) &= \left( \frac{P_1^{(0)}(\mathbf{x}_1^{(0)})}{p_1^{(1)}(y_1^{(1)})} \frac{P_2^{(0)}(\mathbf{x}_2^{(0)})}{p_2^{(1)}(y_2^{(1)})} \dots \frac{P_{N_0}^{(0)}(\mathbf{x}_{N_0}^{(0)})}{p_{N_0}^{(1)}(y_{N_0}^{(1)})} \right) \\
&\times \left( \frac{P_1^{(1)}(\mathbf{x}_1^{(1)})}{p_1^{(2)}(y_1^{(2)})} \frac{P_2^{(1)}(\mathbf{x}_2^{(1)})}{p_2^{(2)}(y_2^{(2)})} \dots \frac{P_{N_1}^{(1)}(\mathbf{x}_{N_1}^{(1)})}{p_{N_1}^{(2)}(y_{N_1}^{(2)})} \right) \\
&\vdots \\
&\times \left( \frac{P_1^{(L-1)}(\mathbf{x}_1^{(L-1)})}{p_1^{(L)}(y_1^{(L)})} \frac{P_2^{(L-1)}(\mathbf{x}_2^{(L-1)})}{p_2^{(L)}(y_2^{(L)})} \dots \frac{P_{N_{L-1}}^{(L-1)}(\mathbf{x}_{N_{L-1}}^{(L-1)})}{p_{N_{L-1}}^{(L)}(y_{N_{L-1}}^{(L)})} \right) \\
&\times p_{12\dots N_{L-1}}^{(L)}(y_1^{(L)}, y_2^{(L)}, \dots, y_{N_{L-1}}^{(L)}) \tag{2.13}
\end{aligned}$$

We write  $Q_{mem}(\mathbf{X})$  in a form that arranges the various factors differently from Equation 2.5 and Equation 2.11. We do this to reveal that  $Q_{mem}(\mathbf{X})$  is simply the joint PDF  $p_{12\dots N_{L-1}}^{(L)}(y_1^{(L)}, y_2^{(L)}, \dots, y_{N_{L-1}}^{(L)})$  of the nodes in the final layer of the hierarchy, weighted by a product of factors which compensate for the compressions introduced by the various transformations  $y_i^{(l)} = y_i^{(l)}(\mathbf{x}_i^{(l-1)})$ .

Note that for  $L = 1$  Equation 2.13 reduces to Equation 2.11, where  $N = N_0$ ,  $\mathbf{X} = \mathbf{X}^{(0)}$ ,  $\mathbf{x}_i^{(0)} = \mathbf{x}_i$ ,  $y_i^{(1)} = y_i$ ,  $P_i^{(0)}(\mathbf{x}_i^{(0)}) = P_i(\mathbf{x}_i)$  and  $p_i^{(1)}(y_i^{(1)}) = p_i(y_i)$ .

### III. QUANTIFYING THE PERFORMANCE OF THE GIBBS DISTRIBUTION

In this section we define a measure of the quality of the maximum entropy estimate  $Q_{mem}(\mathbf{X})$  of  $P(\mathbf{X})$  given in Equation 2.13 and we relate the properties of this measure to the properties of mutual information as described in Appendix B.

#### A. Relative entropy

A measure of the quality of the maximum entropy estimate is relative entropy defined as

$$G \equiv - \int d\mathbf{X} P(\mathbf{X}) \log \left( \frac{P(\mathbf{X})}{Q_{mem}(\mathbf{X})} \right) \leq 0 \tag{3.1}$$

where  $Q_{mem}(\mathbf{X})$  is given in Equation 2.13.  $G$  has the following interpretation in terms of generalised Bernoulli trials. Define a model PDF  $Q_{mem}(\mathbf{X})$ , and use it as a source of independent samples of  $\mathbf{X}$ . Now construct a set containing a large number  $M$  of samples drawn from  $Q_{mem}(\mathbf{X})$ , and ask what is the probability that this set could have been drawn from  $Q_{mem}(\mathbf{X})$  (irrespective of the order in which the individual samples were drawn). If we use  $P(\mathbf{X})$  to denote the relative frequency of the various  $\mathbf{X}$  that is observed in this set of  $M$  samples, then a brief derivation (making use of Stirling's approximation  $\log M! \simeq M \log M - M$ ) yields an answer  $e^{MG}$  for the required probability. Thus the smaller  $e^{MG}$  is (or, equivalently, the more negative  $G$  is) the more surprised we are that the observed  $P(\mathbf{X})$  is genuinely a possible relative frequency (ignoring ordering) of independent samples from a model  $Q_{mem}(\mathbf{X})$ .

#### B. Mutual information

In Appendix B we outline some of the basic properties of mutual information, the most important of which is

$$I[X_1; X_2; \dots; X_n] = I[\mathbf{x}_1; \mathbf{x}_2; \dots; \mathbf{x}_N] + \sum_{i=0}^{N-1} I[X_{p_i}; X_{p_i+1}; \dots; X_{p_{i+1}}] \tag{3.2}$$

where we define the notation in Equation B5. Equation 3.2 is very intuitive, because it states that the total mu-

tual information between all components  $X_i$  of  $\mathbf{X}$  may be divided into two parts. The first part is the mutual infor-

mation between separate clusters of components  $\mathbf{x}_i$  of  $\mathbf{X}$ . The second part is the mutual information between the components  $(X_{p_i+1}, X_{p_i+2}, \dots, X_{p_{i+1}})$  of a single cluster, summed over clusters. We call these the inter-cluster and the intra-cluster mutual information, respectively. For layer  $l$  of the hierarchy Equation 3.2 becomes

$$I_{\text{total}}^{(l)} = I_{\text{inter}}^{(l)} + \sum_{i=1}^{N_l} I_{\text{intra},i}^{(l)} \quad (3.3)$$

### C. Decomposition of relative entropy into a sum of mutual informations

We now insert Equation 2.13 into Equation 3.1 to obtain some explicit results for the relative entropy. For the case  $L = 1$  we obtain (using  $\mathbf{X} = \mathbf{X}^{(0)}$ )

$$\begin{aligned} G &= - \int d\mathbf{X} P(\mathbf{X}) \log \left( \frac{P(\mathbf{X})}{P_1^{(0)}(\mathbf{x}_1^{(0)}) P_2^{(0)}(\mathbf{x}_2^{(0)}) \dots P_{N_0}^{(0)}(\mathbf{x}_{N_0}^{(0)})} \right) \\ &\quad + \int d\mathbf{X} P(\mathbf{X}) \log \left( \frac{p_{12\dots N_0}^{(1)}(y_1^{(1)}, y_2^{(1)}, \dots, y_{N_0}^{(1)})}{p_1^{(1)}(y_1^{(1)}) p_2^{(1)}(y_2^{(1)}) \dots p_{N_0}^{(1)}(y_{N_0}^{(1)})} \right) \\ &= I_{\text{total}}^{(1)} - I_{\text{inter}}^{(1)} \\ &\leq 0 \end{aligned} \quad (3.4)$$

where  $y_i^{(1)} = y_i^{(1)}(\mathbf{x}_i^{(0)})$  is to be understood. Note that we have used Equation B7 for  $l = 1$  and Equation B8 for  $l = 0$  to express  $G$  in terms of mutual information. The final inequality in Equation 3.4 follows from Equation B9 when we set  $l = 0$ .

We may generalise Equation 3.4 to the case  $L \geq 1$

$$G = \sum_{l=1}^{L-1} \left( I_{\text{inter}}^{(l+1)} - I_{\text{inter}}^{(l)} \right) \quad (3.5)$$

We see that the mutual information loss  $I_{\text{inter}}^{(l)} - I_{\text{inter}}^{(l+1)}$  in transforming from  $l$  to layer  $l + 1$  determines to what extent  $G$  (and hence the quality of the Gibbs distribution model) is affected by that layer of the hierarchy. Finally, we may express Equation 3.5 in an alternative form by using Equation B13 to yield

$$\begin{aligned} G &= I_{\text{total}}^{(L)} - I_{\text{inter}}^{(0)} + \sum_{l=1}^{L-1} \sum_{i=1}^{N_l} I_{\text{intra},i}^{(l)} \\ &= \sum_{\text{cliques}} I_{\text{clique}} + \text{constant} \end{aligned} \quad (3.6)$$

where a “clique” is a set of sibling nodes in the hierarchical network.

#### IV. IMAGE-LIKE REPRESENTATION OF RELATIVE ENTROPY

In this section we apply the result in Equation 3.6 to a binary tree of transformations of image data  $\mathbf{X}$ . There are many ways of connecting a binary tree to the pixels of

an image. However, we restrict our attention to a binary tree that processes the pixels in the order shown in figure 1.

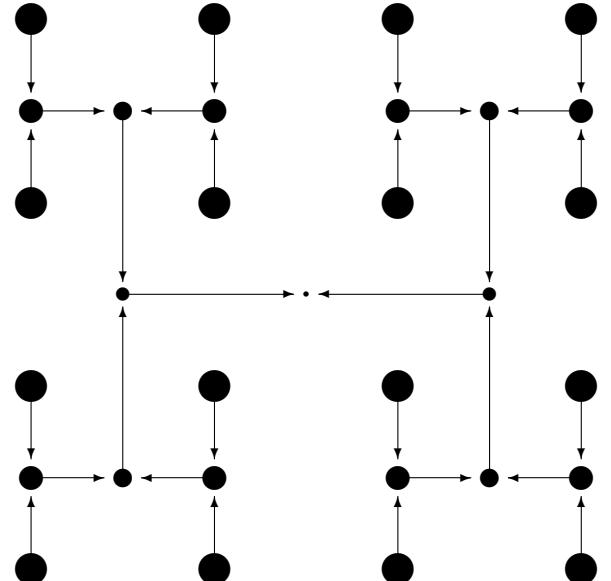


Figure 1: Order of pixel processing for an image.

We also show how to represent the various contributions to  $G$  as images, which may then be inspected visually to locate contributions of interest, and we demonstrate this procedure using a Brodatz texture image.

### A. Clique mutual information

For a binary tree, each clique has a state that we represent as a vector  $(x_1^{(cl)}, x_2^{(cl)})$ , and a mutual information

$I_{clique}$  given by

$$\begin{aligned} I_{clique} &= I[x_1^{(cl)}; x_2^{(cl)}] \\ &= \int dx_1^{(cl)} dx_2^{(cl)} P(x_1^{(cl)}, x_2^{(cl)}) \log \left( \frac{P(x_1^{(cl)}, x_2^{(cl)})}{P(x_1^{(cl)}) P(x_2^{(cl)})} \right) \end{aligned} \quad (4.1)$$

Note that  $I_{clique}$  depends on  $P(x_1^{(cl)}, x_2^{(cl)})$ , which, in turn, depends on the co-occurrence matrix of the elements of the vector  $(x_1^{(cl)}, x_2^{(cl)})$ . Ideally, in order to evaluate the integral  $\int dx_1^{(cl)} dx_2^{(cl)} P(x_1^{(cl)}, x_2^{(cl)}) (\dots)$  we require a representative ensemble of data  $\mathbf{X}$ , which would yield an approximation

$$I_{clique} \simeq \left\langle \log \left( \frac{P(x_1^{(cl)}, x_2^{(cl)})}{P(x_1^{(cl)}) P(x_2^{(cl)})} \right) \right\rangle \quad (4.2)$$

where  $\langle \dots \rangle$  represents an ensemble average. However, in practice we use a single snapshot of the data, which yields only a very crude estimate of  $I_{clique}$ .

### B. Relative entropy image

It proves to be very convenient (and visually intuitive) to combine the single snapshot estimates of the various  $I_{clique}$  into an image-like representation. Each clique  $(x_1^{(cl)}, x_2^{(cl)})$  has a state that derives from the patch of input data  $\mathbf{X}$  to which it is connected via the binary tree of transformations (as shown in Figure 1). Therefore a convenient way of making use of each  $I_{clique}$  estimate is to propagate its value back down to its corresponding leaf nodes. The sum of the  $I_{clique}$  values in Equation 3.6 will then build up into an image as the various contributions accumulate at their corresponding leaf nodes. This image will have a blocky appearance because its contributions derive from a binary tree that is connected to the data pixels as shown in Figure 1, but we avoid this problem by forming a further average over an ensemble of closely related relative entropy images that are derived from binary trees that are translated to all possible positions relative to the image data.

### C. Brodatz texture

In Figure 2 we show an image of a Brodatz texture.

This image is  $256 \times 256$  pixels in size, and each pixel is an integer in the interval  $[0, 255]$  (i.e. 1 byte per pixel).

The image is slightly unevenly illuminated and has a fairly low contrast, but nevertheless its statistical properties are almost translation invariant.

In Figure 3 we show a set of relative entropy images, which are all displayed so that large values map to black pixels and small values map to white pixels. Each image is  $256 \times 256$  pixels in size, and is scaled so that its pixel values fill the interval  $[0, 255]$ . Note how the these images become smoother as we progress from Figure 3a to Figure 3f, due to the increasing amount of averaging that occurs over the different placements of the binary tree.

Figure 3e and especially Figure 3f reveal a highly localised anomaly in the original image. Figure 3f corresponds to a length scale of  $8 \times 8$  pixels, which is the approximate size of the fault that is about  $\frac{1}{4}$  of the way down and slightly to the left of centre of Figure 2. The fault does not show up clearly on the other figures in Figure 3, because their characteristic length scales are either too short or too long to be sensitive to the fault.

## V. CONCLUSIONS

In this paper we have summarised the derivation of a useful type of Gibbs distribution by constraining various of its marginal PDFs. In particular, we have presented the expression for the Gibbs distribution that emerges when we first of all develop a hierarchical transformation (e.g. a binary tree) of the original data, and then demand that the Gibbs distribution be consistent with the observed marginal PDFs between sibling nodes of the hierarchy. This type of situation is typical of multi-resolution analyses of data. We have also derived the relative entropy of this Gibbs distribution (relative to the ensemble of data vectors whose PDF it purports to model) to obtain a useful decomposition of the relative entropy as a sum of all of the mutual informations within the sets of sibling nodes of the hierarchy that defined the Gibbs distribution in the first place. It is important to note that each of these mutual informations is measured *laterally* within a layer of the hierarchy, rather than *vertically* between layers of the hierarchy. We have found that the relative entropy is (minus) the sum of the mutual info-

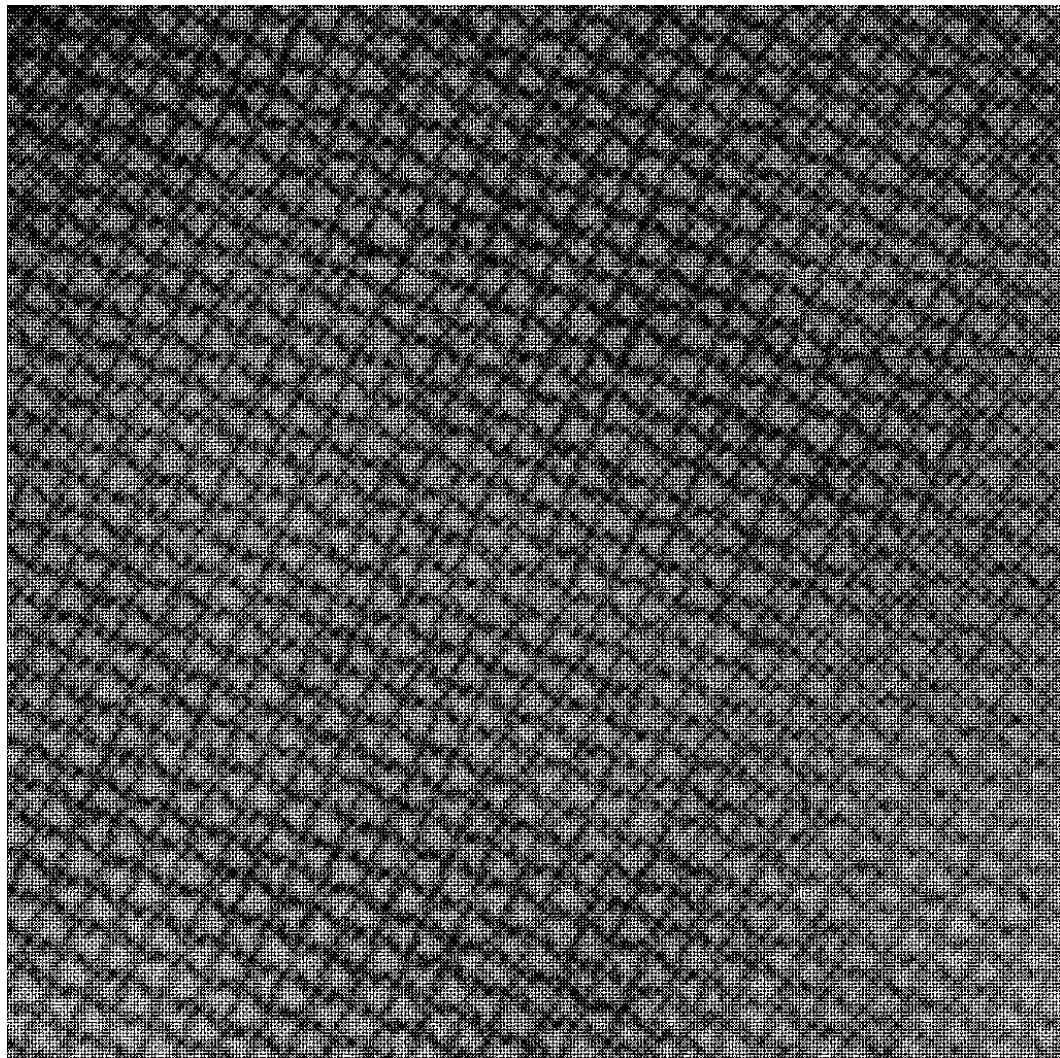


Figure 2: Optical image of a Brodaz texture.

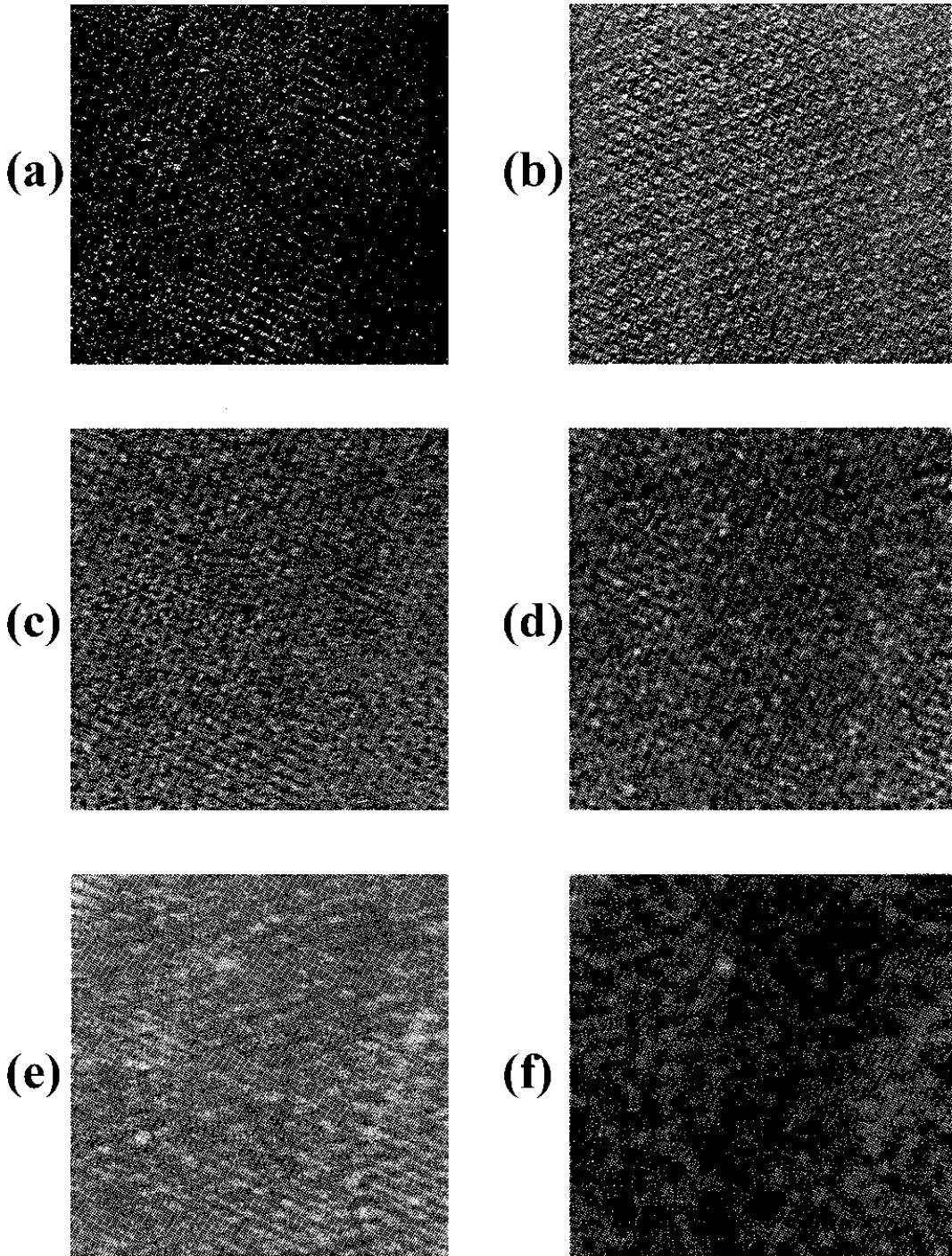


Figure 3: Relative entropy images of Brodatz texture.

mation losses as we pass through the hierarchy, which corresponds to the fact that the nodes of the hierarchy do not preserve all the information about correlations between the clusters of pixel values from which they derive. Finally, we have shown how it is possible to display the various terms in the relative entropy expansion as images.

This type of approach where we investigate statistical correlations in textured (or cluttered) images by devel-

oping hierarchical transformations, and then measuring various statistical properties, is not itself new. However, our Gibbs distribution and the decomposition of its relative entropy as a sum of mutual informations, which then may be used to build up a relative entropy image, is novel. This gives insight into the nature of this type of Gibbs distribution model which may be used to interpret other, more complicated, Gibbs distribution models.

### Appendix A: Maximum entropy derivations

Define the entropy functional

$$H \equiv \int d\mathbf{x}_1 d\mathbf{x}_2 Q_{12}(\mathbf{x}_1, \mathbf{x}_2) \log(Q_{12}(\mathbf{x}_1, \mathbf{x}_2)) \quad (\text{A1})$$

where  $Q_{12}(\mathbf{x}_1, \mathbf{x}_2)$  denotes an estimate of  $P_{12}(\mathbf{x}_1, \mathbf{x}_2)$ . Now define a cost function to impose the required constraints on the marginal PDFs

$$\begin{aligned} C \equiv & H + \int d\mathbf{x}_1 d\mathbf{x}_2 Q_{12}(\mathbf{x}_1, \mathbf{x}_2) (\lambda_1(\mathbf{x}_1) + \lambda_2(\mathbf{x}_2)) \\ & + \int dy_1 dy_2 \mu(y_1, y_2) \left( \int d\mathbf{x}_1 d\mathbf{x}_2 \delta(y_1 - y_1(\mathbf{x}_1)) \delta(y_2 - y_2(\mathbf{x}_2)) Q_{12}(\mathbf{x}_1, \mathbf{x}_2) \right) \end{aligned} \quad (\text{A2})$$

where we use Lagrange multiplier functions  $\lambda_i(\mathbf{x}_i)$  and  $\mu(y_1, y_2)$  to introduce the required constraints on  $Q_{12}(\mathbf{x}_1, \mathbf{x}_2)$ . Functionally differentiating  $C$  yields

$$\frac{\delta C}{\delta Q_{12}(\mathbf{x}_1, \mathbf{x}_2)} = -1 - \log(Q_{12}(\mathbf{x}_1, \mathbf{x}_2)) + \lambda_1(\mathbf{x}_1) + \lambda_2(\mathbf{x}_2) + \mu(y_1(\mathbf{x}_1), y_2(\mathbf{x}_2)) \quad (\text{A3})$$

Set  $\frac{\delta C}{\delta Q_{12}(\mathbf{x}_1, \mathbf{x}_2)} = 0$  to obtain the maximum entropy solution  $Q_{mem,12}(\mathbf{x}_1, \mathbf{x}_2)$  as

$$\begin{aligned} Q_{mem,12}(\mathbf{x}_1, \mathbf{x}_2) &\propto \exp(\lambda_1(\mathbf{x}_1) + \lambda_2(\mathbf{x}_2) + \mu(y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))) \\ &= P_1(\mathbf{x}_1) P_2(\mathbf{x}_2) \frac{p_{12}(y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))}{p_1(y_1(\mathbf{x}_1)) p_2(y_2(\mathbf{x}_2))} \end{aligned} \quad (\text{A4})$$

The fact that this solution is in closed form (i.e. determined, rather than undetermined, Lagrange multipliers) is the most important property of this type of maximum entropy model.

### Appendix B: Mutual information

The basic definition of the mutual information  $I[X; Y]$  between a pair of random variables  $X$  and  $Y$  may be given in three equivalent forms

$$I[X; Y] = H[X] + H[Y] - H[X, Y] \quad (\text{B1})$$

$$I[X; Y] = H[X] - H[X|Y] \quad (\text{B2})$$

$$I[X; Y] = H[Y] - H[Y|X] \quad (\text{B3})$$

where  $H[r]$  denotes the entropy functional applied to the PDF of random variable  $r$ .  $e^{H[r]}$  is the effective number of states which is available to  $r$ , which we define as  $Z_{eff}[r]$ . Thus  $e^{H[X]+H[Y]}$  is the value of  $Z_{eff}[X, Y]$  ignoring any mutual dependencies between  $X$  and  $Y$ , and  $e^{H[X,Y]}$  is the same including dependencies. Thus  $e^{I[X;Y]}$  in Equation B1 measures the reduction in  $Z_{eff}[X, Y]$  which occurs when mutual dependencies between  $X$  and  $Y$  are accounted for. The two alternative definitions in Equation B2 and Equation B3 may be similarly interpreted, although  $X$  and  $Y$  now enter asymmetrically.

The most important property of mutual information is that it is an *objective* measure of correlations, as required of a consistent measure of information. Therefore it treats correlations, anti-correlations (and anything else) equivalently provided that they all lead to the same reduction in  $Z_{eff}[X, Y]$ . Distinctions between various types of correlations made on the basis of measures other than  $Z_{eff}[X, Y]$  are necessarily *subjective*.

We may generalise Equation B1 as follows

$$I[X_1; X_2; \dots; X_n] = \sum_{i=0}^n H[X_i] - H[X_1, X_2, \dots, X_n] \quad (\text{B4})$$

$e^{I[X_1; X_2; \dots; X_n]}$  measures the reduction in  $Z_{eff}[X_1, X_2, \dots, X_n]$  when mutual dependencies amongst the  $X_i$  are taken into account.

We may further generalise these results by partitioning the random variables into mutually exclusive sets

$$\begin{aligned}
\mathbf{X} &= (X_1, X_2, \dots, X_n) \\
\mathbf{x}_1 &= (X_1, X_2, \dots, X_{p_1}) \\
\mathbf{x}_2 &= (X_{p_1+1}, X_{p_1+2}, \dots, X_{p_2}) \\
&\vdots \\
\mathbf{x}_N &= (X_{p_{N-1}+1}, X_{p_{N-1}+2}, \dots, X_n) \\
\mathbf{X} &= (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \\
p_0 &= 0 \\
p_N &= n
\end{aligned} \tag{B5}$$

$$I[X_1; X_2; \dots; X_n] = I[\mathbf{x}_1; \mathbf{x}_2; \dots; \mathbf{x}_N] + \sum_{i=0}^{N-1} I[X_{p_i+1}; X_{p_i+2}; \dots; X_{p_{i+1}}] \tag{B6}$$

Note that we have defined dummy partition boundaries  $p_0$  and  $p_N$  for convenience.

It is convenient to define two further quantities.

$$\begin{aligned}
I_{total}^{(l)} &\equiv I[y_1^{(l)}; y_2^{(l)}; \dots; y_{N_l-1}^{(l)}] \\
&= \int dy_1^{(l)} dy_2^{(l)} \dots dy_{N_l-1}^{(l)} p_{12\dots N_l-1}^{(l)}(y_1^{(l)}, y_2^{(l)}, \dots, y_{N_l-1}^{(l)}) \log \left( \frac{p_{12\dots N_l-1}^{(l)}(y_1^{(l)}, y_2^{(l)}, \dots, y_{N_l-1}^{(l)})}{p_1^{(l)}(y_1^{(l)}) p_2^{(l)}(y_2^{(l)}) \dots p_{N_l-1}^{(l)}(y_{N_l-1}^{(l)})} \right)
\end{aligned} \tag{B7}$$

$$\begin{aligned}
I_{inter}^{(l)} &\equiv I[\mathbf{x}_1^{(l)}; \mathbf{x}_2^{(l)}; \dots; \mathbf{x}_{N_l}^{(l)}] \\
&= \int d\mathbf{x}_1^{(l)} d\mathbf{x}_2^{(l)} \dots d\mathbf{x}_{N_l}^{(l)} P^l(\mathbf{X}^{(l)}) \log \left( \frac{P^l(\mathbf{X}^{(l)})}{P_1^{(l)}(\mathbf{x}_1^{(l)}) P_2^{(l)}(\mathbf{x}_2^{(l)}) \dots P_{N_l}^{(l)}(\mathbf{x}_{N_l}^{(l)})} \right)
\end{aligned} \tag{B8}$$

$I_{total}^{(l)}$  in Equation B7 and  $I_{inter}^{(l)}$  in Equation B8 are the total mutual information between *all* components of layer  $l$ , and the mutual information between *clusters* of components of layer  $l$  respectively, and they satisfy the inequality

$$I_{inter}^{(l)} \geq I_{total}^{(l)} \geq 0 \tag{B9}$$

This can be proved by applying Jensen's inequality to the convex  $\cap$  logarithm function. Equation B9 states that one cannot gain any mutual information by transforming the  $\mathbf{x}_i^{(l)}$  via the functions  $y_i^{(l+1)}(\mathbf{x}_i^{(l)})$ .

We obtain the multilayer generalisation of Equation B6 by defining

$$\begin{aligned}
N_{-1} &= \dim(\mathbf{X}^{(0)}) \\
\mathbf{X}^{(0)} &= (y_1^{(0)}, y_2^{(0)}, \dots, y_{N_{-1}}^{(0)}) \\
I_{total}^{(0)} &= I[y_1^{(0)}; y_2^{(0)}; \dots; y_{N_{-1}}^{(0)}]
\end{aligned} \tag{B10}$$

and

$$\mathbf{x}_i^{(l)} = (x_{i_1}^{(l)}, x_{i_2}^{(l)}, \dots, x_{i_{n_l}}^{(l)}); \quad l = 0, 1, \dots, L-1; \quad i = 1, 2, \dots, N_l \tag{B11}$$

Also define the mutual information between the components *within a cluster* as

$$\begin{aligned}
I_{intra,i}^{(l)} &\equiv I[x_{i_1}^{(l)}; x_{i_2}^{(l)}; \dots; x_{i_{n_l}}^{(l)}] \\
&= \int dx_{i_1}^{(l)} dx_{i_2}^{(l)} \dots dx_{i_{n_l}}^{(l)} P_i^{(l)}(\mathbf{x}_i^{(l)}) \log \left( \frac{P_i^{(l)}(\mathbf{x}_i^{(l)})}{P_{i_1}^{(l)}(\mathbf{x}_{i_1}^{(l)}) P_{i_2}^{(l)}(\mathbf{x}_{i_2}^{(l)}) \dots P_{i_{n_l}}^{(l)}(\mathbf{x}_{i_{n_l}}^{(l)})} \right)
\end{aligned} \tag{B12}$$

Finally we obtain the generalisation of Equation B6 as

$$I_{total}^{(l)} = I_{inter}^{(l)} + \sum_{i=1}^{N_l} I_{intra,i}^{(l)} \quad (\text{B13})$$

- 
- 
- [1] D H Ackley, G E Hinton, and T J Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Science **9** (1985), no. 2, 147–169.
- [2] P Brodatz, *Textures - a photographic album for artists and designers*, Dover, New York, 1966.
- [3] R T Cox, *Probability, frequency and reasonable expectation*, American Journal of Physics **14** (1946), no. 1, 1–13.
- [4] H Jeffreys, *Theory of probability*, Clarendon Press, Oxford, 1939.
- [5] S P Luttrell, *The implications of Boltzmann-type machines for SAR data processing: a preliminary survey*, Memorandum 3815, Royal Signals and Radar Establishment, Malvern, 1985.
- [6] \_\_\_\_\_, *Maximum entropy and Bayesian methods*, ch. The use of Bayesian and entropic methods in neural network theory, pp. 363–370, Kluwer, Dordrecht, 1989.
- [7] \_\_\_\_\_, *Adaptive Cluster Expansion (ACE): a hierarchical Bayesian network*, (1991).

# Self-Supervised Training of Hierarchical Vector Quantisers \*

S P Luttrell<sup>†</sup>

DRA, Electronics Division, Malvern, WORCS, WR14 3PS, UK.

## I. INTRODUCTION

In Luttrell [4, 5, 8, 10] we developed a hierarchical vector quantisation (VQ) model, and in Luttrell [6, 7] we successfully applied it to time series and image compression respectively. The goal of this paper is to derive an extension to this model, in which we backpropagate signals from higher to lower layers of the hierarchy to self-supervise the training of the VQ.

We review the basic properties of our VQ model and its relationship to neural network methods. We extend the model to an ensemble of VQs, and we derive its properties in the limit of a large codebook size (i.e. the continuum limit). Finally, we demonstrate how self-supervision emerges naturally in this type of model.

## II. HIERARCHICAL VECTOR QUANTISATION MODEL

In this section we present a resumé of the VQ theory of Linde et al [3] (the LBG algorithm), and its extension to hierarchical VQs [4, 5, 8, 10]. These extensions are related to the VQ theory of Kumazawa et al [2] for communication over a noisy channel, and to the topographic mapping theory of Kohonen [1] for training self-organising neural networks.

Define  $\mathbf{x}$  as the input data,  $\mathbf{y}$  as the compressed data, and  $\mathbf{x}'$  as the reconstruction of the input data. Define  $\mathbf{y}(\mathbf{x})$  as the compression operation  $\mathbf{x} \rightarrow \mathbf{y}$ , and  $\mathbf{x}'(\mathbf{y})$  as the reconstruction operation  $\mathbf{y} \rightarrow \mathbf{x}'$ , which yields overall  $\mathbf{x}' = \mathbf{x}'(\mathbf{y}(\mathbf{x}))$ . Also define the probability density function (PDF)  $P(\mathbf{x})$  of possible input data  $\mathbf{x}$ . Note that throughout this paper we use the notation  $P(\dots)$  to denote a PDF, whose functional form may be deduced from its context.

We may combine these quantities to obtain the *average*  $L_2$  (i.e. Euclidean) distortion  $D_1$ , as follows

$$D_1 = \int d\mathbf{x} P(\mathbf{x}) \|\mathbf{x}'(\mathbf{y}(\mathbf{x})) - \mathbf{x}\|^2 \quad (2.1)$$

The VQ whose (continuum limit) codebook is defined by the pair of functions  $(\mathbf{y}(\mathbf{x}), \mathbf{x}'(\mathbf{y}))$  can be optimised

by minimising  $D_1$  with respect to variations of  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$ .

A more general form of Equation 2.1 that gives the average  $L_2$  distortion for a VQ with a noisy communication channel [2, 5] is

$$D_2 = \int d\mathbf{x} P(\mathbf{x}) \int d\mathbf{y}' \pi(\mathbf{y}' - \mathbf{y}(\mathbf{x})) \|\mathbf{x}'(\mathbf{y}') - \mathbf{x}\|^2 \quad (2.2)$$

In Equation 2.2 we assume that  $\mathbf{y}' = \mathbf{y}(\mathbf{x}) + \mathbf{n}$ , where  $\mathbf{n}$  is an independent random variable with PDF  $\pi(\mathbf{n})$ , so  $P(\mathbf{x}, \mathbf{n}) = P(\mathbf{x}) \pi(\mathbf{n})$ .

We functionally differentiate  $D_2$  to calculate the zero(s) of  $\frac{\delta D_2}{\delta \mathbf{y}(\mathbf{x})}$  and  $\frac{\delta D_2}{\delta \mathbf{x}'(\mathbf{y})}$ , which yields (see [5, 10] for the details)

$$\mathbf{y}(\mathbf{x}) = \arg \min_{\mathbf{y}} \int d\mathbf{y}' \pi(\mathbf{y}' - \mathbf{y}) \|\mathbf{x}'(\mathbf{y}') - \mathbf{x}\|^2 \quad (2.3)$$

$$\text{batch : } \mathbf{x}'(\mathbf{y}) = \frac{\int d\mathbf{x} P(\mathbf{x}) \pi(\mathbf{y} - \mathbf{y}(\mathbf{x})) \mathbf{x}}{\int d\mathbf{x} P(\mathbf{x}) \pi(\mathbf{y} - \mathbf{y}(\mathbf{x}))}$$

$$\text{continuous : } \Delta \mathbf{x}'(\mathbf{y}) = \varepsilon \pi(\mathbf{y} - \mathbf{y}(\mathbf{x})) (\mathbf{x} - \mathbf{x}'(\mathbf{y})) \quad (2.4)$$

In the noiseless case  $\pi(\mathbf{n}) = \delta(\mathbf{n})$  Equation 2.3 specifies that the optimum  $\mathbf{y}(\mathbf{x})$  is a nearest neighbour prescription, i.e. given  $\mathbf{x}$  select as  $\mathbf{y}(\mathbf{x})$  the  $\mathbf{y}$  that minimises  $\|\mathbf{x}'(\mathbf{y}') - \mathbf{x}\|^2$ . This nearest neighbour property is preserved in lowest order when  $\pi(\mathbf{n}) \neq \delta(\mathbf{n})$ , provided that the noise is zero mean [7]. Equation 2.4 further specifies two methods of updating  $\mathbf{x}'(\mathbf{y})$ . Firstly, the *batch* update prescription in Equation 2.4 is equivalent to one cycle of the LBG algorithm [3]. Secondly, the *continuous* update prescription in Equation 2.4 is identical to the topographic mapping training algorithm [1], so  $\pi(\mathbf{n})$  can be interpreted as a topographic neighbourhood function.

In Luttrell [11] we derive the asymptotic density of code vectors of topographic mappings trained using the minimum distortion prescription of Equation 2.3, and we find that it is independent of  $\pi(\mathbf{n})$ , assuming *scalar* quantisation and mild monotonicity constraints on  $\pi(\mathbf{n})$ . In Luttrell [9] we present an informal derivation of this result for the *vector* quantisation case. Because  $\pi(\mathbf{n}) = \delta(\mathbf{n})$  is a standard VQ, this result leads to some of the asymptotic properties of VQs being the same as those of topographic mappings, provided that we use the minimum distortion rather than the nearest neighbour prescription.

Another generalisation of Equation 2.1 is to create a set of nested VQs. For instance, in Figure 1 we show a two-stage VQ. The components of the input pair  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$  are separately transformed to yield the corresponding components of the pair  $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)$ , which

\*Typeset in LATEX on May 11, 2019.

This appeared in Proceedings of the 2<sup>nd</sup> IEE International Conference on Artificial Neural Networks, Bournemouth, 5-9, 1991. © British Crown Copyright 1991/MOD. Published with the permission of the Controller of Her Britannic Majesty's Stationery Office.

<sup>†</sup>Electronic address: luttrell%uk.mod.hermes@relay.mod.uk

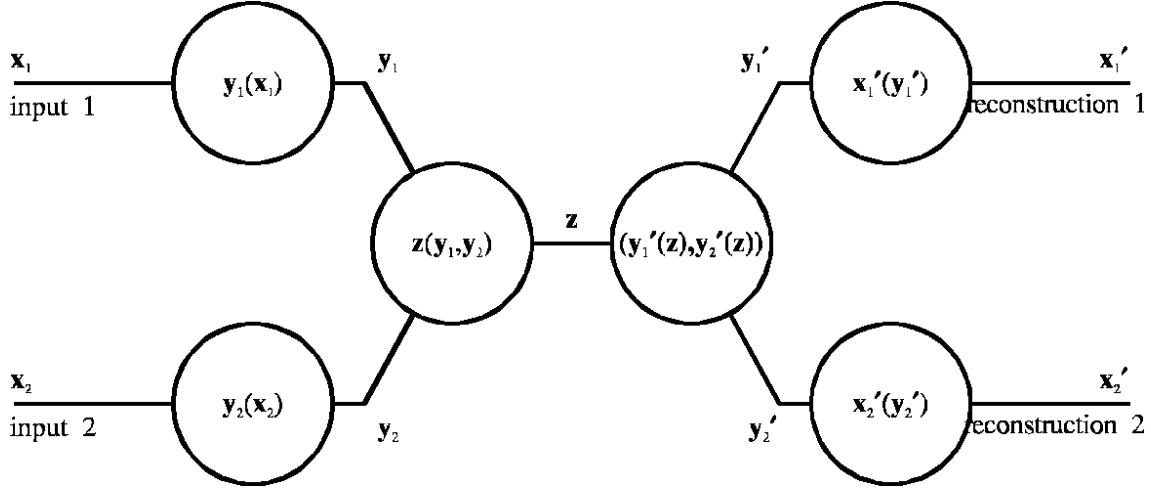


Figure 1: Two-stage vector quantisation. Two channels  $\mathbf{x}_1 \rightarrow \mathbf{y}_1 \cdots \mathbf{y}_1' \rightarrow \mathbf{x}_1'$  and  $\mathbf{x}_2 \rightarrow \mathbf{y}_2 \cdots \mathbf{y}_2' \rightarrow \mathbf{x}_2'$  are coupled through a common channel  $(\mathbf{y}_1, \mathbf{y}_2) \rightarrow \mathbf{z} \rightarrow (\mathbf{y}_1', \mathbf{y}_2')$ .

is then input to a standard VQ, whose output components  $\mathbf{y}' = (\mathbf{y}_1', \mathbf{y}_2')$  are used separately to reconstruct the corresponding components of  $\mathbf{x}' = (\mathbf{x}_1', \mathbf{x}_2')$ . In [5, 10] we present a method of training the channels  $\mathbf{x}_1 \rightarrow \mathbf{y}_1 \cdots \mathbf{y}_1' \rightarrow \mathbf{x}_1'$  and  $\mathbf{x}_2 \rightarrow \mathbf{y}_2 \cdots \mathbf{y}_2' \rightarrow \mathbf{x}_2'$  separately, by modelling them as a pair of noisy channel VQs (as in Equation 2.2). Although this scheme assumes (incorrectly) that the channels do not mutually interfere, it nevertheless leads to useful results. We may generalise Figure 1 by creating a multi-stage structure of nested VQs, which may be used for time series and image compression, as we demonstrate in [6, 7].

### III. ENSEMBLE AVERAGE TRAINING

In this section we consider the effect that mutual channel coupling has on optimising a nested VQ.

In Figure 2 we show an ensemble average version of Figure 1, where we take the average over realisations of the nested VQ  $(\mathbf{y}_1, \mathbf{y}_2) \rightarrow \mathbf{z} \rightarrow (\mathbf{y}_1', \mathbf{y}_2')$ . The statistical properties of the ensemble in Figure 2 are expressed using a transition probability  $P(\mathbf{y}_1', \mathbf{y}_2' | \mathbf{y}_1, \mathbf{y}_2)$

$$\begin{aligned} P(\mathbf{y}_1', \mathbf{y}_2' | \mathbf{y}_1, \mathbf{y}_2) &= \int d[\mathbf{z}(\mathbf{y}_1, \mathbf{y}_2)] d[\mathbf{y}_1'(\mathbf{z})] d[\mathbf{y}_2'(\mathbf{z})] \\ &\quad P(\mathbf{y}_1'(\mathbf{z}), \mathbf{y}_2'(\mathbf{z}), \mathbf{z}(\mathbf{y}_1, \mathbf{y}_2)) \delta(\mathbf{y}_1' - \mathbf{y}_1'(\mathbf{z}(\mathbf{y}_1, \mathbf{y}_2))) \delta(\mathbf{y}_2' - \mathbf{y}_2'(\mathbf{z}(\mathbf{y}_1, \mathbf{y}_2))) \end{aligned} \quad (3.1)$$

where  $d[\mathbf{z}(\mathbf{y}_1, \mathbf{y}_2)] d[\mathbf{y}_1'(\mathbf{z})] d[\mathbf{y}_2'(\mathbf{z})]$  is a volume element in function space,  $P(\mathbf{y}_1'(\mathbf{z}), \mathbf{y}_2'(\mathbf{z}), \mathbf{z}(\mathbf{y}_1, \mathbf{y}_2))$  is a PDF which specifies a measure in function space, and  $\delta(\mathbf{y}_1' - \mathbf{y}_1'(\mathbf{z}(\mathbf{y}_1, \mathbf{y}_2))) \delta(\mathbf{y}_2' - \mathbf{y}_2'(\mathbf{z}(\mathbf{y}_1, \mathbf{y}_2)))$  is the contribution to  $P(\mathbf{y}_1', \mathbf{y}_2' | \mathbf{y}_1, \mathbf{y}_2)$  from each  $(\mathbf{y}_1, \mathbf{y}_2) \rightarrow \mathbf{z} \rightarrow (\mathbf{y}_1', \mathbf{y}_2')$  member of the ensemble. The ensemble average approach is appropriate in situations where the  $(\mathbf{y}_1, \mathbf{y}_2) \rightarrow \mathbf{z} \rightarrow (\mathbf{y}_1', \mathbf{y}_2')$  transformation is continuously updated by a training algorithm. Ideally the  $\mathbf{x}_1 \rightarrow \mathbf{y}_1 \cdots \mathbf{y}_1' \rightarrow \mathbf{x}_1'$  and  $\mathbf{x}_2 \rightarrow \mathbf{y}_2 \cdots \mathbf{y}_2' \rightarrow \mathbf{x}_2'$  transformations should be optimised to adapt to the changes to the  $(\mathbf{y}_1, \mathbf{y}_2) \rightarrow \mathbf{z} \rightarrow (\mathbf{y}_1', \mathbf{y}_2')$  transforma-

tion, but this is time consuming. Rather, it is better to adapt to the properties of the *ensemble* of  $(\mathbf{y}_1, \mathbf{y}_2) \rightarrow \mathbf{z} \rightarrow (\mathbf{y}_1', \mathbf{y}_2')$  transformations that might occur.

In [5, 10] we did not introduce  $P(\mathbf{y}_1'(\mathbf{z}), \mathbf{y}_2'(\mathbf{z}), \mathbf{z}(\mathbf{y}_1, \mathbf{y}_2))$  explicitly, but directly modelled  $P(\mathbf{y}_1', \mathbf{y}_2' | \mathbf{y}_1, \mathbf{y}_2)$  by introducing additive noise independently into the  $\mathbf{x}_1 \rightarrow \mathbf{y}_1 \cdots \mathbf{y}_1' \rightarrow \mathbf{x}_1'$  and  $\mathbf{x}_2 \rightarrow \mathbf{y}_2 \cdots \mathbf{y}_2' \rightarrow \mathbf{x}_2'$  channels. Because of the independence assumption this approach is crude, but it is demonstrably effective and useful [6, 7]. Using  $P(\mathbf{y}_1', \mathbf{y}_2' | \mathbf{y}_1, \mathbf{y}_2)$  the expression for the overall distortion in Figure 2 is given by

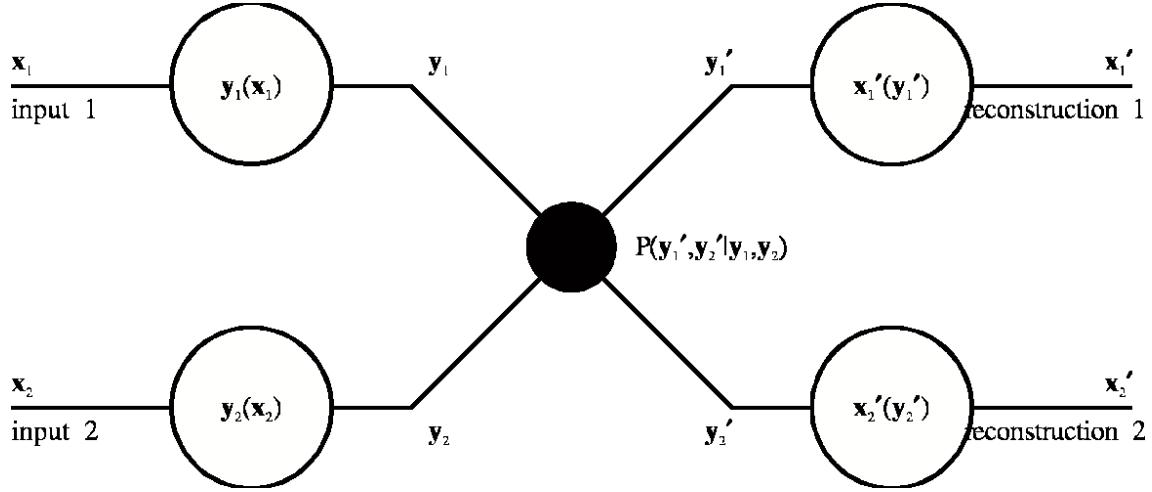


Figure 2: Ensemble average two-stage vector quantisation.  $P(y_1', y_2' | y_1, y_2)$  models the distortion due to the ensemble average of feasible  $(y_1, y_2) \rightarrow z \rightarrow (y_1', y_2')$ .

$$D_3 = \int d\mathbf{x}_1 d\mathbf{x}_2 P(\mathbf{x}_1, \mathbf{x}_2) \int dy_1' P(y_1' | y_1(\mathbf{x}_1), y_2(\mathbf{x}_2)) \|x_1'(y_1') - \mathbf{x}_1\|^2 + (1 \leftrightarrow 2) \quad (3.2)$$

where  $P(y_1' | y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))$  is a marginal PDF obtained by integrating  $P(y_1', y_2' | y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))$  over  $y_2'$ , and  $(1 \leftrightarrow 2)$  indicates that there is an analogous term for channel 2. We optimise the nested VQ by minimising  $D_3$  to obtain

$$(y_1(\mathbf{x}_1), y_2(\mathbf{x}_2)) = \arg \min_{(y_1, y_2)} \left( \int dy_1' P(y_1' | y_1, y_2) \|x_1'(y_1') - \mathbf{x}_1\|^2 + (1 \leftrightarrow 2) \right) \quad (3.3)$$

$$\begin{aligned} \text{batch : } \mathbf{x}_k'(y_k) &= \frac{\int d\mathbf{x}_1 d\mathbf{x}_2 P(\mathbf{x}_1, \mathbf{x}_2) P(y_k | y_1(\mathbf{x}_1), y_2(\mathbf{x}_2)) \mathbf{x}_k}{\int d\mathbf{x}_1 d\mathbf{x}_2 P(\mathbf{x}_1, \mathbf{x}_2) P(y_k | y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))} \\ \text{continuous : } \Delta \mathbf{x}_k'(y_k) &= \varepsilon P(y_k | y_1(\mathbf{x}_1), y_2(\mathbf{x}_2)) (\mathbf{x}_k - \mathbf{x}_k'(y_k)) \end{aligned} \quad (3.4)$$

which should be compared with Equation 2.3 and Equation 2.4. Note how Equation 3.3 specifies a minimum distortion prescription in which  $y_1(\mathbf{x}_1)$  and  $y_2(\mathbf{x}_2)$  are *simultaneously* optimised, using  $P(y_1' | y_1, y_2)$  and  $P(y_2' | y_1, y_2)$  instead of  $\pi(y' - y)$ . We usually approximate this by using a nearest neighbour prescription, although see [11] for an exception to this.

In Figure 3 we represent diagrammatically in  $(y_1, y_2)$ -space (and  $(y_1', y_2')$ -space) the various terms of Equation 3.2. We represent the contours of a typical  $P(y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))$ , a typical  $P(y_1', y_2' | y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))$ , and the profiles of its two marginals  $P(y_1' | y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))$  and  $P(y_2' | y_1(\mathbf{x}_1), y_2(\mathbf{x}_2))$ . These marginals have a shape that depends on  $(\mathbf{x}_1, \mathbf{x}_2)$ , which therefore mutually couples the contributions to the distortion in Equation 3.2 arising from the two transformations  $\mathbf{x}_1 \rightarrow y_1 \cdots y_1' \rightarrow \mathbf{x}_1'$  and  $\mathbf{x}_2 \rightarrow y_2 \cdots y_2' \rightarrow \mathbf{x}_2'$ . It is both pleasing and economical that the ensemble average nested VQ in Figure 2 automatically deter-

mines the topographic neighbourhood functions for its  $\mathbf{x}_1 \rightarrow y_1 \cdots y_1' \rightarrow \mathbf{x}_1'$  and  $\mathbf{x}_2 \rightarrow y_2 \cdots y_2' \rightarrow \mathbf{x}_2'$  transformations, thus eliminating the need to introduce them by hand.

#### IV. ANALYTICALLY SOLVABLE QUANTISATION MODEL

In this section we present an analytically solvable model of the ensemble distortion  $P(y_1', y_2' | y_1, y_2)$  shown in Figure 3. This is the main result that we present in this paper.

##### A. Code vector density

The definition of the ensemble average in Equation 3.1 is controlled by the  $P(y_1'(\mathbf{z}), y_2'(\mathbf{z}), \mathbf{z}(y_1, y_2))$  factor, which specifies those transformations that we

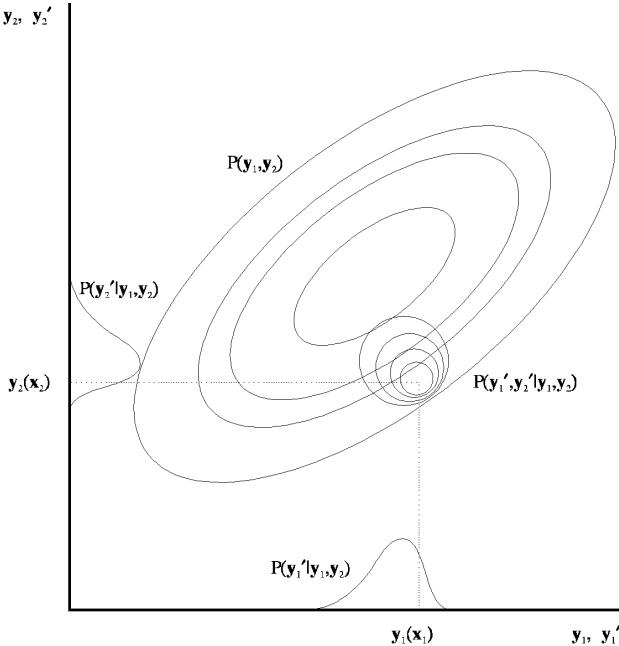


Figure 3: The marginal PDFs  $P(y_1'|y_1, y_2)$  and  $P(y_2'|y_1, y_2)$  of the ensemble average distortion  $P(y_1', y_2'|y_1, y_2)$  determine the topographic neighbourhood functions for optimising the  $x_1 \rightarrow y_1 \cdots y_1' \rightarrow x_1'$  and  $x_2 \rightarrow y_2 \cdots y_2' \rightarrow x_2'$  channels.

consider to be likely outcomes of the optimisation of  $(y_1, y_2) \rightarrow z \rightarrow (y_1', y_2')$ . In practice, the global minimum of  $D_3$  has many feasible solutions for  $(y_1'(z), y_2'(z), z(y_1, y_2))$ , and there are many other solutions lying close to this minimum. The problem of deriving  $P(y_1'(z), y_2'(z), z(y_1, y_2))$  would seem to be intractable because of the complicated correlations that must exist between the feasible solutions for  $y_1'(z)$ ,  $y_2'(z)$  and  $z(y_1, y_2)$ .

If the number of code vectors  $(y_1'(z), y_2'(z))$  in the  $(y_1, y_2) \rightarrow z \rightarrow (y_1', y_2')$  codebook is very large, we may calculate  $P(y_1', y_2'|y_1, y_2)$  directly. Thus we model the ensemble properties of the codebook by defining  $\rho(y_1, y_2)$ , which specifies the density of code vectors  $(y_1'(z), y_2'(z))$  in  $(y_1, y_2)$ -space.

### B. Transition probability: integral equation

Note that we usually write  $\rho(\mathbf{y})$  and  $P(\mathbf{y}'|\mathbf{y})$  to stand for the density of code vectors  $\rho(y_1, y_2)$  and the transition probability  $P(y_1', y_2'|y_1, y_2)$  respectively.

In Figure 4 we compare the nearest neighbour prescription for a *single* VQ with that for an *ensemble* of VQs. In Figure 4a we show an input vector (represented by a cross) and the known positions of the code vectors of a single VQ. The nearest neighbour can be located by expanding a circle centred on the input vector until it grazes the nearest code vector, as shown. In Figure 4b we show the ensemble version of the same diagram, in

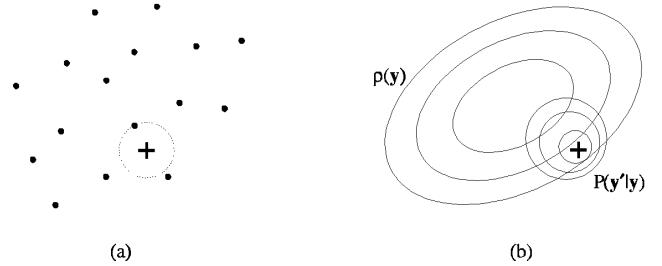


Figure 4: (a) Determining the nearest neighbour code vector position for a single vector quantiser. (b) Determining the PDF  $P(y'|y)$  of the nearest neighbour code vector position from the code vector density  $\rho(y)$  of an ensemble of vector quantisers.

which the precise code vector positions are unknown, so there is a distribution  $P(y'|y)$  of possible nearest neighbour locations.

For completeness, we discuss here the intermediate case where the positions of the code vectors are partially known. The most important way of acquiring partial knowledge is to note the positions of the nearest neighbour code vectors during training. However, such knowledge must be continuously updated because migration of the code vector positions gradually erases any memory of their earlier positions. Partial knowledge lies between the extremes of Figure 4a and Figure 4b, and its analysis is very complicated. We choose to analyse the extreme case in Figure 4b because it underestimates, rather than overestimates, the knowledge that is available.

We may develop an integral equation that relates  $P(y'|y)$  to  $\rho(y)$  as follows.

$$P(y'|y) \delta y' = \left( 1 - \int_{\|\xi-y\| \leq \|y'-y\|} d\xi P(\xi|y) \right) \rho(y') \delta y' \quad (4.1)$$

where the first term on the right hand side is the probability that there is no nearest neighbour code vector in the sphere of radius  $\|y' - y\|$  centred on  $y$ , and the second term is the probability of finding a code vector in the volume  $\delta y'$  at  $y'$ . The product of these two terms gives the probability of finding the nearest neighbour code vector in the volume  $\delta y'$  at  $y'$ .

### C. Transition probability: constant code vector density case

We now solve Equation 4.1 for the case  $\rho(y) = \rho_0 = \text{constant}$ . The nearest neighbour code vector is then equally likely to lie in any direction from  $y$ , so  $P(y'|y)$  must be a function only of the radial distance  $\|y' - y\|$ ,

which gives

$$P(\|\mathbf{y}' - \mathbf{y}\|) = \left( 1 - \int_{\|\boldsymbol{\xi} - \mathbf{y}\| \leq \|\mathbf{y}' - \mathbf{y}\|} d\boldsymbol{\xi} P(\|\boldsymbol{\xi} - \mathbf{y}\|) \right) \rho_0 \quad (4.2)$$

where  $\|\mathbf{y}' - \mathbf{y}\|^2 \equiv (\mathbf{y}' - \mathbf{y})^T \cdot (\mathbf{y}' - \mathbf{y})$ . The integrand is spherically symmetric so we may use the transformation

$$\int_{\|\boldsymbol{\xi} - \mathbf{y}\| \leq \|\mathbf{y}' - \mathbf{y}\|} d\boldsymbol{\xi} P(\|\boldsymbol{\xi} - \mathbf{y}\|) = \alpha_N \int_{\|\boldsymbol{\xi}\| \leq \|\mathbf{y}' - \mathbf{y}\|} d\|\boldsymbol{\xi}\| \|\boldsymbol{\xi}\|^{N-1} P(\|\boldsymbol{\xi}\|) \quad (4.3)$$

where  $\alpha_N$  is a constant deriving from the angular integration in  $N$  dimensions. Differentiate Equation 4.2 with respect to the upper limit  $\|\mathbf{y}' - \mathbf{y}\|$  of the  $\|\boldsymbol{\xi}\|$  integration to yield

$$\frac{dP(\|\mathbf{y}' - \mathbf{y}\|)}{d\|\mathbf{y}' - \mathbf{y}\|} = -\alpha_N \rho_0 \|\mathbf{y}' - \mathbf{y}\|^{N-1} P(\|\mathbf{y}' - \mathbf{y}\|) \quad (4.4)$$

and integrate to yield finally

$$P(\|\mathbf{y}' - \mathbf{y}\|) = P_0 \exp\left(-\frac{\alpha_N \rho_0 \|\mathbf{y}' - \mathbf{y}\|^N}{N}\right) \quad (4.5)$$

where  $P_0$  should be adjusted to ensure that  $P(\|\mathbf{y}' - \mathbf{y}\|)$  is normalised correctly. The  $N = 2$  case reduces to a Gaussian distribution with  $P_0 = \rho_0$ .

#### D. Transition probability: variable code vector density case

We now extend the previous results to the case

$$\rho(\mathbf{y}') = \rho(\mathbf{y}) + (\mathbf{y}' - \mathbf{y})^T \cdot \nabla \rho(\mathbf{y}) \quad (4.6)$$

which is a first order Taylor expansion of  $\rho(\mathbf{y}')$  about the point  $\mathbf{y}' = \mathbf{y}$ . We anticipate that the first order expansion of  $P(\mathbf{y}'|\mathbf{y})$  has the form of Equation 4.5 with an extra factor to account for the angular dependence in Equation 4.6.

$$P(\mathbf{y}'|\mathbf{y}) \approx P_0(\mathbf{y}) \left( 1 + (\mathbf{y}' - \mathbf{y})^T \cdot \mathbf{a}(\mathbf{y}) \right) \exp\left(-\frac{\alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^N}{N}\right) \quad (4.7)$$

where  $\mathbf{a}(\mathbf{y})$  has to be determined. Insert Equation 4.7 into Equation 4.1 and differentiate with respect to  $\mathbf{y}'$  to obtain  $\mathbf{a}(\mathbf{y}) = \frac{\nabla \rho(\mathbf{y})}{\rho(\mathbf{y})}$  (ignoring terms  $\mathcal{O}\left(\frac{\|\nabla \rho(\mathbf{y})\|^2}{\rho(\mathbf{y})^2}\right)$ ). Finally, we may write

$$P(\mathbf{y}'|\mathbf{y}) \approx P_0(\mathbf{y}) \left( 1 + \frac{(\mathbf{y}' - \mathbf{y})^T \cdot \nabla \rho(\mathbf{y})}{\rho(\mathbf{y})} \right) \exp\left(-\frac{\alpha_N \rho(\mathbf{y}) \|\mathbf{y}' - \mathbf{y}\|^N}{N}\right) \quad (4.8)$$

In practical applications we must estimate  $\rho(\mathbf{y})$  and  $\nabla \rho(\mathbf{y})$  from limited information. Clearly, it is not correct to approximate  $\rho(\mathbf{y})$  as a mixture of Dirac deltas located at each of the current code vector positions, because this would provide knowledge of the exact VQ, rather than its ensemble properties. We suggest that a local average of this mixture should be made in order to eradicate precise knowledge of the code vector positions.

Strictly speaking, Equation 4.8 does not specify a valid probability distribution because it yields a negative prob-

ability when  $(\mathbf{y}' - \mathbf{y})^T \cdot \frac{\nabla \rho(\mathbf{y})}{\rho(\mathbf{y})} < -1$ . However, this result is the leading order term in a Taylor expansion about  $\mathbf{y}' = \mathbf{y}$  (see Equation 4.6), therefore we implicitly assume  $\|(\mathbf{y}' - \mathbf{y})^T \cdot \frac{\nabla \rho(\mathbf{y})}{\rho(\mathbf{y})}\| \ll 1$ . The effect of the  $1 + (\mathbf{y}' - \mathbf{y})^T \cdot \frac{\nabla \rho(\mathbf{y})}{\rho(\mathbf{y})}$  term is to relocate the maximum of  $P(\mathbf{y}'|\mathbf{y})$  from its original position at  $\mathbf{y}' = \mathbf{y}$  (see Equation

4.5) to a new position given by

$$\mathbf{y}' \approx \mathbf{y} + \left( \frac{\|\nabla \rho(\mathbf{y})\|}{\alpha_N \rho(\mathbf{y})^2} \right)^{\frac{1}{N-1}} \frac{\nabla \rho(\mathbf{y})}{\|\nabla \rho(\mathbf{y})\|} \quad (4.9)$$

The direction of shift is consistent with the bias in  $P(\mathbf{y}'|\mathbf{y})$  that we show in Figure 3 and Figure 4.

Finally, we marginalise the joint distribution  $P(\mathbf{y}_1', \mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  in Equation 4.8 in order to calculate  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  (which are needed in Equation 3.3 and Equation 3.4). For the 2-dimensional case ( $N = 2$ ,  $\alpha_2 = 2\pi$ ,  $(\mathbf{y}_1, \mathbf{y}_2) \rightarrow (\mathbf{y}_1', \mathbf{y}_2')$ ) this is easy because the exponential factors are Gaussians, leading to the result

$$P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2) \approx P_0(\mathbf{y}_1, \mathbf{y}_2) \left( 1 + \frac{(\mathbf{y}_1' - \mathbf{y}_1)}{\rho(\mathbf{y}_1, \mathbf{y}_2)} \frac{\partial \rho(\mathbf{y}_1, \mathbf{y}_2)}{\partial \mathbf{y}_1} \right) \exp(-\pi \rho(\mathbf{y}_1, \mathbf{y}_2) (\mathbf{y}_1' - \mathbf{y}_1)^2) \quad (4.10)$$

with an analogous result for  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$ . These results may be used to model the marginals in Figure 3.

Recall that  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  serve as topographic neighbourhood functions for optimising the  $\mathbf{x}_1 \rightarrow \mathbf{y}_1 \cdots \mathbf{y}_1' \rightarrow \mathbf{x}_1'$  and  $\mathbf{x}_2 \rightarrow \mathbf{y}_2 \cdots \mathbf{y}_2' \rightarrow \mathbf{x}_2'$  transformations. In the ensemble average model, these neighbourhood functions emerge naturally from the ensemble properties of  $(\mathbf{y}_1, \mathbf{y}_2) \rightarrow \mathbf{z} \rightarrow (\mathbf{y}_1', \mathbf{y}_2')$ , so we do not need to supply them manually. We call this property *self-supervision*, because the topographic neighbourhood functions that are required by one part of the network are automatically supplied by another part of the network.

### E. Optimisation of the joint output density

The update prescription depends on the biased marginals  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$ , which causes a migration of  $P(\mathbf{y}_1, \mathbf{y}_2)$  as shown in Figure 5, where the arrows represent the direction of migration. The widths of the marginals  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$  determine the widths of the vertical and horizontal bands of  $P(\mathbf{y}_1, \mathbf{y}_2)$  that are affected.

Strictly speaking, the change to  $P(\mathbf{y}_1, \mathbf{y}_2)$  is not restricted entirely to the vicinity of the two regions indicated in Figure 5. For instance, the movement of the code vectors in the topographic neighbourhood of  $\mathbf{y}_1(\mathbf{x})$  and  $\mathbf{y}_2(\mathbf{x})$  can change the shape of the quantisation cells of other code vectors, which, in turn, causes other changes to  $P(\mathbf{y}_1, \mathbf{y}_2)$ . However, this is a second order effect.

We see from Figure 5 that the net migration averaged over all inputs has the effect of *squeezing* the  $P(\mathbf{y}_1, \mathbf{y}_2)$  distribution. This inward pressure is counterbalanced by the *stretching* tendency of each marginal  $P(\mathbf{y}_1)$  and  $P(\mathbf{y}_2)$  to become approximately uniform, as normally occurs in VQs. Informally, we can interpret this competition as tending to maximise the mutual information  $I[\mathbf{y}_1; \mathbf{y}_2]$  between  $\mathbf{y}_1$  and  $\mathbf{y}_2$ . Thus  $I[\mathbf{y}_1; \mathbf{y}_2] = H[\mathbf{y}_1] + H[\mathbf{y}_2] - H[\mathbf{y}_1, \mathbf{y}_2] \geq 0$ , where  $H[\dots]$  is the entropy of its argument, and stretching causes  $H[\mathbf{y}_1]$  and

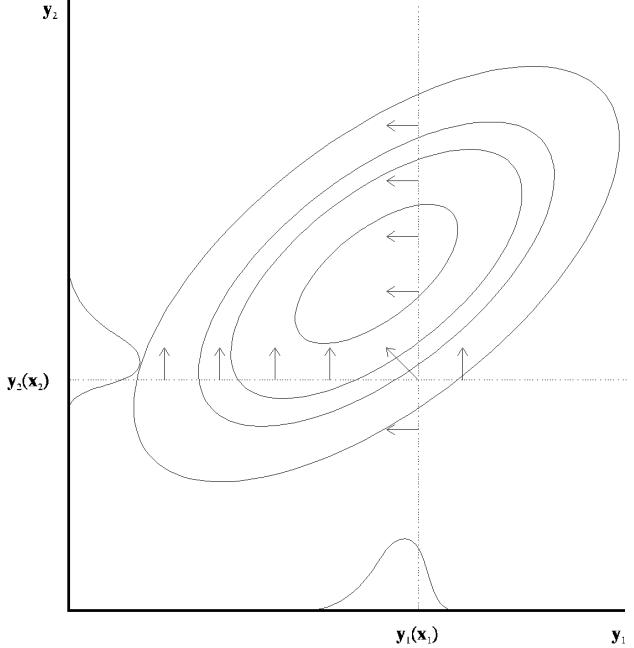


Figure 5: Migration of the PDF  $P(\mathbf{y}_1, \mathbf{y}_2)$  due to the self-supervision effect of the marginal PDFs  $P(\mathbf{y}_1'|\mathbf{y}_1, \mathbf{y}_2)$  and  $P(\mathbf{y}_2'|\mathbf{y}_1, \mathbf{y}_2)$ , which are the topographic neighbourhood functions for optimising the  $\mathbf{x}_1 \rightarrow \mathbf{y}_1 \cdots \mathbf{y}_1' \rightarrow \mathbf{x}_1'$  and  $\mathbf{x}_2 \rightarrow \mathbf{y}_2 \cdots \mathbf{y}_2' \rightarrow \mathbf{x}_2'$  channels. Compare Figure 3.

$H[\mathbf{y}_2]$  to increase, whereas squeezing causes  $H[\mathbf{y}_1, \mathbf{y}_2]$  to decrease, hence  $I[\mathbf{y}_1; \mathbf{y}_2]$  tends to increase, although this is not absolutely guaranteed.

Mutual information can be used as our basic optimisation criterion instead of  $L_2$  distortion minimisation. An example of this approach and its relationship to the optimisation of a novel class of hierarchical Gibbs distributions can be found in Luttrell [12].

## V. CONCLUSIONS

This paper makes two main points. Firstly, we introduce an ensemble hierarchical VQ model, where the lower layers of the hierarchy use knowledge of the *ensemble* properties of the higher layers of the hierarchy, rather than their *exact* properties. Secondly, we present an analytically solvable model, where the ensemble properties of the higher layers give rise to data dependent topographic neighbourhood functions, which influence the

optimisation of the lower layers. We call this effect *self-supervision*.

We may express the concept of self-supervision in a language that is more appropriate to unsupervised neural networks. The higher layers of a network respond to longer space/time scale features in the input data, and issue backpropagating signals that cause the lower layers to process the data in such a way that overall the network responds more strongly to the input data.

- [1] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [2] H Kumazawa, M Kasahara, and T Namekawa, *A construction of vector quantisers for noisy channels*, Electronics and Engineering in Japan **67** (1984), no. 4, 39–47.
- [3] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [4] S P Luttrell, *Self organising multilayer topographic mappings*, Proceedings of IEEE conference on neural networks (San Diego), IEEE, 1988, pp. 93–100.
- [5] ———, *Hierarchical self-organising networks*, Proceedings of IEE Conference on Artificial Neural Networks (London), IEE, 1989, pp. 2–6.
- [6] ———, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
- [7] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
- [8] ———, *Self-organisation: a derivation from first principles of a class of learning algorithms*, Proceedings of IJCNN international joint conference on neural networks (Washington DC), IEEE, 1989, pp. 495–498.
- [9] ———, *Asymptotic code vector density in topographic vector quantisers*, Memorandum 4392, Royal Signals and Radar Establishment, 1990.
- [10] ———, *Derivation of a class of training algorithms*, IEEE Transactions on Neural Networks **1** (1990), no. 2, 229–232.
- [11] ———, *Code vector density in topographic mappings: scalar case*, IEEE Transactions on Neural Networks **2** (1991), no. 4, 427–436.
- [12] ———, *A hierarchical network for clutter and texture modelling*, Proceedings of SPIE conference on adaptive signal processing (San Diego), SPIE, 1991, pp. 518–528.



# Adaptive Bayesian Networks \*

S P Luttrell<sup>†</sup>

Defence Research Agency, St Andrews Rd, Malvern, WORCS, WR14 3PS

The theory of adaptive Bayesian networks is summarised. A detailed discussion of the Adaptive Cluster Expansion (ACE) network is presented. ACE is a scalable Bayesian network designed specifically for high-dimensional applications, such as image processing.

## I. INTRODUCTION

In the first half of this paper the theory of adaptive Bayesian networks is presented. This type of network performs Bayesian inference [3, 5] using a probability density function (PDF) model that it learns during a training programme. Because a large number of degrees of freedom (or hidden variables) need to be integrated out, Bayesian networks can be computationally expensive to simulate [1]. In the second half of this paper the Adaptive Cluster Expansion (ACE) is offered as a computationally cheaper alternative.

## II. GENERAL THEORETICAL FRAMEWORK

In this section the principles of adaptive Bayesian networks are discussed. The emphasis is on clarifying the underlying principles and exposing hidden approximations. The notation used in this paper can be found in the appendix. It is important to realise that there are two different types of PDF used: Bayesian (denoted as Q), and frequentist (denoted as P). Q is used to denote a Bayesian model PDF, whereas P is used to denote a frequency derived from a training set (used only where a full Bayesian analysis is impractical).

## III. MAKING PREDICTIONS FROM A MODEL

The fundamental problem is this: given a training set, and a model PDF, use Bayesian methods to make pre-

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.  
This appeared in Proceedings of the SPIE Conference on Adaptive and Learning Systems, Orlando, 140-151, 1992.

dictions about test data. Bayesian inference leaves no freedom of action once the problem is thus specified, and it yields the predictive model

$$\begin{aligned} Q(\mathbf{x}^+|\mathbf{x}^-) &= \int d\mathbf{s} Q(\mathbf{x}^+|\mathbf{s}) Q(\mathbf{s}|\mathbf{x}^-) && \text{exact} \\ Q(\mathbf{x}^+|\mathbf{x}^-) &\approx Q(\mathbf{x}^+|\mathbf{s}(\mathbf{x}^-)) && \text{approximate} \end{aligned} \quad (3.1)$$

The exact result is an application of a hidden variables model, in which the model parameters are integrated out thus

$$Q(\mathbf{x}^+, \mathbf{x}^-) = \int d\mathbf{s} Q(\mathbf{x}^+, \mathbf{x}^-, \mathbf{s}) \quad (3.2)$$

The quality of the exact result is limited only by the validity of the model. If there are several contending models, perhaps none of which is actually correct, then a single winner could be selected by, for instance, choosing the model that yielded the greatest probability of generating the data.

On the other hand, the approximate result avoids the computationally expensive integral over parameters. There is a variety of such methods, including

<sup>†</sup>Electronic address: luttrell%hermes.mod.uk@relay.mod.uk

1. maximum likelihood  $\mathbf{s}(\mathbf{x}^-) = \arg \max_{\mathbf{s}} Q(\mathbf{x}^-|\mathbf{s})$
2. maximum posterior probability  $\mathbf{s}(\mathbf{x}^-) = \arg \max_{\mathbf{s}} Q(\mathbf{s}|\mathbf{x}^-)$
3. maximum discriminatory likelihood  $\mathbf{s}(\mathbf{x}^-) = \arg \max_{\mathbf{s}} Q(\mathbf{x}_{\text{out}}^-|\mathbf{x}_{\text{in}}^-, \mathbf{s})$

The choice of approximation scheme can be used to

influence the capabilities of the predictive model. If the

posterior probability over parameters is highly localised, then schemes 1 and 2 are good approximations. On the other hand, the data might be partitioned into separate input and output spaces, and the model might need to be optimised for computing the conditional probability of the output given the input (i.e. to discriminate between alternative outputs). This would be a job for scheme 3.

In all schemes, whether exact or approximate, there is the problem of missing data. The exact method is unaffected by this, because missing data can be treated as hidden variables, and eliminated by integration. Whether or not an approximate method survives the problem of missing data depends on its structure. For instance, if the output data were omitted, then scheme 3 would be meaningless. There are many ways in which an approximate method could fail due to violations of its underlying assumptions. The moral is to construct the approximate method to suit the data, rather than the other way

around.

#### IV. VISIBLE AND HIDDEN VARIABLES

The data space may be augmented by appending an additional unobserved vector to the original data vector. In this case, the data vector comprises “visible variables”, whereas the unobserved vector comprises “hidden variables”. The model PDF then becomes

$$Q(\mathbf{x}|\mathbf{s}) = \int d\mathbf{h} Q(\mathbf{x}, \mathbf{h}|\mathbf{s}) = \int d\mathbf{h} Q(\mathbf{x}|\mathbf{h}, \mathbf{s}) Q(\mathbf{h}|\mathbf{s}) \quad (4.1)$$

Hidden variables can always be appended to a model, at the cost of additional theoretical and computational effort.

The prediction equation then becomes

---


$$\begin{aligned} Q(\mathbf{x}^+|\mathbf{x}^-) &= \int d\mathbf{s} d\mathbf{h}^+ d\mathbf{h}^- Q(\mathbf{x}^+, \mathbf{h}^+|\mathbf{s}) Q(\mathbf{s}, \mathbf{h}^-|\mathbf{x}^-) && \text{exact} \\ Q(\mathbf{x}^+|\mathbf{x}^-) &\approx \int d\mathbf{h}^+ Q(\mathbf{x}^+, \mathbf{h}^+|\mathbf{s}(\mathbf{x}^-)) && \text{approximate} \end{aligned} \quad (4.2)$$


---

where the notation used is self-explanatory.

#### V. LOGARITHMIC LIKELIHOOD AND RELATIVE ENTROPY

If  $N$  data samples are drawn independently from the training set, then the logarithmic likelihood may be written as

$$L(\mathbf{s}) \equiv \log \left( \prod_{k=1}^N Q(\mathbf{x}^k|\mathbf{s}) \right) \approx N \int d\mathbf{x} P(\mathbf{x}) \log(Q(\mathbf{x}|\mathbf{s})) \quad (5.1)$$

The integration measure  $P$  is a frequentist (i.e. non-Bayesian) PDF; it is used as a convenient shorthand notation, and will not enter into any Bayesian manipulations.

On the other hand, the relative entropy (between  $P$  and  $Q$ ) is defined as

$$G(\mathbf{s}) \equiv - \int d\mathbf{x} P(\mathbf{x}) \log \left( \frac{P(\mathbf{x})}{Q(\mathbf{x}|\mathbf{s})} \right) \leq 0 \quad (5.2)$$

Relative entropy is zero if, and only if,  $P$  and  $Q$  are identical, so it may be used as a fitness criterion for comparing  $P$  and  $Q$ . Relative entropy is the logarithm of the probability (per sample) that samples taken from  $Q$  have the frequency of occurrence specified by  $P$ .

These results can be combined to yield

$$L(\mathbf{s}) \approx N G(\mathbf{s}) + \int d\mathbf{x} P(\mathbf{x}) \log(P(\mathbf{x})) \quad (5.3)$$

The second term on the right hand side does not depend on the model parameters, so relative entropy is (up to an additive constant) approximately equal to logarithmic likelihood. In the limit of a large number of training samples, relative entropy maximisation is equivalent to maximum likelihood maximisation (i.e. it corresponds to approximate scheme 1).

#### VI. SOME ADAPTIVE NETWORKS REVISITED

In this section a brief review of some familiar adaptive Bayesian networks is presented.

#### VII. MIXTURE DISTRIBUTION

A mixture distribution is a hidden variables model with a PDF of the form

$$Q(\mathbf{x}|\mathbf{s}) = \sum_{c=1}^M Q(\mathbf{x}|c, \mathbf{s}) Q(c|\mathbf{s}) \quad (7.1)$$

where  $c$  is a “class label” which is a discrete-valued hidden variable. If the dependence on parameters is modified as follows

$$\begin{aligned} Q(\mathbf{x}|c, \mathbf{s}) Q(c|\mathbf{s}) &\longrightarrow Q(\mathbf{x}|c, \mathbf{s}) Q(c) \\ Q(\mathbf{x}|\mathbf{s}) &\longrightarrow Q(\mathbf{x}|\mathbf{s}, Q(c)) \end{aligned} \quad (7.2)$$

where the prior probability factor itself comprises  $M - 1$  of the parameters (i.e. it is non-parametric), then the iterative re-estimation prescription [2, 4] for maximising relative entropy becomes

$$\begin{aligned}\hat{\mathbf{s}} &= \arg \max_{\mathbf{s}'} \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^M Q(c|\mathbf{x}, \mathbf{s}) \log(Q(\mathbf{x}|c, \mathbf{s}')) \\ \hat{Q}(c) &= \int d\mathbf{x} P(\mathbf{x}) Q(c|\mathbf{x}, \mathbf{s})\end{aligned}\quad (7.3)$$

The structure of a mixture distribution network is shown in Figure 1.

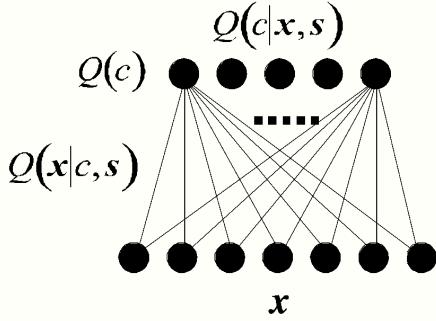


Figure 1: Diagram showing a network suitable for computing a mixture distribution. The input layer receives the data vector, and the output layer produces the corresponding class conditional probabilities. The network parameters are located as indicated.

### VIII. HIDDEN MARKOV MODEL

A hidden Markov model is a mixture distribution whose hidden variables have a memory of their previous state in time. Thus the mixture distribution PDF is modified to become

$$Q(\mathbf{x}^t|\mathbf{s}, Q(c^t|c^{t-1})) = \sum_{c^t=1}^M Q(\mathbf{x}^t|c^t, \mathbf{s}) Q(c^t|c^{t-1}) \quad (8.1)$$

where a discrete time index  $t$  now appears, and the prior probability is replaced by a transition matrix. The full joint PDF for a time series of data is then given by

$$Q(\mathbf{x}|\mathbf{s}, Q(c^t|c^{t-1})) = Q(c^0) \prod_{t=1}^T \left( \sum_{c^t=1}^M Q(\mathbf{x}^t|c^t, \mathbf{s}) Q(c^t|c^{t-1}) \right) \quad (8.2)$$

where a prior probability which specifies the initial distribution of the hidden variable  $c$  is included. The re-estimation prescription may be used to optimise the model parameters.

The structure of a hidden Markov model network is shown in Figure 2.

### IX. BOLTZMANN MACHINE

Gibbs distributions [6] (or, equivalently, Markov random fields) are a maximum entropy family of model PDF's with the form

$$\begin{aligned}Q(\mathbf{x}|\mathbf{s}) &= \frac{1}{Z(\mathbf{s})} \exp(-\mathbf{s} \cdot \mathbf{U}(\mathbf{x})) \\ \text{where } Z(\mathbf{s}) &= \int d\mathbf{x} \exp(-\mathbf{s} \cdot \mathbf{U}(\mathbf{x}))\end{aligned}\quad (9.1)$$

which depend exponentially on a sum of “potentials”. Hidden variables may readily be introduced into such models.

Because Gibbs distributions do not generally have the simple structure of mixture distributions or hidden Markov models, it is not usually possible to use the re-estimation prescription to optimise them. A relative entropy gradient ascent algorithm could be used instead, based on the result [9]

$$\frac{\partial G(\mathbf{s})}{\partial \mathbf{s}} = \int d\mathbf{x} d\mathbf{h} Q(\mathbf{x}, \mathbf{h}|\mathbf{s}) \mathbf{U}(\mathbf{x}, \mathbf{h}) - \int d\mathbf{x} d\mathbf{h} P(\mathbf{x}) Q(\mathbf{h}|\mathbf{x}, \mathbf{s}) \mathbf{U}(\mathbf{x}, \mathbf{h}) \quad (9.2)$$

This leads to a generalised form of the Boltzmann machine training algorithm [1], which optimises the model in a way that depends on the difference between the “free” average and the “clamped” average of the potentials.

### X. ADAPTIVE CLUSTER EXPANSION (ACE)

In this section an up-to-date discussion of the Adaptive Cluster Expansion (ACE) method [9] is presented, followed by a detailed theoretical analysis of a simple ACE network based on mixture distributions. A good example of the application of this approach is the anomaly detec-

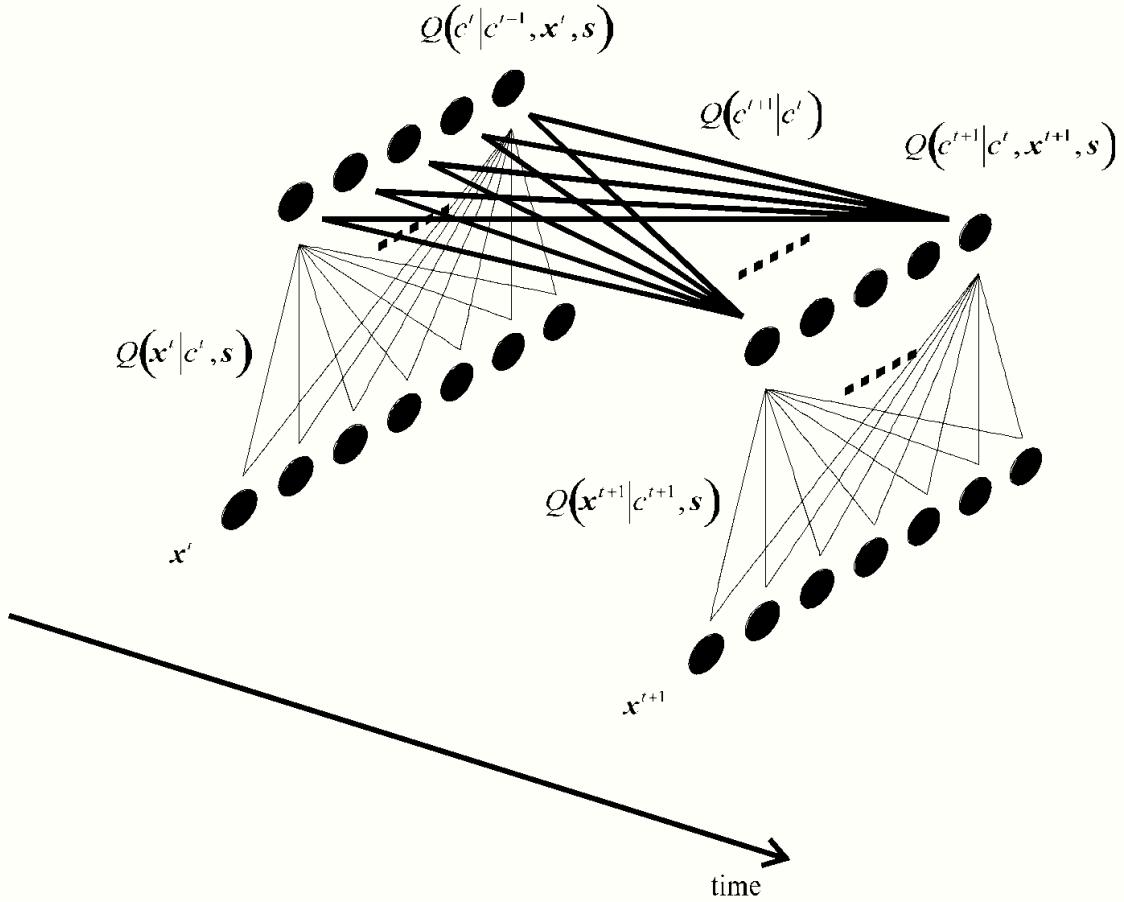


Figure 2: Diagram showing a network suitable for computing a hidden Markov model. As in the mixture distribution network, the input layer receives the data vector, and the output layer produces the corresponding class conditional probabilities. The network parameters are located as indicated. The output layer is influenced by its own previous output values, which endows the network with discrete time dynamics.

tor [10, 11]. Other uses (i.e. not PDF models) of the ACE approach have also been published [7, 8].

## XI. ACE PHILOSOPHY

One of the major problems with Bayesian networks with hidden variables is their tendency to consume large amounts of computing resources when evaluating the integration over hidden variables (including model parameters) in the exact prediction equation

$$Q(\mathbf{x}^+ | \mathbf{x}^-) = \int d\mathbf{s} d\mathbf{h}^+ d\mathbf{h}^- Q(\mathbf{x}^+, \mathbf{h}^+ | \mathbf{s}) Q(\mathbf{s}, \mathbf{h}^- | \mathbf{x}^-) \quad (11.1)$$

These numerical problems can be especially acute if Monte Carlo simulations are used.

The ACE approach factorises this equation into the following form

$$Q(\mathbf{x}^+ | \mathbf{x}^-) = Q_1(\mathbf{x}_1^+ | \mathbf{x}_1^-) Q_2(\mathbf{x}_2^+ | \mathbf{x}_2^-) J(\mathbf{y}_1(\mathbf{x}_1^+), \mathbf{y}_2(\mathbf{x}_2^+) | \mathbf{x}_1^-, \mathbf{x}_2^-) \quad (11.2)$$

where the first two factors model the PDF in two independent subspaces (or clusters), and the third factor models the joint PDF of transformed versions of the clusters. In the J-factor, the choice of cluster transformations

is arbitrary, though the choice affects the performance of the prediction equation. To ensure that the overall normalisation is correct, it is necessary to write the J-factor

in the form

$$J(\mathbf{y}_1, \mathbf{y}_2 | \mathbf{x}_1^-, \mathbf{x}_2^-) = \frac{Q_{12}(\mathbf{y}_1^+, \mathbf{y}_2^+ | \mathbf{y}_1^-, \mathbf{y}_2^-)}{Q_1(\mathbf{y}_1^+ | \mathbf{x}_1^-) Q_2(\mathbf{y}_2^+ | \mathbf{x}_2^-)} \quad (11.3)$$

where the numerator models the PDF of the transformed clusters, and the denominator is the product of the transformed PDF's of the original clusters, defined by

$$\begin{aligned} Q_1(\mathbf{y}_1^+ | \mathbf{x}_1^-) &= \int d\mathbf{x}_1^+ Q_1(\mathbf{x}_1^+ | \mathbf{x}_1^-) \delta(\mathbf{y}_1^+ - \mathbf{y}_1(\mathbf{x}_1^+)) \\ Q_2(\mathbf{y}_2^+ | \mathbf{x}_2^-) &= \text{similar} \end{aligned} \quad (11.4)$$

The above factorisation cannot be derived from the original prediction equation; it has to be assumed, and justified as follows. Suppose there are four Bayesians, each of whom has partial access to the training data. Bayesian 1 has access to cluster 1, Bayesian 2 has ac-

cess to cluster 2, and Bayesian 3 has access to the transformed cluster pairs. Bayesians 1-3 can each independently form an exact prediction equation for his own (sub)space. Bayesian 4 can then use these three results (and no further information) to form a uniquely consistent combination, which is the factorised prediction equation stated earlier.

Finally, each of the factors can be modelled by any of the methods discussed thus far: exact or approximate, and with or without additional hidden variables.

## XII. APPROXIMATE ACE: REMOVE THE INTEGRALS OVER PARAMETERS

The above ACE expression for two cluster may be approximated as

$$Q(\mathbf{x}^+ | \mathbf{x}^-) \approx Q_1(\mathbf{x}_1^+ | \mathbf{s}_1(\mathbf{x}_1^-, \mathbf{x}_2^-)) Q_2(\mathbf{x}_2^+ | \mathbf{s}_2(\mathbf{x}_1^-, \mathbf{x}_2^-)) \frac{Q_{12}(\mathbf{y}_1(\mathbf{x}_1^+), \mathbf{y}_2(\mathbf{x}_2^+) | \mathbf{s}_{12}(\mathbf{x}_1^-, \mathbf{x}_2^-))}{Q_1(\mathbf{y}_1(\mathbf{x}_1^+) | \mathbf{s}_1(\mathbf{x}_1^-, \mathbf{x}_2^-)) Q_2(\mathbf{y}_2(\mathbf{x}_2^+) | \mathbf{s}_2(\mathbf{x}_1^-, \mathbf{x}_2^-))} \quad (12.1)$$

where the parameter values have been set to their maximum likelihood values (i.e. approximate scheme 1)

$$\begin{pmatrix} \mathbf{s}_1(\mathbf{x}_1^-, \mathbf{x}_2^-) \\ \mathbf{s}_2(\mathbf{x}_1^-, \mathbf{x}_2^-) \\ \mathbf{s}_{12}(\mathbf{x}_1^-, \mathbf{x}_2^-) \end{pmatrix} = \arg \max_{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12}} Q_1(\mathbf{x}_1^- | \mathbf{s}_1) Q_2(\mathbf{x}_2^- | \mathbf{s}_2) \frac{Q_{12}(\mathbf{y}_1(\mathbf{x}_1^-), \mathbf{y}_2(\mathbf{x}_2^-) | \mathbf{s}_{12})}{Q_1(\mathbf{y}_1(\mathbf{x}_1^-) | \mathbf{s}_1) Q_2(\mathbf{y}_2(\mathbf{x}_2^-) | \mathbf{s}_2)} \quad (12.2)$$

Note that the parameters need to be jointly optimised because the term arising from the J-factor is not independent of the first two factors.

When the amount of training data is large, this result is equivalent to relative entropy maximisation. Thus define

$$\begin{aligned} G_1(\mathbf{s}_1) &= - \int d\mathbf{x}_1 P(\mathbf{x}_1) \log \left( \frac{P(\mathbf{x}_1)}{Q_1(\mathbf{x}_1 | \mathbf{s}_1)} \right) \\ G_2(\mathbf{s}_2) &= \text{similar} \\ I_{12}(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12}) &= \int d\mathbf{y}_1 d\mathbf{y}_2 P(\mathbf{y}_1, \mathbf{y}_2 | \mathbf{s}_1, \mathbf{s}_2) \log \left( \frac{Q_{12}(\mathbf{y}_1, \mathbf{y}_2 | \mathbf{s}_{12})}{Q_1(\mathbf{y}_1 | \mathbf{s}_1) Q_2(\mathbf{y}_2 | \mathbf{s}_2)} \right) \\ I_0 &= \int d\mathbf{x}_1 d\mathbf{x}_2 P(\mathbf{x}_1, \mathbf{x}_2) \log \left( \frac{P_{12}(\mathbf{x}_1, \mathbf{x}_2)}{P_1(\mathbf{x}_1) P_2(\mathbf{x}_2)} \right) \end{aligned} \quad (12.3)$$

to obtain [12].

## XIII. MIXTURE DISTRIBUTION ACE

$$G(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12}) = G_1(\mathbf{s}_1) + G_2(\mathbf{s}_2) + I_{12}(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12}) - I_0 \quad (12.4)$$

The first two terms are the relative entropies evaluated independently for each cluster, the third term is (a model of) the mutual information between the transformed clusters, and the fourth term is the (constant) mutual information between the clusters themselves.

Although the integrals over parameters were removed above, it is still possible to introduce additional hidden variables. Thus model each PDF in the ACE expression

as a mixture distribution

$$\begin{aligned} Q_1(\mathbf{x}_1|\mathbf{s}_1) &= \sum_{c=1}^M Q_1(\mathbf{x}_1|c, \mathbf{s}_1) Q_1(c|\mathbf{s}_1) \\ Q_2(\mathbf{x}_2|\mathbf{s}_2) &= \text{similar} \\ Q_{12}(\mathbf{y}_1, \mathbf{y}_2|\mathbf{s}_{12}) &= \sum_{c=1}^M Q_{12}(\mathbf{y}_1, \mathbf{y}_2|c, \mathbf{s}_{12}) Q_{12}(c|\mathbf{s}_{12}) \end{aligned} \quad (13.1)$$


---

Furthermore, it is possible to define the cluster transformations using quantities that are computed by the mixture distribution model. For instance, one possible choice is the vector of posterior class probabilities. Omitting the cluster suffices, this may be written as

$$\mathbf{y}(\mathbf{x}|\mathbf{s}) = \frac{(Q(\mathbf{x}|c=1, \mathbf{s}), Q(c=1|\mathbf{s}), Q(\mathbf{x}|c=2, \mathbf{s}), Q(c=2|\mathbf{s}), \dots)}{\sum_{c=1}^M Q(\mathbf{x}|c, \mathbf{s}) Q(c|\mathbf{s})} \quad (13.2)$$


---

This choice is non-Bayesian, because a probability is being used as if it were a state vector, which is not permitted for Bayesian probabilities. However, because it is being used only as a cluster transformation, and not in any further Bayesian manipulations, this procedure is allowed.

The structure of an ACE network suitable for computing the above results is shown in Figure 3.

In Appendix B, expressions for the derivatives of the relative entropy are presented, together with a brief discussion on self-supervision.

#### XIV. GENERAL ACE MODEL

By including a J-factor for each cluster transformation, the above ACE network can readily be generalised to a tree-structured network with any number of layers. The expression for the relative entropy is similarly modified to incorporate the additional J-factors.

The gradients of the relative entropy may be derived as follows. Introduce some simplified notation

$$\begin{aligned} \mathbf{x}_L &= \text{input to layer } L \\ \mathbf{y}_L &= \text{output from layer } L \\ &= \mathbf{x}_{L+1} \\ &= \text{input to layer } L+1 \\ \mathbf{y}_L &= \mathbf{y}_L(\mathbf{x}_L) \\ &= \text{cluster transformations in layer } L \\ G &= \int d\mathbf{x} P(\mathbf{x}) G(\mathbf{x}) \\ &= \text{decomposition by input sample} \\ G(\mathbf{x}) &= \sum_{L=1}^N J_L(\mathbf{x}_L) \\ &= \text{decomposition by network layer} \end{aligned} \quad (14.1)$$


---

The relative entropy may be differentiated with respect to the parameters in network layer L to yield

$$\frac{\partial G(\mathbf{x})}{\partial \mathbf{s}_L} = \frac{\partial J_L(\mathbf{x}_L)}{\partial \mathbf{s}_L} + \frac{\partial \mathbf{y}_L}{\partial \mathbf{s}_L} \cdot \mathbf{b}_{L+1} \quad (14.2)$$

where the back-propagated signal from layer L + 1 is obtained by iterating the back-propagation equations

$$\begin{aligned} \mathbf{b}_r &= \frac{\partial J_r(\mathbf{x}_r)}{\partial \mathbf{s}_r} + \frac{\partial \mathbf{y}_r}{\partial \mathbf{s}_r} \cdot \mathbf{b}_{r+1} \quad r = N, N-1, \dots, 3, 2, 1 \\ \mathbf{b}_{N+1} &= \mathbf{0} \end{aligned} \quad (14.3)$$

The flow of information needed to compute the above results is shown in Figure 4.

It is also possible to replace the mixture distributions used above, by hidden Markov models. This is a natural way of introducing temporal behaviour into ACE.

#### XV. CONCLUSIONS

The theory of adaptive Bayesian networks is very powerful, and contains many familiar methods as special cases. The main problem with the Bayesian method, when applied rigorously, is the need to perform integrations over hidden variables, which can involve lengthy Monte Carlo simulations in some cases. The Adaptive Cluster Expansion (ACE) method reviewed in this paper provides one possible escape route from the need for such integrations. ACE provides a scalable network architecture that can readily be applied to large datasets, such as images.

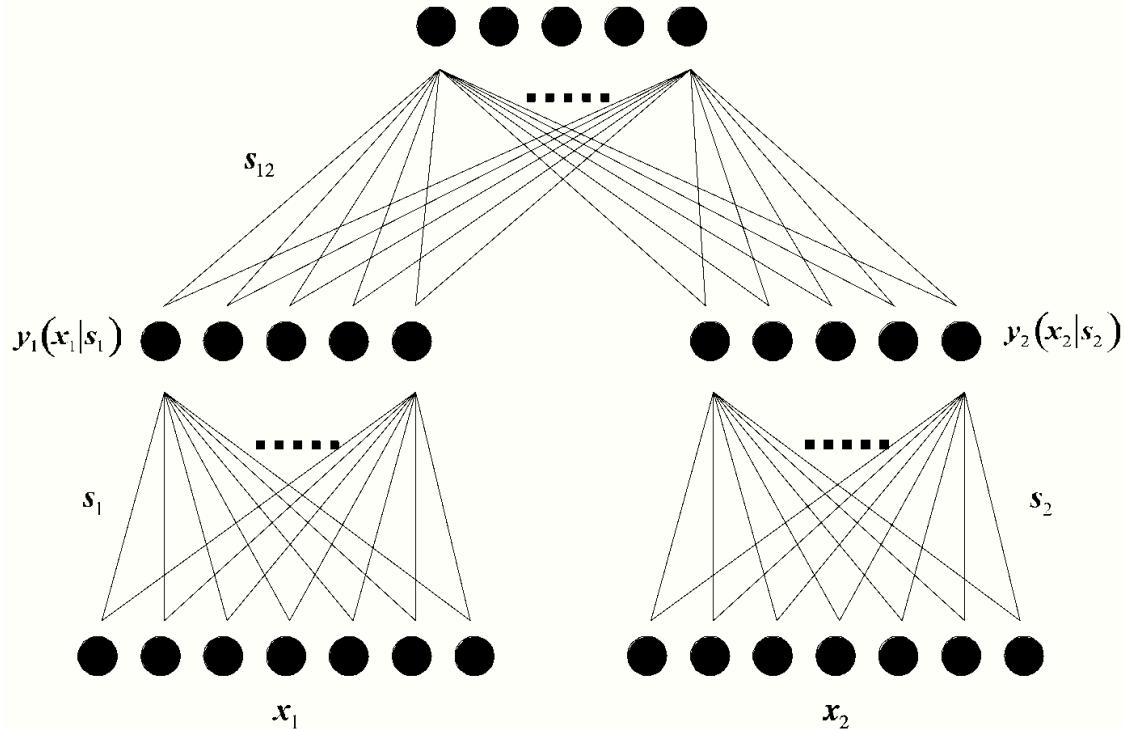


Figure 3: Diagram showing an ACE network. The second stage receives as input two vectors of posterior class probabilities computed in the first stage.

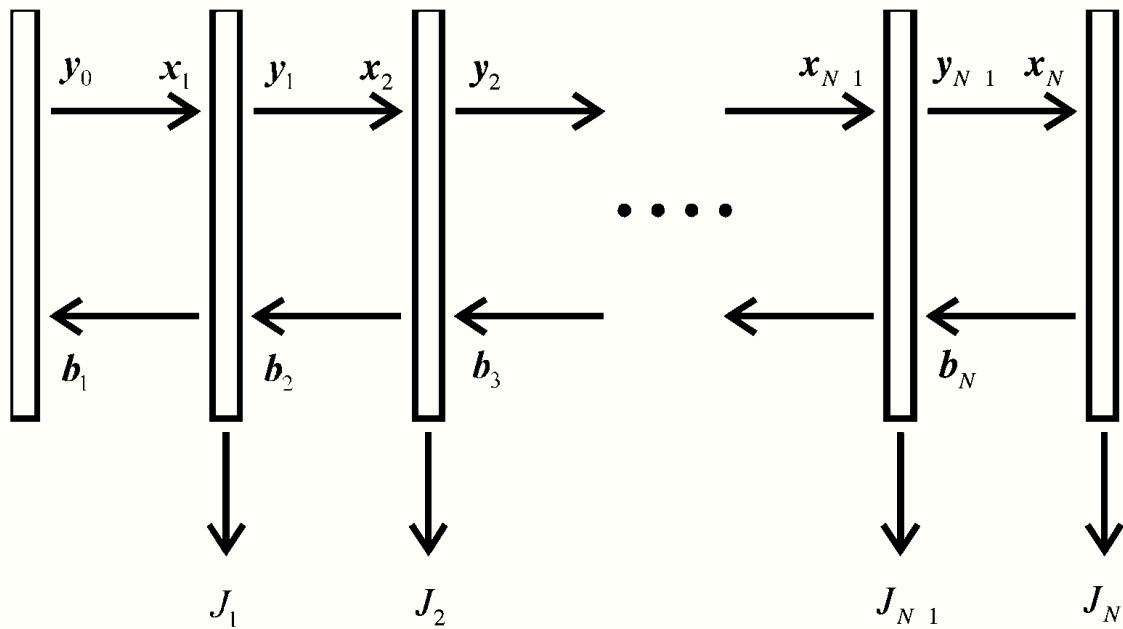


Figure 4: Diagram showing a schematic multilayer ACE network. The flow from left-to-right is the forward-propagating data, and the flow from right-to-left is the backward-propagating derivatives generated within the network. The contributions to the relative entropy and its derivatives are output from each network layer, as shown.

## Appendix A: Basic notation

$\mathbf{x}^k$	= k-th vector in data space
$\mathbf{x}^-$	= $(\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^N)$ = past data (training data)
$\mathbf{x}^+$	= $(\mathbf{x}^{N+1}, \mathbf{x}^{N+2}, \dots)$ = future data (test data)
$\mathbf{y}(\mathbf{x})$	= transformation of data vector
$\mathbf{y}(\mathbf{x}^\pm)$	= $\mathbf{y}^\pm$ = transformation of all future/past data
$\mathbf{s}, \mathbf{h}$	= hidden variables vector
$P(\mathbf{x})$	= PDF of data (frequentist "real world")
$Q(\mathbf{x} \mathbf{s})$	= conditional PDF of data given a parameter vector (Bayesian model)
$Q(\mathbf{x}^+ \mathbf{x}^-)$	= conditional PDF of future data given past data (Bayesian model)
$L(\mathbf{s})$	= logarithmic likelihood
$G(\mathbf{s})$	= relative entropy

## Appendix B: Gradient of relative entropy in mixture distribution ACE

The gradients of the relative entropy are

$$\begin{aligned}
 \frac{\partial G_1(\mathbf{x}_1)}{\partial \mathbf{s}_1} &= \int d\mathbf{x}_1 P(\mathbf{x}_1) \frac{\partial \log Q_1(x_1|\mathbf{s}_1)}{\partial \mathbf{s}_1} \\
 \frac{\partial G_2(\mathbf{x}_2)}{\partial \mathbf{s}_2} &= \text{similar} \\
 \frac{\partial I_{12}(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12})}{\partial \mathbf{s}_1} &= \int d\mathbf{x}_1 d\mathbf{x}_2 P(\mathbf{x}_1, \mathbf{x}_2) \left. \frac{\partial \mathbf{y}_1}{\partial \mathbf{s}_1} \cdot \frac{\partial \log Q_{12}(\mathbf{y}_1, \mathbf{y}_2|\mathbf{s}_{12})}{\partial \mathbf{y}_1} \right|_{\substack{\mathbf{y}_1=\mathbf{y}_1(\mathbf{x}_1|\mathbf{s}_1) \\ \mathbf{y}_2=\mathbf{y}_2(\mathbf{x}_2|\mathbf{s}_2)}} - \int d\mathbf{y}_1 P(\mathbf{y}_1|\mathbf{s}_1) \frac{\partial \log Q_1(\mathbf{y}_1|\mathbf{s}_1)}{\partial \mathbf{s}_1} \\
 \frac{\partial I_{12}(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12})}{\partial \mathbf{s}_2} &= \text{similar} \\
 \frac{\partial I_{12}(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12})}{\partial \mathbf{s}_{12}} &= \int d\mathbf{y}_1 d\mathbf{y}_2 P(\mathbf{y}_1, \mathbf{y}_2|\mathbf{s}_1, \mathbf{s}_2) \frac{\partial \log Q_{12}(\mathbf{y}_1, \mathbf{y}_2|\mathbf{s}_{12})}{\partial \mathbf{s}_{12}}
 \end{aligned} \tag{B1}$$

The derivative of (the model of) the mutual information with respect to the parameters in the first stage of the network causes the training of the left and right cluster transformations to become mutually coupled [13, 14].

This effect, which is called self-supervision, is important because it co-ordinates the training of different layers in a multilayer unsupervised network (of which the ACE network is but a particular example).

- [1] D H Ackley, G E Hinton, and T J Sejnowski, *A learning algorithm for Boltzmann machines*, Cognitive Science **9** (1985), no. 2, 147–169.
- [2] L E Baum, *An inequality and associated maximisation technique in statistical estimation for probabilistic functions of Markov processes*, Inequalities **3** (1972), no. 1, 1–8.
- [3] R T Cox, *Probability, frequency and reasonable expectation*, American Journal of Physics **14** (1946), no. 1, 1–13.
- [4] A P Dempster, N M Laird, and D B Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society B **39** (1977), no. 1, 1–38.
- [5] H Jeffreys, *Theory of probability*, Clarendon Press, Oxford, 1939.
- [6] R Kindermann and J L Snell, *Markov random fields and their applications*, Contemporary Mathematics, vol. 1, American Mathematical Society, Providence, 1980.
- [7] S P Luttrell, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
- [8] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
- [9] ———, *Maximum entropy and Bayesian methods*, ch. The use of Bayesian and entropic methods in neural network theory, pp. 363–370, Kluwer, Dordrecht, 1989.
- [10] ———, *A trainable texture anomaly detector using the adaptive cluster expansion (ACE) method*, Memorandum 4437, Royal Signals and Radar Establishment, Malvern, 1990.
- [11] ———, *Adaptive Cluster Expansion (ACE): a multi-layer network for estimating probability density functions*,

- (1991).
- [12] ———, *A hierarchical network for clutter and texture modelling*, Proceedings of SPIE conference on adaptive signal processing (San Diego), SPIE, 1991, pp. 518–528.
- [13] ———, *Self-supervised training of hierarchical vector quantisers*, Proceedings of IEE conference on artificial neural networks (Bournemouth), IEE, 1991, pp. 5–9.
- [14] ———, *Self-supervised adaptive networks*, Proceedings of the IEE F **139** (1992), no. 6, 371–377.



# Gibbs Distribution Theory of Adaptive n-Tuple Networks \*

S. P. Luttrell<sup>†</sup>

*Image Processing Research Section, Defence Research Agency,  
Royal Signals and Radar Establishment, St Andrews Road,  
Malvern, Worcestershire, WR14 3PS, United Kingdom*

In this paper it is demonstrated that the theory of optimising Gibbs distributions leads to training schemes for n-tuple networks. Both unsupervised and supervised networks emerge naturally from this analysis.

## I. GIBBS DISTRIBUTION OPTIMISATION

In this section unsupervised and supervised criteria for optimising Gibbs distributions are reviewed.

### A. Maximum relative entropy criterion

A Gibbs distribution is defined in terms of a set of potentials and Lagrange multipliers as

$$Q(\mathbf{x}|\mathbf{s}) \equiv \frac{1}{Z} \exp(-\mathbf{s} \cdot \mathbf{U}(\mathbf{x}))$$

where  $Z \equiv \int d\mathbf{x} \exp(-\mathbf{s} \cdot \mathbf{U}(\mathbf{x}))$  (1.1)

The relative entropy of a “real world” PDF with respect to this model is defined as

$$G(\mathbf{s}) \equiv - \int d\mathbf{x} P(\mathbf{x}) \log \left( \frac{P(\mathbf{x})}{Q(\mathbf{x}|\mathbf{s})} \right) \quad (1.2)$$

which has the following gradient with respect to its parameters

$$\frac{\partial G(\mathbf{s})}{\partial \mathbf{s}} = \int d\mathbf{x} P(\mathbf{x}) \left( \int d\mathbf{y} Q(\mathbf{y}|\mathbf{s}) \mathbf{U}(\mathbf{y}) - \mathbf{U}(\mathbf{x}) \right) \quad (1.3)$$

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This paper appeared in Proceedings of International Conference on Artificial Neural Networks, 1992, vol. 1, pp. 313-316. © British Crown Copyright 1992 /MOD. Published with the permission of

Note that Equation 1.3 has been written in a way that is best for deducing a gradient ascent training algorithm. The Gibbs distribution is optimised when this gradient is zero.

### B. Maximum discriminatory relative entropy criterion

If discrimination between a number of Gibbs distributions, each identified by its own class label  $c$ , is required, then Bayes’ theorem must first be used to compute the posterior probability over classes, to yield

$$Q(c|\mathbf{x}, \mathbf{s}_c) = \frac{Q(\mathbf{x}|\mathbf{s}_c) P(c)}{\sum_{c'=1}^N Q(\mathbf{x}|\mathbf{s}_{c'}) P(c')} \quad (1.4)$$

where the prior class probabilities are assumed to be known. The average (over data) of the relative entropy of the posterior class probabilities is defined as

$$G(\mathbf{s}_1, \dots, \mathbf{s}_N) \equiv - \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N P(c|\mathbf{x}) \log \left( \frac{P(c|\mathbf{x})}{Q(c|\mathbf{x}, s_c)} \right) \quad (1.5)$$

which yields the gradient

the Controller of Her Britannic Majesty’s Stationery Office.

†Electronic address: luttrell%hermes.mod.uk@relay.mod.uk

---


$$\frac{\partial G(\mathbf{s}_1, \dots, \mathbf{s}_N)}{\partial \mathbf{s}_{c'}} = \sum_{c=1}^N \int d\mathbf{x} P(\mathbf{x}, c) \left( \int d\mathbf{y} Q(\mathbf{y}|s_{c'}) \mathbf{U}(\mathbf{y}) - \mathbf{U}(\mathbf{x}) \right) (\delta_{c,c'} - Q(c'|\mathbf{x})) \quad (1.6)$$

## II. CHOICE OF POTENTIALS

In this section the above results are specialised to the case of localised potentials, and the corresponding optimisation equations are derived.

### A. Localised potentials

If the potentials are highly localised functions of position, then they may be modelled as Dirac delta functions [2]. This type of model requires the following notation to be used

$$\begin{aligned} \mathbf{U}(\mathbf{x}) &\longrightarrow \delta(\mathbf{x} - \mathbf{x}') \\ \mathbf{s} &\longrightarrow \mathbf{s}(\mathbf{x}') \\ \mathbf{s} \cdot \mathbf{U}(\mathbf{x}) &\longrightarrow \int d\mathbf{x}' \mathbf{s}(\mathbf{x}') \delta(\mathbf{x} - \mathbf{x}') \\ &= \mathbf{s}(\mathbf{x}) \end{aligned} \quad (2.1)$$

$$\begin{aligned} Q(\mathbf{x}|\mathbf{s}) &\longrightarrow \frac{1}{Z} \exp(-\mathbf{s}(\mathbf{x})) \\ \text{where } Z &\equiv \int d\mathbf{x} \exp(-\mathbf{s}(\mathbf{x})) \end{aligned} \quad (2.2)$$

Note that the vector of Lagrange multipliers is replaced by a Lagrange function. The relative entropy gradients in Equation 1.3 and Equation 1.6 become

$$\begin{aligned} \frac{\delta G}{\delta \mathbf{s}(\mathbf{x}')} &= \int d\mathbf{x} P(\mathbf{x}) (Q(\mathbf{x}') - \delta(\mathbf{x} - \mathbf{x}')) \\ \frac{\delta G}{\delta \mathbf{s}_{c'}(\mathbf{x}')} &= \sum_{c=1}^N \int d\mathbf{x} P(\mathbf{x}, c) (Q(\mathbf{x}'|c') - \delta(\mathbf{x} - \mathbf{x}')) (\delta_{c,c'} - Q(c'|\mathbf{x})) \end{aligned} \quad (2.3)$$

which yield the following gradient ascent training schemes for unsupervised and supervised training, respectively

$$\delta \mathbf{s}(\mathbf{x}') \propto (Q(\mathbf{x}') - \delta(\mathbf{x} - \mathbf{x}')) \quad (2.4)$$

$$\delta \mathbf{s}_{c'}(\mathbf{x}') \propto (Q(\mathbf{x}'|c') - \delta(\mathbf{x} - \mathbf{x}')) (\delta_{c,c'} - Q(c'|\mathbf{x})) \quad (2.5)$$

In both cases, after presentation with a training example, the Lagrange function is updated for all  $\mathbf{x}'$  (unsupervised), and for all  $\mathbf{x}'$  and  $c'$  (supervised).

### B. Interpretation

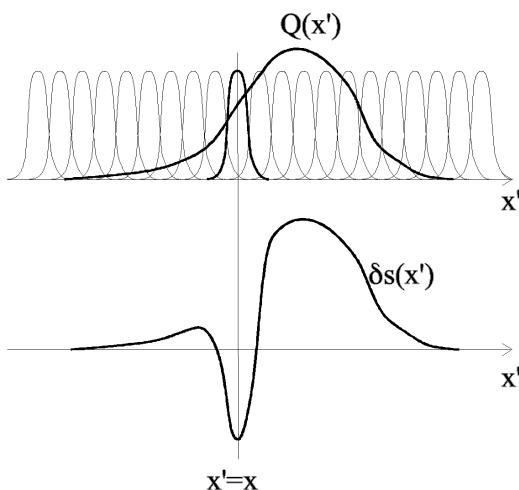


Figure 1: Diagrams representing a 1-dimensional version of Equation 2.4. In the upper diagram, an array of localised potentials and a Gibbs distribution are shown. The potential drawn in bold is the one concentrated about the current value of  $\mathbf{x}$  supplied by the training set. In the lower diagram, the corresponding update to the Lagrange function is shown. The overall effect is to increase the Gibbs distribution in the vicinity of  $\mathbf{x}$ , and to suppress it elsewhere.

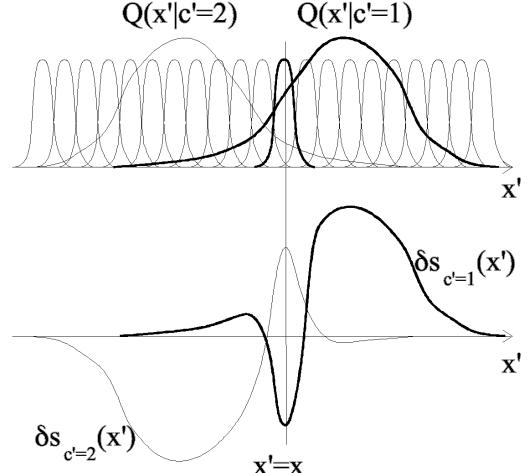


Figure 2: Diagrams representing a 1-dimensional version of Equation 2.5. In the upper diagram, there are two Gibbs distributions. The potential and the Gibbs distribution drawn in bold indicate the current values of  $\mathbf{x}$  and  $c$  supplied by the training set, respectively. In the lower diagram, the corresponding updates to the two Lagrange functions are shown. The overall effect is to increase the Gibbs distribution labelled  $c$  in the vicinity of  $\mathbf{x}$ , and to suppress it elsewhere. However, the other Gibbs distribution (i.e. not labelled  $c$ ) is affected the opposite sense.

In Figure 1 and Figure 2 diagrammatic interpretations of Equation 2.4 and Equation 2.5 are presented using finite width localised potentials. As long as the scale on which the Gibbs distribution PDFs vary is much greater than the width of the potentials, it does not matter much whether Dirac delta functions or finite width potentials are used.

It is important to realise that the update prescription is applied to the Lagrange functions; a localised increase in the value of a Lagrange function causes a corresponding localised decrease in the value of the corresponding Gibbs distribution (and also some global renormalisation). In the unsupervised case shown in Figure 1 this causes the Gibbs distribution to approximate the “real world” PDF. In the supervised case shown in Figure 2 the two Gibbs distributions optimise their combined ability to classify the data; this means that they need only produce a good

model of the decision boundary between the two classes. This effect also occurs when non-local potentials are used.

### C. N-tuple networks

N-tuple networks are a useful category of limited inter-connect networks, with the property that their internal computations each depend on only a small subset of the input data, so they can be implemented directly using table look-up techniques; this leads to some very efficient hardware implementations [1]. In order to make the connection between Gibbs distribution theory and n-tuple networks, it is necessary to generalise the notation in Equation 2.2 as follows

$$\begin{aligned}
 \mathbf{x} &\longrightarrow (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M) \\
 \mathbf{U}(\mathbf{x}) &\longrightarrow (\delta(\mathbf{x} - \mathbf{x}_1'), \delta(\mathbf{x} - \mathbf{x}_2'), \dots, \delta(\mathbf{x} - \mathbf{x}_M')) \\
 \mathbf{s} &\longrightarrow (\mathbf{s}_1(\mathbf{x}_1'), \mathbf{s}_2(\mathbf{x}_2'), \dots, \mathbf{s}_M(\mathbf{x}_M')) \\
 \mathbf{s} \cdot \mathbf{U}(\mathbf{x}) &\longrightarrow \left( \int d\mathbf{x}_1' \mathbf{s}_1(\mathbf{x}_1') \delta(\mathbf{x} - \mathbf{x}_1'), \dots \right) \\
 &= (\mathbf{s}_1(\mathbf{x}_1), \mathbf{s}_2(\mathbf{x}_2), \dots, \mathbf{s}_M(\mathbf{x}_M)) \\
 Q(\mathbf{x}|\mathbf{s}) &\longrightarrow \prod_{i=1}^M \frac{1}{Z_i} \exp(-\mathbf{s}_i(\mathbf{x}_i)) \\
 \text{where } Z_i &\equiv \int d\mathbf{x}_i \exp(-\mathbf{s}_i(\mathbf{x}_i))
 \end{aligned} \tag{2.6}$$

where the data has been partitioned into a vector of non-overlapping n-tuples, each of which has its own private set of localised potentials. The overall effect is to replace the single Gibbs distribution by a product of Gibbs distributions (one for each n-tuple).

G-maximisation then leads to the following update equations

$$\delta \mathbf{s}_i(\mathbf{x}_i') \propto (Q_i(\mathbf{x}_i') - \delta(\mathbf{x}_i - \mathbf{x}_i')) \tag{2.7}$$

$$\delta \mathbf{s}_{i,c'}(\mathbf{x}') \propto (Q_i(\mathbf{x}_i'|c_i') - \delta(\mathbf{x}_i - \mathbf{x}_i')) (\delta_{c,c'} - Q(c'|x)) \tag{2.8}$$

where the  $i$  index denotes which n-tuple has its Lagrange function(s) updated. Equation 2.7 and Equation 2.8 have the same interpretation as Equation 2.4 and Equation 2.5, respectively; they are the Gibbs distribution formulation (unsupervised and supervised, respectively) of an n-tuple network training scheme.

### III. CONCLUSIONS

Entropic criteria have been used to optimise Gibbs distribution models (both unsupervised and supervised). When the Gibbs potentials are localised, and the data vector is partitioned into n-tuples, this leads to a novel derivation of an n-tuple network training scheme. This theory is directly applicable to WISARD [1].

Maximum relative entropy optimisation of Gibbs distribution models may also be used to derive many other results that extend the basic n-tuple technique [3].

[1] I Aleksander and T J Stonham, *Guide to pattern recognition using random access memories*, Computers and Digital Techniques **2** (1979), no. 1, 29–40.  
317

ital Techniques **2** (1979), no. 1, 29–40.

- [2] S P Luttrell, *Maximum entropy and Bayesian methods*, ch. The use of Bayesian and entropic methods in neural network theory, pp. 363–370, Kluwer, Dordrecht, 1989.
- [3] ———, *Adaptive Bayesian networks*, Proceedings of SPIE conference on adaptive signal processing (Orlando), SPIE, 1992, pp. 140–151.

# An Adaptive Bayesian Network for Low-Level Image Processing \*

S P Luttrell<sup>†</sup>

Defence Research Agency, Malvern, Worcs, WR14 3PS, UK.

## I. INTRODUCTION

Probability calculus, based on the axioms of inference, Cox [1], is the only consistent scheme for performing inference; this is also known as Bayesian inference. The objects which this approach manipulates, namely probability density functions (PDFs), may be created in a variety of ways, but the focus of this paper is on the use of adaptive PDF networks. Adaptive mixture distribution (MD) networks are already widely used, Luttrell [4]. In this paper an extension of the standard MD approach is presented; it is called a partitioned mixture distribution (PMD). PMD networks are designed specifically to scale sensibly to high-dimensional problems, such as image processing. Several numerical simulations are performed which demonstrate that the emergent properties of PMD networks are similar to those of biological low-level vision processing systems.

## II. THEORY

In this section the use of PDFs as a vehicle for solving inference problems is discussed, and the standard theory of MDs is summarised. The new theory of PMDs is then presented.

### A. The Bayesian Approach

The axioms of inference, Cox [1], lead to the use of probabilities as the unique scheme for performing consistent inference. This approach has three essential stages: (i) choose a state space  $\mathbf{x}$ , (ii) assign a joint PDF  $Q(\mathbf{x})$ , and (iii) draw inferences by computing conditional PDFs. This is generally called the Bayesian approach. Usually,  $\mathbf{x}$  is split into two or more subspaces in order to distinguish between externally accessible components (e.g. data) and inaccessible components (e.g. model parameters).

To illustrate the use of the Bayesian approach, the joint PDF  $Q(\mathbf{x}^+, \mathbf{x}^-, \mathbf{s})$  (where  $\mathbf{x}^-$  is the past data,  $\mathbf{x}^+$  is the

future data, and  $\mathbf{s}$  are the model parameters) allows predictions to be made by computing the conditional PDF of the future given the past.

$$Q(\mathbf{x}^+|\mathbf{x}^-) = \int d\mathbf{s} Q(\mathbf{x}^+|\mathbf{s}) Q(\mathbf{s}|\mathbf{x}^-) \quad (2.1)$$

If  $Q(\mathbf{s}|\mathbf{x}^-)$  has a single sharp peak (as a function of  $\mathbf{s}$ ) then the integration can be approximated as follows.

$$Q(\mathbf{x}^+|\mathbf{x}^-) \approx Q(\mathbf{x}^+|\mathbf{s}(\mathbf{x}^-)) \quad (2.2)$$
$$\mathbf{s}(\mathbf{x}^-) = \begin{cases} \arg \max_{\mathbf{s}} Q(\mathbf{x}^-|\mathbf{s}) & \text{max. likelihood} \\ \arg \max_{\mathbf{s}} Q(\mathbf{s}|\mathbf{x}^-) & \text{max. posterior} \end{cases}$$

This approximation is accurate only in situations where the integration over model parameters is dominated by contributions in the vicinity of  $\mathbf{s} = \mathbf{s}(\mathbf{x}^-)$ .

The above approximations can be interpreted in “neural network” language as follows. The parameters  $\mathbf{s} = \mathbf{s}(\mathbf{x}^-)$  correspond to the neural network weights (or whatever) that emerge from the training process. Subsequent computation of  $Q(\mathbf{x}|\mathbf{s})$  corresponds to testing the neural network in the field. This analogy holds for both supervised and unsupervised networks, although the details of the training algorithm and the correspondence between  $(\mathbf{x}, \mathbf{s})$  and (network inputs/outputs, network weights) is different in each case. This correspondence between PDF models and neural networks will be tacitly assumed throughout this paper.

### B. The Mixture Distribution

An MD can be written as (omitting  $\mathbf{s}$ , for clarity)

$$Q(\mathbf{x}) = \sum_{c=1}^n Q(\mathbf{x}, c) = \sum_{c=1}^n Q(\mathbf{x}|c) Q(c) \quad (2.3)$$

An additional variable  $c$  is introduced, and the required PDF  $Q(\mathbf{x})$  is recovered by averaging the joint PDF  $Q(\mathbf{x}, c)$  over the possible states of  $c$ .  $c$  can be interpreted as a class label, in which case the  $Q(\mathbf{x}|c)$  are class likelihoods, and the  $Q(c)$  are class prior probabilities.

In Figure 1 a MD is drawn as if it were a neural network. It is a type of “radial basis function” network in which the input layer is transformed through a set of non-linear functions  $Q(\mathbf{x}|c)$  (which do not necessarily have a purely radial dependence), and then transformed through a linear summation operation.

\*Typeset in LATEX on May 21, 2019.

This appeared in Proceedings of the 3<sup>rd</sup> International Conference on Artificial Neural Networks, Brighton, 61-65, 1993. © British Crown Copyright 1993/DRA. Published with the permission of the Controller of Her Britannic Majesty’s Stationery Office

<sup>†</sup>Electronic address: luttrell%uk.mod.hermes@relay.mod.uk

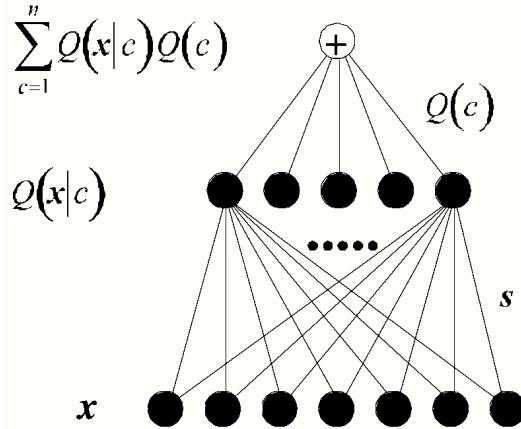


Figure 1: Mixture distribution network.

### C. The Partitioned Mixture Distribution

A useful extension of the standard MD is a set of coupled MDs, called a partitioned mixture distribution (PMD), Luttrell [5]. The basic idea is to retain the additional variable  $c$ , and to construct a large number of MD models, each of which uses only a subset of the states that are available to  $c$ , and each of which sees only a subspace of the input data  $\mathbf{x}$ .

For simplicity, the theory of a 1-dimensional PMD is presented; the extension to higher dimensions or alternative topologies is obvious. Assume that  $\mathbf{x}$  is  $N$ -dimensional,  $c$  has  $N$  possible states, and each MD uses only  $n$  of these states ( $n \leq N$ ,  $n$  is odd-valued). Each MD is then defined as (assuming circular boundary conditions, so that the class label  $c$  is modulo  $N$ )

$$Q_c(\mathbf{x}) = \frac{\sum_{c'=c-\frac{n}{2}}^{c'=\frac{n}{2}} Q(\mathbf{x}|c') Q(c')}{\sum_{c'=c-\frac{n}{2}}^{c'=\frac{n}{2}} Q(c')} \quad c = 1, 2, \dots, N \quad (2.4)$$

where the integer part of  $\frac{n}{2}$  is used. The set of all  $N$  MDs is the PMD.

The summations used in this definition of  $Q_c(\mathbf{x})$  can be generalised to any symmetric weighted summation.

The network representation of a PMD is shown schematically in Figure 2, where multiple overlapping MDs are computed. The phrase “partitioned mixture distribution” derives from the fact that the state space of the class label  $c$  is partitioned into overlapping sets of  $n$  states.

The network in Figure 2 can be extended laterally without encountering any scaling problems because its connectivity is local. However, it is not able to capture long-range statistical properties of the input vector  $\mathbf{x}$  (e.g. the correlation between pixels on opposite sides of an image) because no partition simultaneously sees information from widely separated sources. A multilayer PMD network based on the “Adaptive Cluster Expansion” approach, Luttrell [2, 3], is needed to remove this limitation.

### D. Optimising the Network Parameters

The PMD network is optimised by maximising the geometric mean of the likelihoods of the  $N$  MD models.

$$\mathbf{s}(\mathbf{x}^-) = \arg \max_{\mathbf{s}} \prod_{c=1}^N Q_c(\mathbf{x}^- | \mathbf{s}) \quad (2.5)$$

In the limit of a large training set size this is equivalent to maximising the average logarithmic likelihood.

$$\mathbf{s}(\mathbf{x}^-) = \arg \max_{\mathbf{s}} \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log Q_c(\mathbf{x} | \mathbf{s}) \quad (2.6)$$

The average over training data has been replaced by a shorthand notation in which the PDF  $P(\mathbf{x})$  is used to represent the density of training vectors, as follows

$$\frac{1}{T} \sum_{t=1}^T (\dots) \Big|_{\mathbf{x}=\mathbf{x}_t} \longrightarrow \int d\mathbf{x} P(\mathbf{x}) (\dots) \quad (2.7)$$

The posterior probability for model  $c$  is (omitting the  $\mathbf{s}$  variables)

$$Q_c(c'|\mathbf{x}) = \begin{cases} \frac{Q(\mathbf{x}, c')}{\sum_{c''=c-\frac{n}{2}}^{c'=\frac{n}{2}} Q(\mathbf{x}, c'')} & |c' - c| \leq \frac{n}{2} \\ 0 & |c' - c| > \frac{n}{2} \end{cases} \quad (2.8)$$

which satisfies the normalisation condition  $\sum_{c'=c-\frac{n}{2}}^{c'=\frac{n}{2}} Q_c(c'|\mathbf{x}) = 1$ .

The sum over model index of this posterior probability is

$$\tilde{Q}(c|\mathbf{x}) \equiv \sum_{c'=c-\frac{n}{2}}^{c'=\frac{n}{2}} Q_{c'}(c|\mathbf{x}) = \sum_{c'=c-\frac{n}{2}}^{c'=\frac{n}{2}} \frac{Q(\mathbf{x}, c)}{\sum_{c''=c'-\frac{n}{2}}^{c'=\frac{n}{2}} Q(\mathbf{x}, c'')} \quad (2.9)$$

and similar expression for the sum of the prior probabilities is

$$\tilde{Q}(c) \equiv \sum_{c'=c-\frac{n}{2}}^{c'=\frac{n}{2}} \frac{Q(c)}{\sum_{c''=c'+\frac{n}{2}}^{c'=\frac{n}{2}} Q(c'')} \quad (2.10)$$

These definitions satisfy the normalisation conditions  $\sum_{c=1}^N \tilde{Q}(c|\mathbf{x}) = \sum_{c=1}^N \tilde{Q}(c) = N$ . Normalisation to  $N$ , rather than 1, is used to ensure that  $\tilde{Q}(c|\mathbf{x})$  and  $\tilde{Q}(c)$  do not depend on the dimensionality  $N$  of the input data. Note that this “tilde-transform” can be used heuristically to normalise any array of numbers.

For clarity, two alternative prescriptions for optimising  $\mathbf{s}$  are now presented without proof. The detailed derivations of these two prescriptions can be found in Appendix A

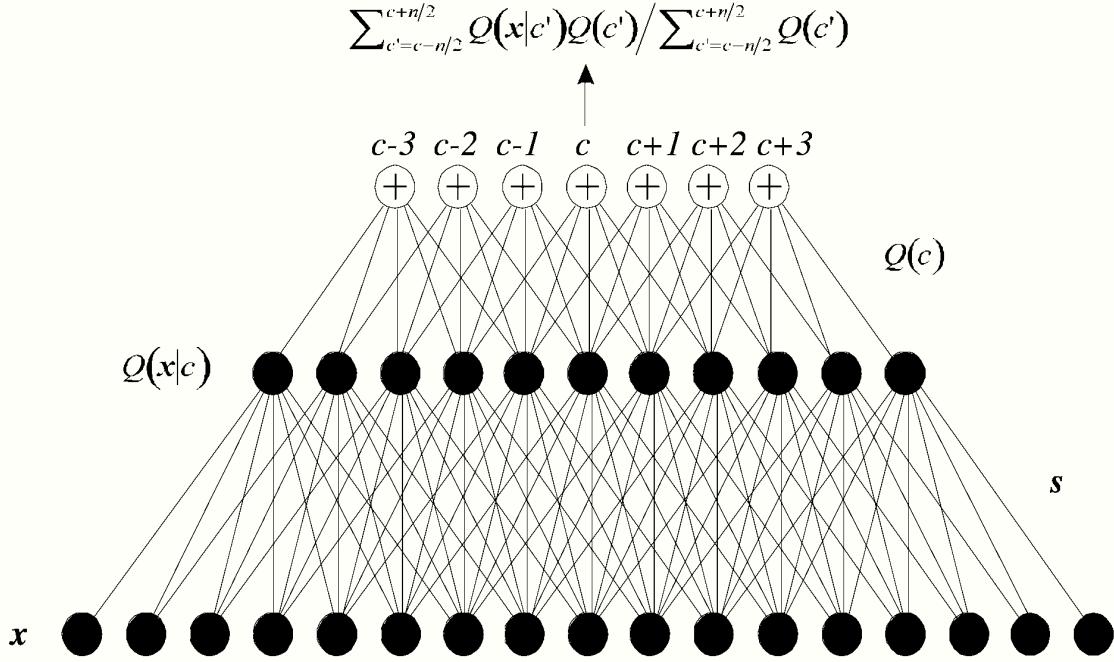


Figure 2: Partitioned mixture distribution network.

1. Re-estimation prescription:

$$\hat{s} = \arg \max_{s'} \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \left( \tilde{Q}(c|\mathbf{x}, s) \log Q(\mathbf{x}, c|s') - \log \sum_{c'=-\frac{n}{2}}^{\frac{n}{2}} Q(c'|s') \right) \quad (2.11)$$

2. Gradient ascent prescription:

$$\Delta s \propto \arg \max_{s'} \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \left( \tilde{Q}(c|\mathbf{x}, s) \frac{\partial \log Q(\mathbf{x}, c|s)}{\partial s} - \tilde{Q}(c|s) \frac{\partial \log Q(c|s)}{\partial s} \right) \quad (2.12)$$

In both of these prescriptions, the second term derives from the denominator of the PMD expression, and in the case  $n = N$  the standard MD prescriptions emerge.

### E. The Gaussian Partitioned Mixture Distribution

A popular model to use for the  $Q(\mathbf{x}|c)$  is a set of Gaussian PDFs with means  $\mathbf{m}(c)$  and covariances  $\mathbf{A}(c)$ . The prior probabilities  $Q(c)$  are their own parameters (i.e. non-parametric).

Define some average moments as

$$\begin{pmatrix} M_0 \\ M_1 \\ M_2 \end{pmatrix}(c) \equiv \int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}) \begin{pmatrix} 1 \\ \mathbf{x} \\ \mathbf{x}\mathbf{x}^T \end{pmatrix} \quad (2.13)$$

The re-estimation equations become

$$\begin{aligned} \tilde{Q}(c) &= M_0(c) \\ \hat{\mathbf{m}}(c) &= \frac{\mathbf{M}_1(c)}{M_0(c)} \\ \hat{\mathbf{A}}(c) &= \frac{\mathbf{M}_2(c)}{M_0(c)} - \hat{\mathbf{m}}(c) \hat{\mathbf{m}}(c)^T \end{aligned} \quad (2.14)$$

The re-estimation equation for  $\hat{Q}(c)$  is the same whether not a Gaussian model is assumed, and it must be solved iteratively. The re-estimation equations for  $\hat{\mathbf{m}}(c)$  and  $\hat{\mathbf{A}}(c)$  are straightforward to solve.

The gradient ascent equations become

$$\begin{aligned} \Delta Q(c) &\propto \frac{\tilde{Q}(c|\mathbf{x}) - \tilde{Q}(c)}{\tilde{Q}(c)} \\ \Delta \mathbf{m}(c) &\propto \tilde{Q}(c|\mathbf{x}) (\mathbf{x} - \mathbf{m}(c)) \\ \Delta \mathbf{A}^{-1}(c) &\propto \tilde{Q}(c|\mathbf{x}) (\mathbf{A}(c) - (\mathbf{x} - \mathbf{m}(c)) (\mathbf{x} - \mathbf{m}(c))^T) \end{aligned} \quad (2.15)$$

These results for a Gaussian PMD reduce to the standard results for the corresponding Gaussian MD when  $n = N$ .

### III. NUMERICAL SIMULATIONS

Numerical simulations were performed, using images as the input data, in order to demonstrate some of the emergent properties of a PMD network. The network architecture that was used in these simulations is shown in Figure 3. (which should be compared with Figure 2).

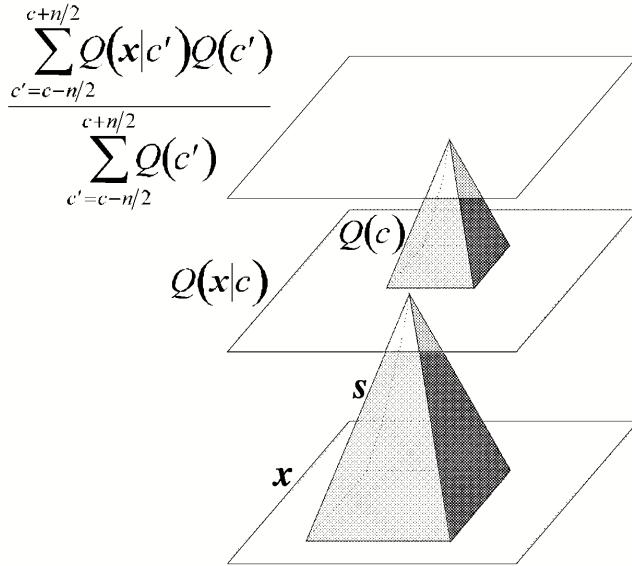


Figure 3: Partitioned mixture distribution network for image processing.

#### A. Completeness of Partitioned Mixture Distribution

This simulation was conducted to demonstrate that each partition of the PMD has sufficient resources to construct a full MD within its input window. This is a type of completeness property.

The network was a  $16 \times 16$  toroidal array, each input window size was  $9 \times 9$ , each mixture window size was  $5 \times 5$ . The training data was normalised by applying the tilde-transform discussed earlier, the variance of the Gaussians was fixed at 0.5, and (for convenience only) the Gaussian means became topographically ordered by including Kohonen-style neighbourhood updates.

Note how a complete repertoire of Gaussian means can be found within each  $5 \times 5$  mixture window, wherever the window is located. The topographic ordering ensures that the Gaussian mean varies smoothly across the montage. Topographic ordering is not necessary for a PMD to function correctly; it is introduced to make the montage easier to interpret visually. However, in a multilayer

PMD network (not studied here) topographic ordering is actually needed for non-cosmetic reasons.

In more sophisticated simulations (e.g. pairs of input images) the striations that occur in the montage can be directly related to the “dominance columns” that are observed in the visual cortex.

#### B. Translation Invariant Partitions

This simulation was conducted to demonstrate that a PMD network can compute MD models in an approximately translation invariant fashion, despite being non-translation invariant at the level of its class probabilities. A toroidal  $64 \times 64$  network was used.

In Figure 5a note that the class probabilities fluctuate dramatically as the input data is moved past the network, but in Figure 5b the mixture probabilities move with the data and fluctuate only slightly. This invariance property is important in image processing applications, where all parts of the image are (initially) treated on an equal footing. For obvious reasons, it is convenient to refer to the images in Figure 5 as “probability images”.

#### C. Normalised Partitioned Mixture Distribution Inputs

This simulation was conducted to demonstrate what happens when the input to each Gaussian is normalised. This forces the PMD to concentrate on the variation of the input data (rather than its absolute value) within each input window. A toroidal  $20 \times 20$  network was used.

In Figure 6a the PMD network develops Gaussian means that resemble small blob or bar-like features with various displacements and orientations. For convenience, the displacement of the centre of mass of each Gaussian mean (with respect to its central pixel) is shown in Figure 6b. This resembles an “orientation map”, as observed in the visual cortex.

Each partition of the PMD contains the full repertoire of Gaussian means that is needed to construct an MD using normalised inputs.

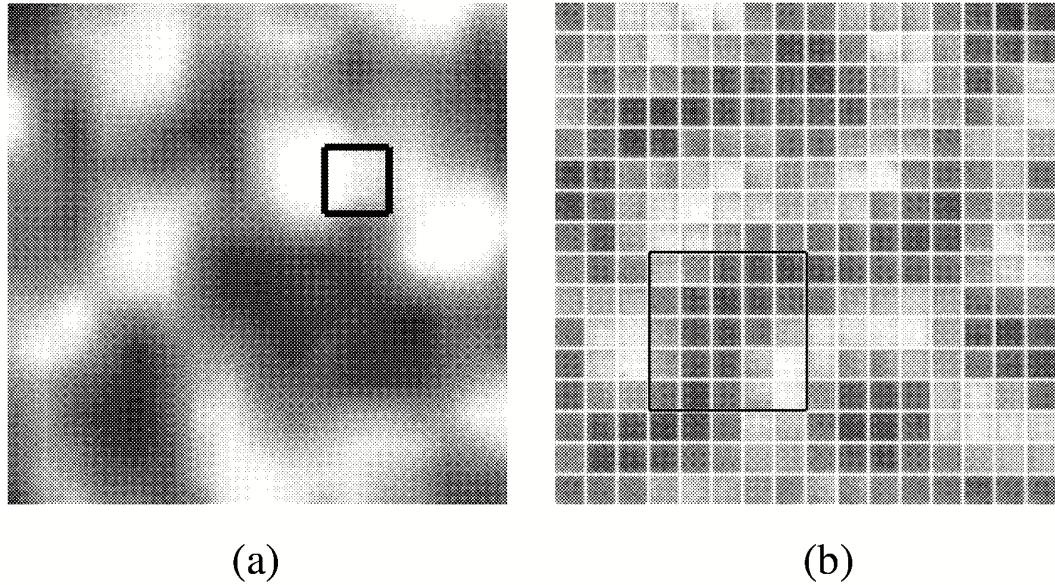


Figure 4: (a) Example of a training image. The superimposed square indicates the size of an input window. (b) Montage of Gaussian means after training. The superimposed square indicates the size of a mixture window.

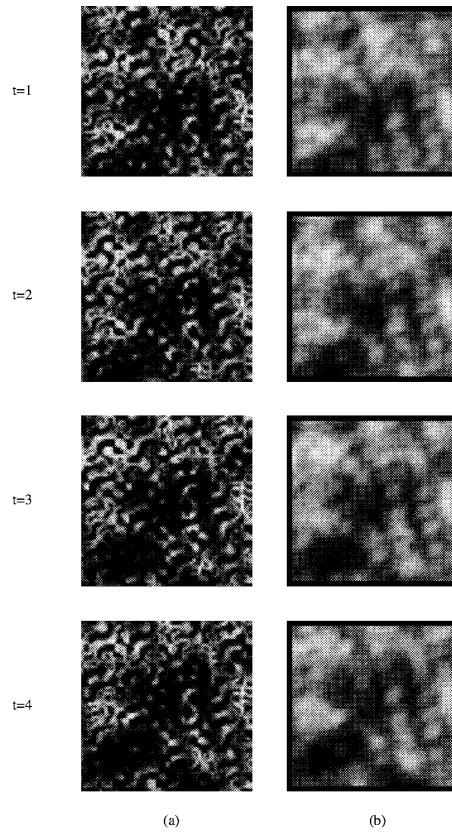


Figure 5: Response patterns that occur as the input image is moved upwards past the network. (a) Class probabilities. (b) Mixture probabilities.

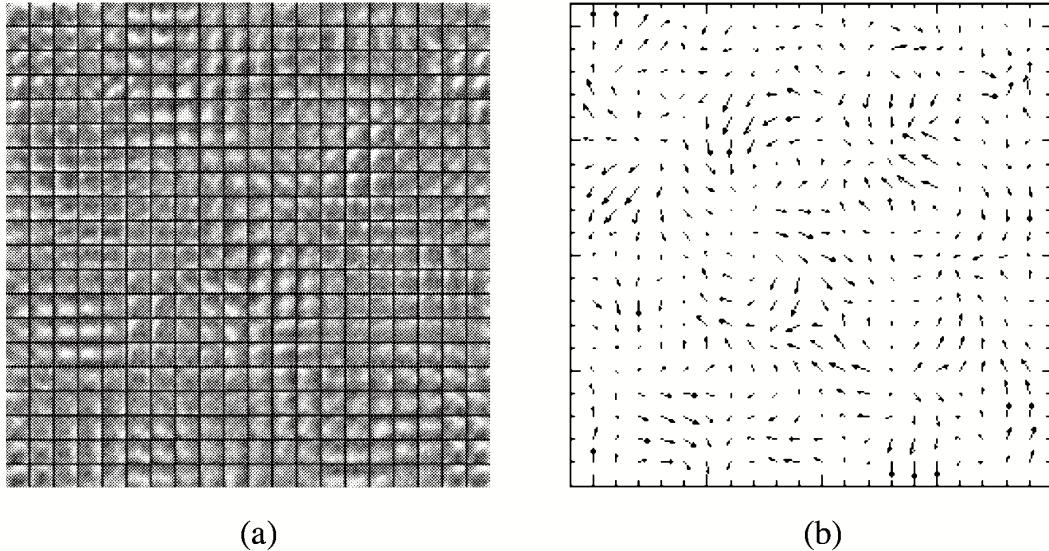


Figure 6: (a) Montage of Gaussian means after training with normalised input vectors. (b) Vector field of displacements after training with normalised input vectors.

#### IV. CONCLUSIONS

Partitioned mixture distribution networks are a scalable generalisation of standard mixture distributions. Ultimately, as in all Bayesian networks, the power of these networks derives from the use of probabilities as the basic computational objects.

The results of the numerical simulations demonstrate

that PMD networks have many desirable properties in common with the visual cortex. Multilayer versions of PMD networks, Luttrell [2, 3], have the potential to extend this correspondence.

PMD networks have a structure that is amenable to hardware implementation. This opens up the possibility of constructing a fast low-level vision engine based entirely on rigorous Bayesian principles.

#### Appendix A

This appendix contains the derivations of two prescriptions for maximising the average logarithmic likelihood that the set of models contained in a PMD fits the training data. The last line of each of these derivations appears in the main text without proof.

### 1. Re-estimation Prescription

$$\begin{aligned}
\Delta L(\mathbf{s}'; \mathbf{s}) &= L(\mathbf{s}') - L(\mathbf{s}) \\
&= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log \left( \frac{Q_c(\mathbf{x}|\mathbf{s}')}{Q_c(\mathbf{x}|\mathbf{s})} \right) \\
&= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log \left( \frac{\sum_{c'=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q(\mathbf{x}, c'|\mathbf{s}')}{Q_c(\mathbf{x}|\mathbf{s}) \sum_{c''=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q(c''|\mathbf{s}')} \right) \\
&= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log \left( \frac{\sum_{c'=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{s}) \frac{Q(\mathbf{x}, c'|\mathbf{s}')}{Q_c(\mathbf{x}, c'|\mathbf{s}) \sum_{c''=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q(c''|\mathbf{s}'')}}{Q_c(\mathbf{x}, c'|\mathbf{s}) \sum_{c''=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q(c''|\mathbf{s}'')} \right) \\
&\geq \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \sum_{c'=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{s}) \log \left( \frac{Q(\mathbf{x}, c'|\mathbf{s}') \sum_{c''=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q(c''|\mathbf{s})}{Q(\mathbf{x}, c'|\mathbf{s}) \sum_{c''=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q(c''|\mathbf{s}'')} \right) \\
&= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \left( \tilde{Q}(c|\mathbf{x}, \mathbf{s}) \log Q(\mathbf{x}, c|\mathbf{s}') - \log \left( \sum_{c'=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q(c'|\mathbf{s}') \right) \right) \\
&\quad + \text{a term that is independent of } \mathbf{s}' \tag{A1}
\end{aligned}$$

The penultimate step was obtained by using Jensen's inequality for convex functions, and the last step was obtained by using the result  $\sum_{c'=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q_c(c'|\mathbf{x}) = 1$  and the result

$$\begin{aligned}
\sum_{c=1}^N \sum_{c'=c-\frac{n}{2}}^{c=c+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{s}) (\dots) &= \sum_{c'=1}^N \sum_{c=c'-\frac{n}{2}}^{c=c'+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{s}) (\dots) \\
&= \sum_{c'=1}^N \tilde{Q}(c'|\mathbf{x}, \mathbf{s}) (\dots) \tag{A2}
\end{aligned}$$

The maximisation of  $L(\mathbf{s}')$  can now be replaced by the maximisation of its lower bound  $\Delta L(\mathbf{s}'; \mathbf{s})$  with respect to  $\mathbf{s}'$ , which immediately yields the re-estimation equation.

## 2. Gradient Ascent Prescription

The gradient ascent prescription can be similarly derived by directly differentiating the logarithmic likelihood with respect to the parameter vector.

$$\begin{aligned}
\frac{\partial L(\mathbf{s})}{\partial \mathbf{s}} &= \frac{\partial}{\partial \mathbf{s}} \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log Q_c(\mathbf{x}|\mathbf{s}) \\
&= \frac{\partial}{\partial \mathbf{s}} \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log \left( \frac{\sum_{\substack{c'=c+\frac{n}{2} \\ c'=c-\frac{n}{2}}} Q(\mathbf{x}, c'|\mathbf{s})}{\sum_{\substack{c'=c+\frac{n}{2} \\ c''=c-\frac{n}{2}}} Q(c''|\mathbf{s})} \right) \\
&= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \frac{\sum_{\substack{c'=c+\frac{n}{2} \\ c''=c-\frac{n}{2}}} Q(\mathbf{x}, c'|\mathbf{s}) \frac{\partial \log Q(\mathbf{x}, c'|\mathbf{s})}{\partial \mathbf{s}}}{\sum_{\substack{c'=c+\frac{n}{2} \\ c''=c-\frac{n}{2}}} Q(\mathbf{x}, c''|\mathbf{s})} - \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \frac{\sum_{\substack{c'=c+\frac{n}{2} \\ c''=c+\frac{n}{2}}} Q(c'|\mathbf{s}) \frac{\partial \log Q(c'|\mathbf{s})}{\partial \mathbf{s}}}{\sum_{\substack{c'=c+\frac{n}{2} \\ c''=c-\frac{n}{2}}} Q(c''|\mathbf{s})} \\
&= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \left( \tilde{Q}(c|\mathbf{x}, \mathbf{s}) \frac{\partial \log Q(\mathbf{x}, c|\mathbf{s})}{\partial \mathbf{s}} - \tilde{Q}(c|\mathbf{s}) \frac{\partial \log Q(c|\mathbf{s})}{\partial \mathbf{s}} \right)
\end{aligned} \tag{A3}$$

As in the re-estimation prescription, the  $c$  and  $c'$  summations are interchanged to obtain the final result.

- [1] R T Cox, *Probability, frequency and reasonable expectation*, American Journal of Physics **14** (1946), no. 1, 1–13.
- [2] S P Luttrell, *A trainable texture anomaly detector using the adaptive cluster expansion (ACE) method*, Memorandum 4437, Royal Signals and Radar Establishment, Malvern, 1990.
- [3] ———, *A hierarchical network for clutter and texture modelling*, Proceedings of SPIE conference on adaptive signal processing (San Diego), SPIE, 1991, pp. 518–528.
- [4] ———, *Adaptive Bayesian networks*, Proceedings of SPIE conference on adaptive signal processing (Orlando), SPIE, 1992, pp. 140–151.
- [5] ———, *Partitioned mixture distributions: an introduction*, Memorandum 4671, Defence Research Agency, 1992.

# An Adaptive Bayesian Network for Texture Modelling \*

Stephen P Luttrell<sup>†</sup>

*Adaptive Systems Theory Section, Defence Research Agency,  
St Andrews Rd, Malvern, Worcestershire, WR14 3PS.*

Bayesian methods are used to analyse the problem of training a model to make predictions about the distribution of data that has yet to be received. Mixture distributions emerge naturally from this framework, but are not well-matched to high-dimensional problems such as arise in image processing applications. An extension to partitioned mixture distributions (PMD) is presented, which is essentially a set of overlapping mixture distributions, and an expectation-maximisation training algorithm is derived. Finally, the results of some numerical simulations are presented, which demonstrate that lateral inhibition arises naturally in PMDs, and that the nodes in a PMD co-operate in such a way that each mixture distribution receives a full complement of what is needed for it to compute a mixture distribution.

## I. INTRODUCTION

The purpose of this paper is to introduce a novel adaptive network for performing low-level image processing operations. Unlike conventional solutions to this problem in which image operators are hand crafted to extract information from images, the adaptive approach taken in this paper discovers for itself how best to process images. Furthermore, in order to ensure that the image processing network is principled, rather than ad hoc, it is derived from Bayesian techniques of information processing, in which probability density functions (PDFs) are used to represent information Cox [1]. In this paper an extension of the standard mixture distribution model, called a partitioned mixture distribution (PMD), is introduced. When it is represented as a 2-layer network it consists of two sheets of nodes (an input layer and an output layer). The state of each node in the output layer depends on the states of the nodes within a limited window of the input layer, and also depends on the states of the neighbouring nodes in the output layer. This limited connectivity allows PMD networks to scale sensibly to high-dimensional problems, such as image processing. Several numerical simulations are performed which demonstrate that the emergent properties of PMD networks are similar to those of biological low-level vision processing systems.

## II. THE BAYESIAN APPROACH

In this section the use of PDFs as a vehicle for solving inference problems is discussed. The Bayesian approach to building a PDF model for use in inference problems is characterised by several standard steps.

Step 1: Select a State Space. Choose a state space to describe the system under study; denote a vector in this space as  $\mathbf{x}$ . For instance, in the case of image processing, the state space might be the space of all images, in which case  $\mathbf{x}$  would be the pixel values of a single image arranged in the form of a vector.

Step 2: Select a PDF model. Define a PDF  $Q(\mathbf{x})$  to model the joint probability (density) of the components of the vector  $\mathbf{x}$ .  $Q(\mathbf{x})$  will have one or more internal variables (or parameters), which may be represented as a vector  $\mathbf{s}$  which is unknown at the outset. Thus  $Q(\mathbf{x})$  is obtained as follows

$$Q(\mathbf{x}) = \int d\mathbf{s} Q(\mathbf{x}|\mathbf{s}) Q(\mathbf{s}) \quad (2.1)$$

where  $Q(\mathbf{x}|\mathbf{s})$  is the PDF with a specific parameter vector inserted, and  $Q(\mathbf{s})$  is a “prior PDF” for the parameter vector. Thus the PDF  $Q(\mathbf{x})$  is a weighted average over the possible contributing PDFs  $Q(\mathbf{x}|\mathbf{s})$ . Note that the  $Q(\dots)$  notation is used generically to denote “the PDF of ...” and should not be taken to imply that  $Q(\dots)$  is a fixed function of its arguments. The notation will be made more explicit only where the meaning is not clear from the context.

Step 3: Select a Training Set. Define a training set  $\mathbf{X}$  of vectors drawn from the state space of the system. If there are  $M$  such vectors then  $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M)$ . Assuming that the training samples are drawn independently from their underlying distribution, the likelihood of the training set  $\mathbf{X}$  given a particular choice of the parameter vector  $\mathbf{s}$  is given by

$$Q(\mathbf{X}|\mathbf{s}) = \prod_{i=1}^M Q(\mathbf{x}_i|\mathbf{s}) \quad (2.2)$$

and the logarithmic likelihood is given by

$$L(\mathbf{X}|\mathbf{s}) \equiv \log Q(\mathbf{X}|\mathbf{s}) = \sum_{i=1}^M \log Q(\mathbf{x}_i|\mathbf{s}) \quad (2.3)$$

Step 4: Train the PDF Model. Present the training data  $\mathbf{X}$  to the model  $Q(\mathbf{x})$  in order that it can learn

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

This appeared in Proceedings of the IEE Seminar on Texture Analysis in Radar and Sonar, London, 6/1-6/10, 1993. © Crown Copyright 1993/DRA.

<sup>†</sup>Electronic address: luttrell@signal.dra.hmg.gb

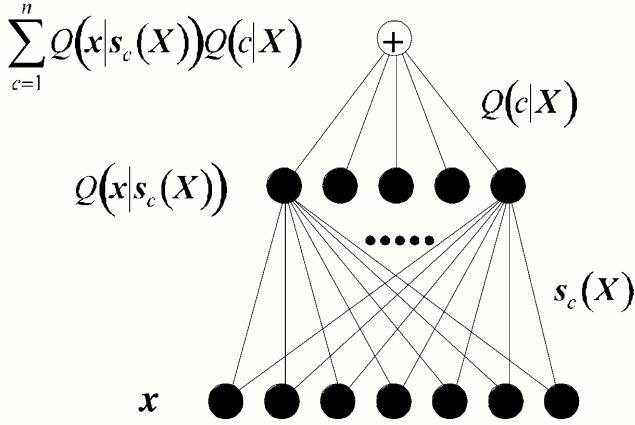


Figure 1: A mixture distribution represented as a neural network. The bottom layer represents the components of the input vector  $\mathbf{x}$ . The middle layer represents the output  $Q(\mathbf{x}|\mathbf{s}_c(\mathbf{X}))$  for each possible value of the class label  $c$ ; these are the class likelihoods. The top layer is the network output, which is a linear combination (or mixture) of the various  $Q(\mathbf{x}|\mathbf{s}_c(\mathbf{X}))$  weighted by the class prior probabilities  $Q(c|\mathbf{X})$ .

something about the true state of its internal parameter vector  $\mathbf{s}$ . The overall effect of this is to convert the prior PDF  $Q(\mathbf{s})$  into a new PDF  $Q(\mathbf{s}|\mathbf{X})$  that embodies the prior information contained in  $Q(\mathbf{s})$  plus the additional information contained in  $\mathbf{X}$ . This conversion trick is implemented thus

$$\begin{aligned} Q(\mathbf{s}|\mathbf{X}) &= \frac{Q(\mathbf{X}|\mathbf{s})Q(\mathbf{s})}{\int d\mathbf{s}' Q(\mathbf{X}|\mathbf{s}')Q(\mathbf{s}')} \\ &= Q(\mathbf{s}) \frac{\prod_{i=1}^M Q(\mathbf{x}_i|\mathbf{s})}{\int d\mathbf{s}' \prod_{i=1}^M Q(\mathbf{x}_i|\mathbf{s}')Q(\mathbf{s}')} \end{aligned} \quad (2.4)$$

$$\mathbf{s}(\mathbf{X}) = \begin{cases} \arg \max_{\mathbf{s}} Q(\mathbf{X}|\mathbf{s}) & \text{maximum likelihood} \\ \arg \max_{\mathbf{s}} Q(\mathbf{s}|\mathbf{X}) & \text{maximum posterior probability} \end{cases} \quad (2.7)$$

The maximum likelihood prescription may be written out explicitly as

$$\mathbf{s}(\mathbf{X}) = \arg \max_{\mathbf{s}} \sum_{i=1}^M \log Q(\mathbf{x}_i|\mathbf{s}) \quad (2.8)$$

A generalisation of Equation 2.6 is the following

$$\begin{aligned} Q(\mathbf{x}|\mathbf{X}) &\approx \int d\mathbf{s} Q(\mathbf{x}|\mathbf{s}) \sum_{c=1}^n Q(c|\mathbf{X}) \delta(\mathbf{s} - \mathbf{s}_c(\mathbf{X})) \\ &= \sum_{c=1}^n Q(\mathbf{x}|\mathbf{s}_c(\mathbf{X})) Q(c|\mathbf{X}) \end{aligned} \quad (2.9)$$

where the product ranges over the (assumed) statistically independent training vectors. The PDF  $Q(\mathbf{s}|\mathbf{X})$  is the Bayes' inverse of the product of PDFs  $\prod_{i=1}^M Q(\mathbf{x}_i|\mathbf{s})$ . The training process converts  $Q(\mathbf{x})$  into  $Q(\mathbf{x}|\mathbf{X})$  thus

$$Q(\mathbf{x}|\mathbf{X}) = \int d\mathbf{s} Q(\mathbf{x}|\mathbf{s}) Q(\mathbf{s}|\mathbf{X}) \quad (2.5)$$

which is a weighted average over the possible contributing PDFs  $Q(\mathbf{x}|\mathbf{s})$ , with the untrained prior PDF  $Q(\mathbf{s})$  replaced by the trained PDF  $Q(\mathbf{s}|\mathbf{X})$ .

Step 5: Approximate the PDF Model.  $Q(\mathbf{x}|\mathbf{X})$  is usually difficult to calculate exactly because it involves integration(s) over alternative parameter vectors  $\mathbf{s}$ , so an approximate expression for  $Q(\mathbf{x}|\mathbf{X})$  is required. There are many ways in which approximations may be made, depending on the peculiarities of the  $\mathbf{s}$  integral and on what  $Q(\mathbf{x}|\mathbf{X})$  will be used for later on. However, if the integral is dominated by contributions from around a single value of  $\mathbf{s}$ , then  $Q(\mathbf{x}|\mathbf{X})$  can be approximated by a point PDF  $\delta(\mathbf{s} - \mathbf{s}(\mathbf{X}))$  located at  $\mathbf{s}(\mathbf{X})$ . This simplifies  $Q(\mathbf{x}|\mathbf{X})$  as follows

$$\begin{aligned} Q(\mathbf{x}|\mathbf{X}) &= \int d\mathbf{s} Q(\mathbf{x}|\mathbf{s}) \delta(\mathbf{s} - \mathbf{s}(\mathbf{X})) \\ &= Q(\mathbf{x}|\mathbf{s}(\mathbf{X})) \end{aligned} \quad (2.6)$$

The optimum location  $\mathbf{s}(\mathbf{X})$  of this point PDF  $\delta(\mathbf{s} - \mathbf{s}(\mathbf{X}))$  is debatable. Two popular schemes are

in which  $Q(\mathbf{s}|\mathbf{X})$  is approximated by a weighted sum of  $n$  point PDFs  $\delta(\mathbf{s} - \mathbf{s}_c(\mathbf{X}))$ , and the  $Q(c|\mathbf{X})$  are normalised as  $\sum_{c=1}^n Q(c|\mathbf{X}) = 1$ . This type of approximation to  $Q(\mathbf{x}|\mathbf{X})$  is called a mixture distribution because  $Q(\mathbf{x}|\mathbf{X})$  is approximated by a PDF that is a weighted mixture of  $n$  separate PDFs  $Q(\mathbf{x}|\mathbf{s}_c(\mathbf{X}))$ . The  $\mathbf{s}_c(\mathbf{X})$  may be optimised according to any of the prescriptions used in the single point PDF case (see Equation 2.7).

To make contact with the more usual terminology of density estimation theory,  $c$  is usually referred to as a class label, in which case  $Q(c|\mathbf{X})$  is the prior probability of class  $c$ , and  $Q(\mathbf{x}|\mathbf{s}_c(\mathbf{X}))$  is the likelihood that data

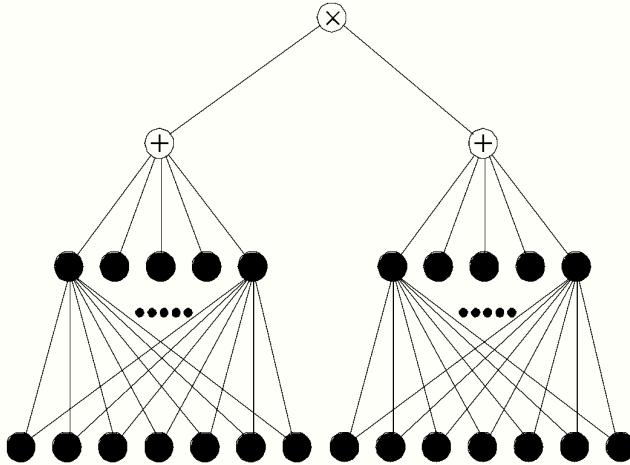


Figure 2: A pair of independent mixture distribution networks. The final output is the product of the output of the pair of mixture distribution networks. The left hand half of the network has input vector  $\mathbf{x}^1$ , parameter vector  $s_{c^1}(\mathbf{X}^1)$ , prior probabilities  $Q(c^1|\mathbf{X}^1)$  and output parameter vectors  $\sum_{c^1=1}^{n^1} Q^k(\mathbf{x}^1|s_{c^1}(\mathbf{X}^1)) Q(c^1|\mathbf{X}^1)$ . Similarly, for the right hand half, but exchange the superscript 1 for a superscript 2. These networks compute a pair of mixture distributions for the left and right halves of the overall input vector  $(\mathbf{x}^1, \mathbf{x}^2)$ , respectively. The final network output is then the product of the outputs of the left and right half networks; this procedure assumes that the left and right halves of the input vector are independent, or, at least, that any correlations between left and right may be ignored.

vector  $\mathbf{x}$  is produced by class  $c$ .

### III. THE NEURAL NETWORK INTERPRETATION OF A MIXTURE DISTRIBUTION

In the previous section it was shown how to calculate the PDF  $Q(\mathbf{x}|\mathbf{X})$  for a particular choice of model, and the idea of using a mixture distribution approximation was introduced (see Equation 2.9). The optimum choice of parameter vectors  $s_c(\mathbf{X})$  and weights  $Q(c|\mathbf{X})$  to use in a mixture distribution is determined by the training data  $\mathbf{X}$  and the chosen optimisation criterion. It will be assumed henceforth that the maximum likelihood prescription is the chosen optimisation scheme. This can be interpreted in “neural network” language as follows. The parameters  $s_c(\mathbf{X})$  correspond to the neural network weights (or whatever) that emerge from the training process. Subsequent computation of the mixture distribution approximation to  $Q(\mathbf{x}|\mathbf{X})$  corresponds to testing the neural network in the field. This correspondence between PDF models and neural networks will be tacitly assumed throughout this paper.

In Figure 1 a mixture distribution is drawn as if it were a neural network. The input layer is transformed through a set of non-linear functions to produce a pattern of hidden layer outputs  $Q(\mathbf{x}|s_c(\mathbf{X}))$  indexed by the label  $c$ , which are then transformed via a linear summation operation weighted by the prior probabilities  $Q(c|\mathbf{X})$  to produce the mixture distribution  $\sum_{c=1}^n Q(\mathbf{x}|s_c(\mathbf{X})) Q(c|\mathbf{X})$ . The goal is to maximise the likelihood of the training data with respect to  $s_c(\mathbf{X})$  and  $Q(c|\mathbf{X})$  thus

$$\arg \max_{\{s_c(\mathbf{X}), Q(c|\mathbf{X}) : c = 1, 2, \dots, n\}} \sum_{c=1}^n \log \sum_{c'=1}^n Q(\mathbf{x}|s_{c'}(\mathbf{X})) Q(c') \quad (3.1)$$

which maximises the average over the  $M$  samples in the training set of the logarithm of network output. This is an example of an unsupervised training procedure.

### IV. MULTIPLE INDEPENDENT MIXTURE DISTRIBUTIONS

In situations where the dimensionality  $\dim \mathbf{x}$  of the input vector  $\mathbf{x}$  is large it is necessary to make some simplifying assumptions in order to construct a useful mixture distribution. For instance, an independence assumption can be made where  $\mathbf{x}$  is partitioned into  $N$  subspaces as  $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^N)$ , and likewise the training data  $\mathbf{X} = (\mathbf{X}^1, \mathbf{X}^2, \dots, \mathbf{X}^N)$ , and a mixture distribu-

tion  $Q^k(\mathbf{x}^k|\mathbf{X}^k) \quad k = 1, 2, \dots, N$  is constructed independently for each subspace. The overall PDF  $Q(\mathbf{x}|\mathbf{X})$  is then the product of the individual  $Q^k(\mathbf{x}^k|\mathbf{X}^k)$ .

$$Q(\mathbf{x}|\mathbf{X}) = \prod_{k=1}^N Q^k(\mathbf{x}^k|\mathbf{X}^k) \quad (4.1)$$

This expression does not take account of any correlations that might exist between the various  $\mathbf{x}^k$  for different values of  $k$ . Each  $Q^k(\mathbf{x}^k|\mathbf{X}^k)$  in this product could be approximated using a mixture distribution network of the type shown in Figure 1, and the outputs from the  $N$  such networks could be combined using a multiplier node

to create the following overall approximation to  $Q(\mathbf{x}|\mathbf{X})$

$$Q(\mathbf{x}|\mathbf{X}) \approx \prod_{k=1}^N \left( \sum_{c^k=1}^{n^k} Q^k(\mathbf{x}^k|s_{c^k}(\mathbf{X}^k)) Q(c^k|\mathbf{X}^k) \right) \quad (4.2)$$

where an obvious generalisation of the earlier notation has been used. The maximum likelihood prescription then leads to  $N$  independent optimisation problems of the form given in Equation 3.1. For  $N = 2$  the neural network representation of this approximation is shown in Figure 2. This network can readily be generalised to arbitrary values of  $N$ .

## V. MULTIPLE OVERLAPPING MIXTURE DISTRIBUTIONS (THE PARTITIONED MIXTURE DISTRIBUTION)

The type of network shown in Figure 2 is undesirable for data which does not clearly divide into a number of separate subspaces, such as image data. What is required is a network structure that treats each patch of data on an equal footing, unlike Figure 2 which partitions the input vector  $\mathbf{x}$  in left and right halves as  $(\mathbf{x}^1, \mathbf{x}^2)$ . There is a two stage derivation of such a network, starting with a standard mixture distribution network, and finishing with what will be called a partitioned mixture distribution (PMD) network, Luttrell [4]. For concreteness, this derivation will refer to the network in Figure 2, but the technique may be applied to arbitrary  $N$ .

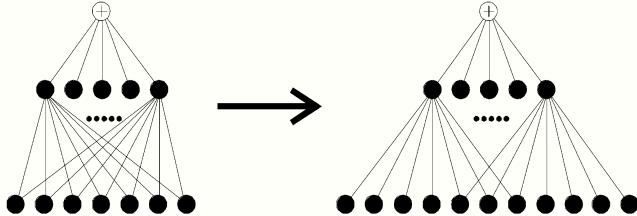


Figure 3: Modification of the basic mixture distribution network. In the standard mixture distribution each mixture component (or class) sees the whole of the input space, whereas in the modified version each class sees only part of the input space, as indicated by the connections drawn in the right hand diagram. This has the effect that as the class label is varied the field of view of the corresponding mixture component moves across the input data. This has important ramifications for the processing of high dimensional input data where no natural division into left and right halves, as used in Figure 2, occurs.

Step 1: Modify the mixture distribution network. In the first stage of the derivation the structure of the network is changed so as to make it possible to construct translation invariant networks when  $N$  is large. The basic trick is to modify the mixture distribution network as shown in Figure 3. In this modified mixture distribution each mixture component sees a different subspace of the

input data, although there is a large amount of overlap between the different components, as can be seen in the right hand diagram in Figure 3. An alternative interpretation of this is that each mixture component sees the whole of the input space, but that its likelihood function  $Q(\mathbf{x}|\mathbf{s}_c(\mathbf{X}))$  is designed to be sensitive only to the part of the input space to which it is joined by the corresponding connections in Figure 3. In the modified mixture distribution shown in Figure 3 the different components of the input vector  $\mathbf{x}$  receive a different degree of attention by the mixture distribution. For instance, the components at the far left and far right of the diagram are seen by only one component of the mixture distribution, whereas the central three components of  $\mathbf{x}$  are seen by all five components of the mixture distribution. Therefore, the scope for tailoring the shape of the mixture distribution  $Q(\mathbf{x}|\mathbf{X})$  is such that the PDF of the central components of the input vector are better modelled than those at the edges. This limitation is not present in the standard mixture distribution. With this modification, the network in Figure 2 would become as shown in Figure 4.

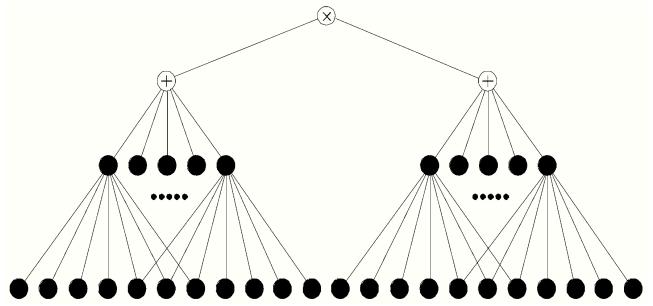


Figure 4: A pair of independent mixture distribution networks, as in Figure 2, but using the modified mixture distribution network shown in Figure 3.

Step 2: Tile the modified mixture distribution network. In the second stage of the derivation a very high dimensional input vector is considered. A network of the type shown in Figure 4 can see only a subspace of such an input vector, but different networks (again as in Figure 4) could be used to view different subspaces. It is convenient to build a composite network that implements simultaneously all possible placements of Figure 4 with respect to the input vector. This network is shown in Figure 5. If a single mixture distribution of the type shown in Figure 4 is required, then it may be obtained from the corresponding output node of the network shown in Figure 5. It is important to note that because of the overlap between the various subnetworks in Figure 5 their parameter vectors are not independent. In the neural network literature this is usually referred to as “weight sharing”.

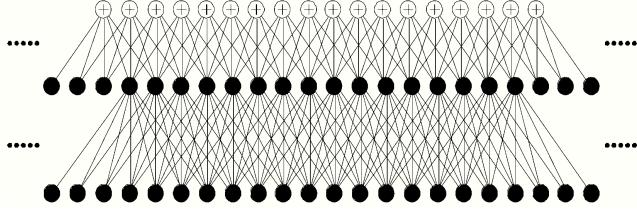


Figure 5: A superposition of mixture distribution networks of the type shown in Figure 4. Because the input vector is assumed to be high-dimensional, the dots on either side of the diagram indicate that the basic structure is to be replicated across the input vector. This network will be referred to as a partitioned mixture distribution (PMD) network.

## VI. PMD THEORY

The basic definition of a mixture distribution (see Equation 2.9) becomes

$$\begin{aligned} Q_c(\mathbf{x}|\mathbf{X}) &= \sum_{c'=\frac{n}{2}}^{\frac{c+\frac{n}{2}}{2}} Q(\mathbf{x}, c'|\mathbf{X}) \\ &= \frac{\sum_{c'=\frac{n}{2}}^{\frac{c+\frac{n}{2}}{2}} Q(\mathbf{x}|\mathbf{s}_{c'}(\mathbf{X})) Q(c'|\mathbf{X})}{\sum_{c'=\frac{n}{2}}^{\frac{c+\frac{n}{2}}{2}} Q(c'|\mathbf{X})} \\ &\quad c = 1, 2, \dots, N \end{aligned} \quad (6.1)$$

The index  $c$  selects which of the  $N$  mixture distributions is required, the integer-valued half-width parameter  $\frac{n}{2}$  is used to determine the number of components in each mixture distribution, and the normalisation factor  $\sum_{c'=\frac{n}{2}}^{\frac{c+\frac{n}{2}}{2}} Q(c'|\mathbf{X})$  ensures that each mixture distribution is normalised so that its integral over  $\mathbf{x}$  is unity. Note that the summations used in this definition of  $Q_c(\mathbf{x}|\mathbf{X})$  can be generalised to any weighted sum. Because each mixture distribution uses only part of the full range of possible values of  $c'$ , the set of overlapping mixture distributions will be referred to as a partitioned mixture distribution (PMD). For convenience, the augmented set  $\mathbf{S}$  of parameters will be defined as

$$\mathbf{S} \equiv \{\mathbf{s}_c, Q(c) : c = 1, 2, \dots, N\} \quad (6.2)$$

and the notation  $\mathbf{S}(\mathbf{X})$  will be used to denote a single choice of  $\mathbf{S}$  (such as the maximum likelihood choice) for training set  $\mathbf{X}$ . The maximum likelihood procedure for optimising  $\mathbf{S}$  for a set of  $N$  independent mixture distributions (see Equation 3.1) must now be generalised to a form that is appropriate for Figure 5. The simplest choice is to require that the geometric mean of the likelihoods (i.e. sum of the logarithmic likelihoods) of the various mixture distributions in Figure 5 be maximised,

which yields

$$\mathbf{S}(\mathbf{X}) = \arg \max_{\mathbf{S}} \sum_{c=1}^N \sum_{i=1}^M \log Q_c(\mathbf{x}_i|\mathbf{S}) \quad (6.3)$$

where

$$Q_c(\mathbf{x}_i|\mathbf{S}) = \frac{\sum_{c'=\frac{n}{2}}^{\frac{c+\frac{n}{2}}{2}} Q(\mathbf{x}_i|\mathbf{s}_{c'}(\mathbf{S})) Q(c')}{\sum_{c'=\frac{n}{2}}^{\frac{c+\frac{n}{2}}{2}} Q(c')} \quad (6.4)$$

The various parameter vectors now cooperate amongst themselves in order to maximise the average logarithmic likelihood. They must not only satisfy the internal constraints that arise from the overlap between the mixture distributions in the PMD, but they must also cooperate so as to ensure that each optimal mixture distribution  $Q_c(\mathbf{x}|\mathbf{S}(\mathbf{X}))$  in the PMD can do a good job of modelling the PDF of the subspace of  $\mathbf{x}$  that it sees. The network in Figure 5 can be extended laterally without encountering any scaling problems because its connectivity is local. However, it is not able to capture long-range statistical properties of the input vector  $\mathbf{x}$  (e.g. the correlation between pixels on opposite sides of an image) because no partition simultaneously sees information from widely separated sources. A multilayer PMD network based on the ‘‘Adaptive Cluster Expansion’’ approach, Luttrell [2, 3], is needed to remove this limitation.

## VII. PMD TRAINING ALGORITHM (BASIC RESULTS).

The basic mixture distribution optimisation prescription in Equation 3.1, and the generalised PMD prescription in Equation 6.3, need to be converted into an explicit training algorithm. There are two basic types of algorithm: the expectation-maximisation (EM) method, and the gradient ascent method. The former is suitable for batch training, whereas the latter is suitable for online training. In this paper only batch training will be considered, so the gradient ascent method will not be discussed. In the limit of a large training set size Equation 6.3 is equivalent to maximising the average logarithmic likelihood.

$$\mathbf{S}(\mathbf{X}) = \arg \max_{\mathbf{S}} \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log Q_c(\mathbf{x}|\mathbf{S}) \quad (7.1)$$

where the shorthand notation  $\mathbf{S} \equiv \{\mathbf{s}_c, Q(c) : c = 1, 2, \dots, N\}$  has been used, and the average over training data has been replaced by a shorthand notation in which the PDF  $P(\mathbf{x})$  is used to represent the density of training vectors, as follows

$$\frac{1}{M} \sum_{i=1}^M (\dots) \longrightarrow \int d\mathbf{x} P(\mathbf{x}) (\dots) \quad (7.2)$$

In PMD calculations it is useful to write the joint probability for the mixture distribution centred on class  $c$  as

$$Q_c(\mathbf{x}, c' | \mathbf{S}) = \frac{Q(\mathbf{x} | \mathbf{s}_{c'}) Q(c')}{\sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'')} \quad |c' - c| \leq \frac{n}{2} \quad (7.3)$$

and the corresponding posterior probability as

$$Q_c(c' | \mathbf{x}, \mathbf{S}) = \frac{Q(\mathbf{x} | \mathbf{s}_{c'}) Q(c')}{\sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(\mathbf{x} | \mathbf{s}_{c''}) Q(c'')} \quad |c' - c| \leq \frac{n}{2} \quad (7.4)$$

which satisfies the normalisation condition  $\sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_c(c' | \mathbf{x}, \mathbf{S}) = 1$ . In a PMD there are  $N$  separate mixture distributions, each with its own set of posterior probabilities, which occupy class labels  $c - \frac{n}{2}, c - \frac{n}{2} + 1, \dots, c + \frac{n}{2}$  for the  $c^{\text{th}}$  mixture distribution. These posterior probabilities overlap each other, and it turns out to be useful to define the sum over posterior probabilities at each class label

$$\begin{aligned} \tilde{Q}(c | \mathbf{x}, \mathbf{S}) &\equiv \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_{c'}(c | \mathbf{x}, \mathbf{S}) \\ &= Q(\mathbf{x} | \mathbf{s}_c) Q(c) \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} \frac{1}{\sum_{c''=c'-\frac{n}{2}}^{c'+\frac{n}{2}} Q(\mathbf{x} | \mathbf{s}_{c''}) Q(c'')} \end{aligned} \quad (7.5)$$

This definition ensures that the normalisation condition  $\sum_{c=1}^N \tilde{Q}(c | \mathbf{x}, \mathbf{S}) = N$  holds, so it could be used as a simple way of converting any function of  $c$  into a normalised function. Normalisation to  $N$ , rather than 1, is

used to ensure that  $\tilde{Q}(c | \mathbf{x}, \mathbf{S})$  does not depend on the dimensionality  $N$  of the input data. A similar definition for the sum of the prior probabilities also turns out to be useful.

$$\tilde{Q}(c | \mathbf{S}) = Q(c) \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} \frac{1}{\sum_{c''=c'-\frac{n}{2}}^{c'+\frac{n}{2}} Q(c'')} \quad (7.6)$$

which satisfies  $\sum_{c=1}^N \tilde{Q}(c | \mathbf{S}) = N$ .

### VIII. PMD TRAINING ALGORITHM (RE-ESTIMATION PRESCRIPTION OR THE EXPECTATION-MAXIMISATION (EM) METHOD).

The EM method is rather subtle, so a pictorial summary is given in Figure 6. The main purpose of the following derivation is to obtain a lower bound function as discussed in the caption to Figure 6.

First of all define the difference  $\Delta L(\mathbf{S}, \mathbf{S}_0; \mathbf{X})$  between the logarithmic likelihood evaluated for two different choices,  $\mathbf{S} \equiv \{\mathbf{s}_c, Q(c) : c = 1, 2, \dots, N\}$  and  $\mathbf{S}_0 \equiv \{\mathbf{s}_{0,c}, Q_0(c) : c = 1, 2, \dots, N\}$ , of the parameters

$$\begin{aligned} \Delta L(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) &\equiv L(\mathbf{X} | \mathbf{S}) - L(\mathbf{X} | \mathbf{S}_0) \\ &= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log \left( \frac{Q_c(\mathbf{x} | \mathbf{S})}{Q_c(\mathbf{x} | \mathbf{S}_0)} \right) \end{aligned} \quad (8.1)$$

The steps of the derivation are as follows.

$$\begin{aligned} \Delta L(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) &= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log \left( \frac{\sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(\mathbf{x} | \mathbf{s}_{c'}) Q(c')}{Q_c(\mathbf{x} | \mathbf{S}_0) \sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'')} \right) \\ &= \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \log \left( \frac{\sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_c(c' | \mathbf{x}, \mathbf{S}_0) Q(\mathbf{x} | \mathbf{s}_{c'}) Q(c')}{Q_c(\mathbf{x}, c' | \mathbf{S}_0) \sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'')} \right) \\ &\geq \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_c(c' | \mathbf{x}, \mathbf{S}_0) \log \left( \frac{Q(\mathbf{x} | \mathbf{s}_{c'}) Q(c') \sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_0(c'')}{Q_c(\mathbf{x} | \mathbf{s}_{0,c'}) Q_0(c') \sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'')} \right) \\ &= \Delta L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) \end{aligned} \quad (8.2)$$

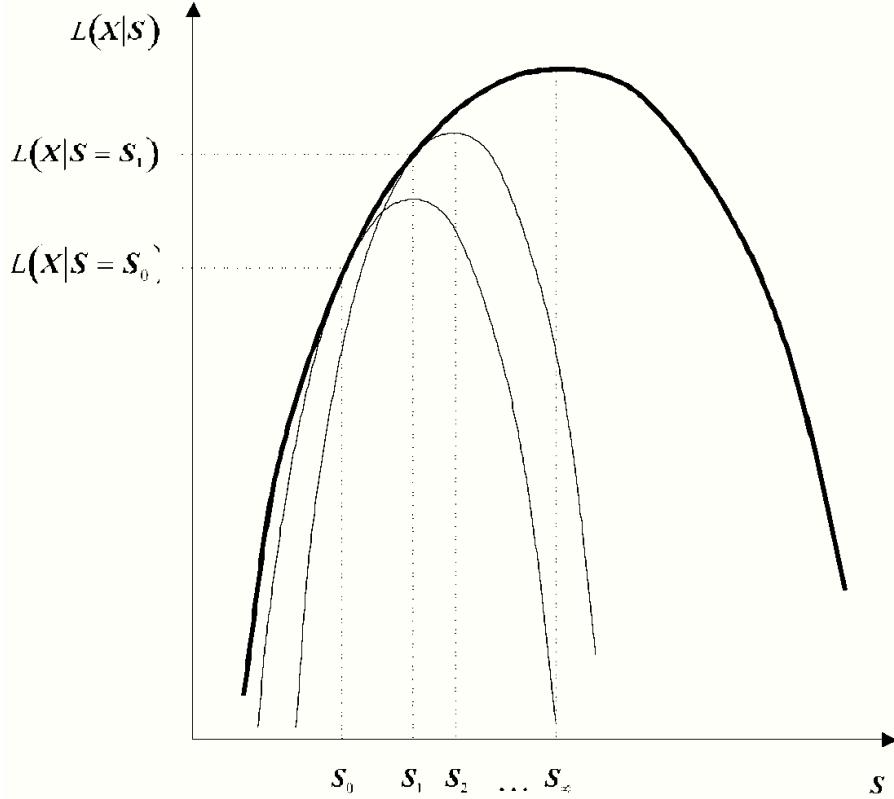


Figure 6: An example of a logarithmic likelihood function  $L(\mathbf{X}|\mathbf{S})$  is shown in bold. Assume that the starting value of the parameter vector is  $\mathbf{S} = \mathbf{S}_0$ , and the goal is to locate the position of the peak of the  $L(\mathbf{X}|\mathbf{S})$  curve. The EM method proceeds by fitting a lower bound function beneath the  $L(\mathbf{X}|\mathbf{S})$  curve, which touches it at the point  $\mathbf{S} = \mathbf{S}_0$ , as shown in the diagram, and then locating the peak of this lower bound, which is the point  $\mathbf{S} = \mathbf{S}_1$  in the diagram. Some progress has been made towards the maximum of  $L(\mathbf{X}|\mathbf{S})$ , as can be seen by inspecting the diagram. This procedure is then repeated with the new starting value  $\mathbf{S} = \mathbf{S}_1$ . After several iterations of this type the value of  $\mathbf{S}$  approaches the required location  $\mathbf{S}_\infty$  of the peak of the logarithmic likelihood function  $L(\mathbf{X}|\mathbf{S})$ , as can readily be seen intuitively in the diagram. The key trick is to ensure that it is easy to locate the peak of the lower bound curve, so that each iteration of the EM algorithm can readily be implemented.

where  $\Delta L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X})$  is defined as

$$\begin{aligned} \Delta L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) &\equiv \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \left( \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0) \log(Q(\mathbf{x}|\mathbf{s}_c)Q(c)) - \log \left( \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c') \right) \right) \\ &\quad + \text{a term that is independent of } \mathbf{S} \end{aligned} \quad (8.3)$$

The penultimate step of the derivation in Equation 8.2 was obtained by using Jensen's inequality for convex functions, and the last step was obtained by using the result  $\sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{S}_0) = 1$  and the result

$$\begin{aligned} \sum_{c=1}^N \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{S}) (\dots) &= \sum_{c'=1}^N \sum_{c=c'-\frac{n}{2}}^{c'+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{S}) (\dots) \\ &= \sum_{c'=1}^N \tilde{Q}(c'|\mathbf{x}, \mathbf{S}) (\dots) \end{aligned} \quad (8.4)$$

where the  $(\dots)$  term is symmetric under interchange of  $c$  and  $c'$ . Combining the above results yields

$$\begin{aligned} L(\mathbf{X}|\mathbf{S}) &\geq \Delta L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) + L(\mathbf{X}|\mathbf{S}_0) \\ L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) &= 0 \end{aligned} \quad (8.5)$$

$\Delta L_0(\mathbf{S}, \mathbf{S}_0; \mathbf{X}) + L(\mathbf{X}|\mathbf{S}_0)$  satisfies all the requirements of the lower bound function discussed in the caption to Figure 6, so the first step of the EM algorithm is thus given by

$$\mathbf{S}_i = \arg \max_{\mathbf{S}} \int d\mathbf{x} P(\mathbf{x}) \sum_{c=1}^N \left( \tilde{Q}(c|\mathbf{x}, \mathbf{S}_0) \log(Q(\mathbf{x}, c|\mathbf{S})) - \log \left( \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'|\mathbf{S}) \right) \right) \quad (8.6)$$


---

and subsequent steps are similarly defined. Note that each update step is not obliged to update all of the parameters in  $\mathbf{S}$ . For instance, the  $Q(c)$  might be held constant and only the  $s_c$  updated, as will be the case in the numerical simulations later. Provided that there are no local maxima of the likelihood function, then the EM update equation (see Equation 8.6) may be iterated to yield the sequence  $\mathbf{S}_0, \mathbf{S}_1, \dots, \mathbf{S}_\infty$  of successively improved estimates of the required  $\mathbf{S}(\mathbf{X})$ . In fact, it is easy to see by considering a variant of Figure 6 with a local

maximum included that it is possible, although not guaranteed, for the EM update equation to step past the local maximum without becoming trapped there.

## IX. THE GAUSSIAN PMD

A popular model to use for  $Q(\mathbf{x}|s_c)$  is a Gaussian PDF with mean  $\mathbf{m}(c)$  and covariance  $\mathbf{A}(c)$ .

$$\begin{aligned} Q(\mathbf{x}|s_c) &\longrightarrow Q(\mathbf{x}|\mathbf{m}(c), \mathbf{A}(c)) \\ &\equiv \frac{1}{\sqrt{\det 2\pi \mathbf{A}(c)}} \exp \left( -\frac{1}{2} (\mathbf{x} - \mathbf{m}(c))^T \mathbf{A}(c)^{-1} (\mathbf{x} - \mathbf{m}(c)) \right) \end{aligned} \quad (9.1)$$


---

The likelihoods  $Q_c(\mathbf{x}|\mathbf{S})$ , joint probabilities  $Q_c(\mathbf{x}, c'|\mathbf{S})$ , posterior probabilities  $Q_c(c'|\mathbf{x}, \mathbf{S})$ , and averaged posterior probabilities  $\tilde{Q}(c|\mathbf{x}, \mathbf{S})$  may be obtained from  $Q(\mathbf{x}|s_c)$  and  $Q(c)$ . The EM method may be used to solve for the maximum likelihood choice of  $\mathbf{S}$  as follows. Define the first three moments of the posterior probability weighted training data as

$$\begin{aligned} M_0(c|\mathbf{S}) &\equiv \int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}) \\ \mathbf{M}_1(c|\mathbf{S}) &\equiv \int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}) \mathbf{x} \\ \mathbf{M}_2(c|\mathbf{S}) &\equiv \int d\mathbf{x} P(\mathbf{x}) \tilde{Q}(c|\mathbf{x}, \mathbf{S}) \mathbf{x} \mathbf{x}^T \end{aligned} \quad (9.2)$$

then the EM update equation becomes

$$\begin{aligned} \hat{\tilde{Q}}(c) &= M_0(c) \\ \hat{\mathbf{m}}(c) &= \frac{\mathbf{M}_1(c)}{M_0(c)} \\ \hat{\mathbf{A}}(c) &= \frac{\mathbf{M}_2(c)}{M_0(c)} - \hat{\mathbf{m}}(c) \hat{\mathbf{m}}(c)^T \end{aligned} \quad (9.3)$$


---

where a hat denotes an updated quantity. The re-estimation equation for  $\hat{\tilde{Q}}(c)$  is the same whether not a Gaussian model is assumed, and because it has a tilde on the left hand side it must be solved iteratively. On the other hand the re-estimation equations for  $\hat{\mathbf{m}}(c)$  and  $\hat{\mathbf{A}}(c)$  are straightforward to solve. These results for a Gaussian PMD reduce to the standard results for the corresponding Gaussian mixture distribution when  $n = N$ .

## X. PROPERTIES OF PMDS

It is now an appropriate point to demonstrate a few of the more interesting properties of PMD networks. The simulations were mostly carried out on a 1-dimensional PMD with 21 nodes, an input window width of 7, a mixture window width (i.e.  $n$ ) of 7, and holding the  $Q(c)$  constant. An interesting modification to the training algorithm (not derived above) was also sometimes used. Thus the averaged posterior probabilities  $\tilde{Q}(c|\mathbf{x}, \mathbf{S})$  were modified as follows

$$\tilde{Q}(c|\mathbf{x}, \mathbf{S}) \longrightarrow (1 - 2\varepsilon) \tilde{Q}(c|\mathbf{x}, \mathbf{S}) + \varepsilon \left( \tilde{Q}(c-1|\mathbf{x}, \mathbf{S}) + \tilde{Q}(c+1|\mathbf{x}, \mathbf{S}) \right) \quad (10.1)$$


---

where  $0 \leq \varepsilon \leq 0.5$ . In effect node  $c$  leaks some of its

activity onto its neighbours. This type of modification

arises naturally in the extension of PMD theory to include time history (which is not discussed in this paper), which can readily be introduced by replacing the prior probabilities  $Q(c)$  by the averaged posterior probabilities  $\tilde{Q}(c|\mathbf{x}, \mathbf{S})$  derived from the previous time instant. This leads to an ordering of the properties of adjacent nodes in much the same way as the neighbourhood function leads to ordering in a self-organising map.

## XI. LATERAL INHIBITION

Lateral inhibition is the term used to describe the tendency for an active node in a neural network to suppress the activity of nearby nodes in the same network layer. This effect can be demonstrated in a PMD with time history when it is driven by a constant input vector. Thus the initial presentation of data  $\mathbf{x}$  causes the PMD to output an average posterior probability  $\tilde{Q}(c|\mathbf{x}, \mathbf{S}; t = 0)$  at each node, which is then fed back to become the prior probability at the next time instant. The input vector remains unchanged, and a new  $\tilde{Q}(c|\mathbf{x}, \mathbf{S}; t = 0)$  is then output, and fed back, and so on. The lateral inhibition effect becomes obvious when the pattern of numbers  $\{\tilde{Q}(c|\mathbf{x}, \mathbf{S}; t) : c = 1, 2, \dots, N\}$  is plotted out at various times  $t$ . For the purpose of this simulation it is not necessary to introduce explicit likelihood functions  $Q(\mathbf{x}|s_c)$ ; it is sufficient to use input data that is itself the pattern of numbers  $\{Q(\mathbf{x}|s_c) : c = 1, 2, \dots, N\}$ .

In Figure 7a and Figure 7b the point stimulus leads to a strong response from the stimulated node, and a weakened response from the surrounding nodes. The lateral inhibition effect arises from the form of the expression used to define the posterior probability in a PMD. Because the node responses are actually posterior probabilities, if one node has a larger than average share of probability, then its neighbours will typically have a smaller than average share. Note how the effect of node activity leakage is to damp down the responses. In Figure 7c and Figure 7d the two equal point stimuli lead to equal responses because neither leads to a single node gaining supremacy. However in Figure 7e and Figure 7f the two unequal point stimuli upset the balance so that the lateral inhibition from the node responding to the stronger stimulus suppresses the response to the smaller stimulus. The effect is highly non-linear; the ratio of the strengths of the stimuli is  $\frac{1.2}{1.1}$  whereas the ratio of the responses is much larger. The effect of node activity leakage is to damp out this behaviour somewhat.

## XII. SYMMETRY BREAKING AND COMPLETENESS

Another interesting effect that PMDs exhibit is symmetry breaking, such that although the statistical properties of the training data are translation invariant (i.e. each node is treated equivalently), the stable solutions

for the parameter vectors  $s_c(\mathbf{X})$  break this symmetry (i.e. the nodes do different jobs). Furthermore, the symmetry is broken in a very careful way, as will become clear below.

The results of Figure 8 show that a Gaussian PMD is very sensitive to the presence of activity leakage from a node to its neighbours. Only 4 iterations of the EM algorithm are shown which is insufficient to lead to convergence, but nevertheless some interesting observations can be made. The effect of leakage is to order the Gaussian means in such a way that they change progressively from node to node along the PMD; this ordering is conspicuously absent in the PMD without leakage. Furthermore, the progression fluctuates regularly (i.e. breaks symmetry) as indicated by the dotted line that connects the means. Note that within each mixture window (which has width 7) there is approximately one cycle of this fluctuation. In fact each mixture window possesses a complete repertoire of Gaussians with which to process the data that it sees in its input window. This will be referred to as the “completeness” property.

When the EM algorithm is run for 10 iterations and the variances are updated as well as the means, then the simulation with leakage leads to the result shown in Figure 9. The dotted line shows that the fluctuations of the means have settled down to an almost perfect sinusoidal form with approximately one cycle per mixture window. The  $\frac{1}{4}$  and  $\frac{1}{2}$  half standard deviation curves are shown to indicate the thickness of the “Gaussian tubes”. These tubes are thinner where they are clustered closer together at the top and bottom of the figure (the sinusoidal training data spends longer in these regions because of its turning points).

In Figure 10 the result of a 2-dimensional simulation is shown for comparison with Figure 9. The fluctuations appear again as in the 1-dimensional simulation. The completeness property may be observed by inspecting the part of the montage that is contained in the superimposed square. The size of this square is the same as a mixture distribution window, and it contains a complete repertoire of Gaussians (i.e. exhibits completeness). This is true wherever the square is placed on the montage.

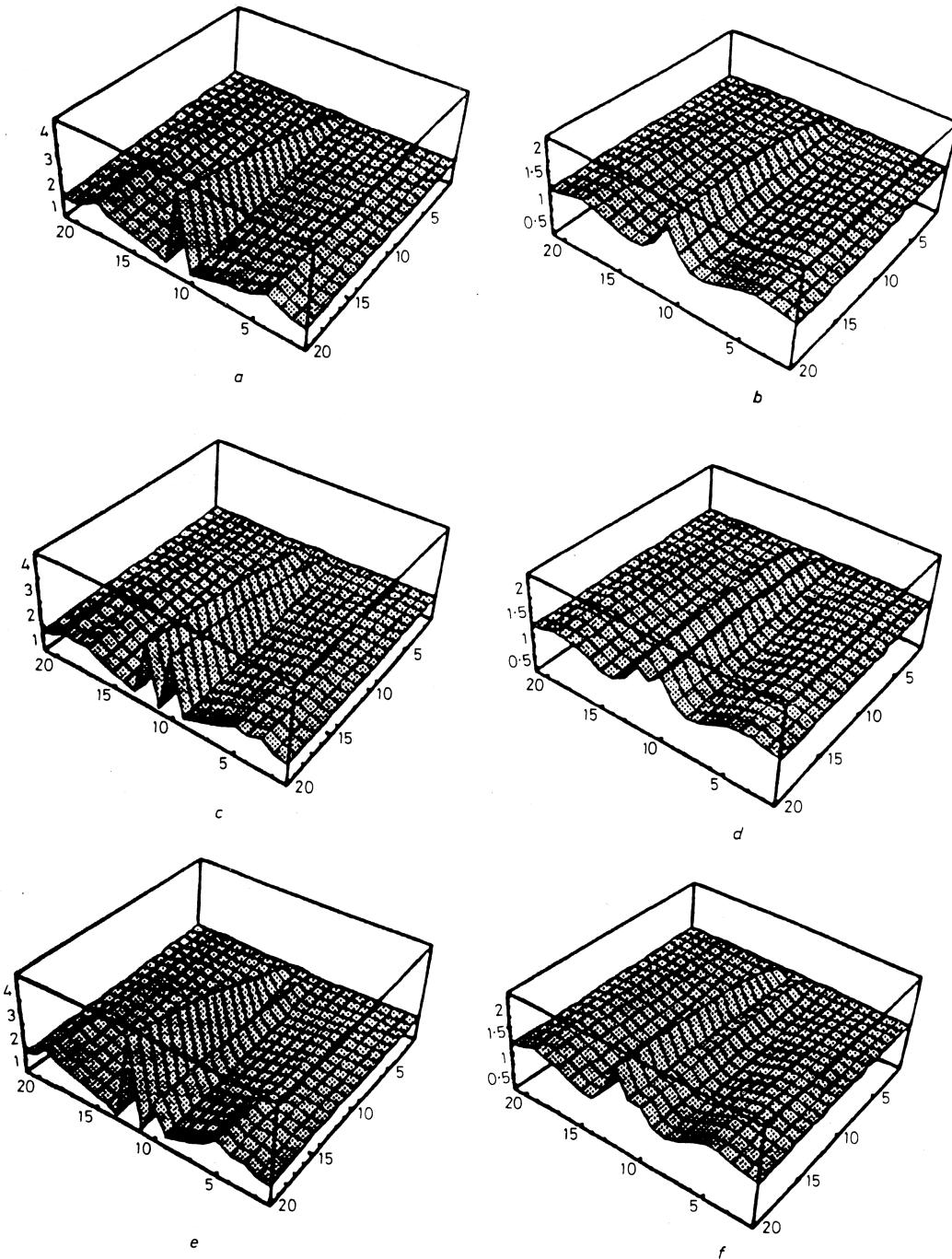


Figure 7: In this simulation the PMD is a 1-dimensional array of 21 nodes with a mixture window width of 7. Each row of the figure uses a different pattern of likelihoods: (a) and (b) have a point stimulus of 1.2 embedded in a constant background stimulus of 1, (c) and (d) have two equal point stimuli of strength 1.15 separated by one node embedded in a constant background stimulus of 1, and (e) and (f) have two unequal point stimuli of strength 1.1 and 1.2 separated by one node embedded in a constant background stimulus of 1. The left hand column is a standard PMD, whereas the right hand column has node activity leakage (i.e. it is topographic) such that 10% of a node's activity leaks onto each of its immediate left and right hand neighbours. In every case the prior pattern of node responses was set to to a constant 1, and 20 iterations of the PMD update equation were made. The resulting patterns of activity are displayed (with the time axis pointing out of the page).

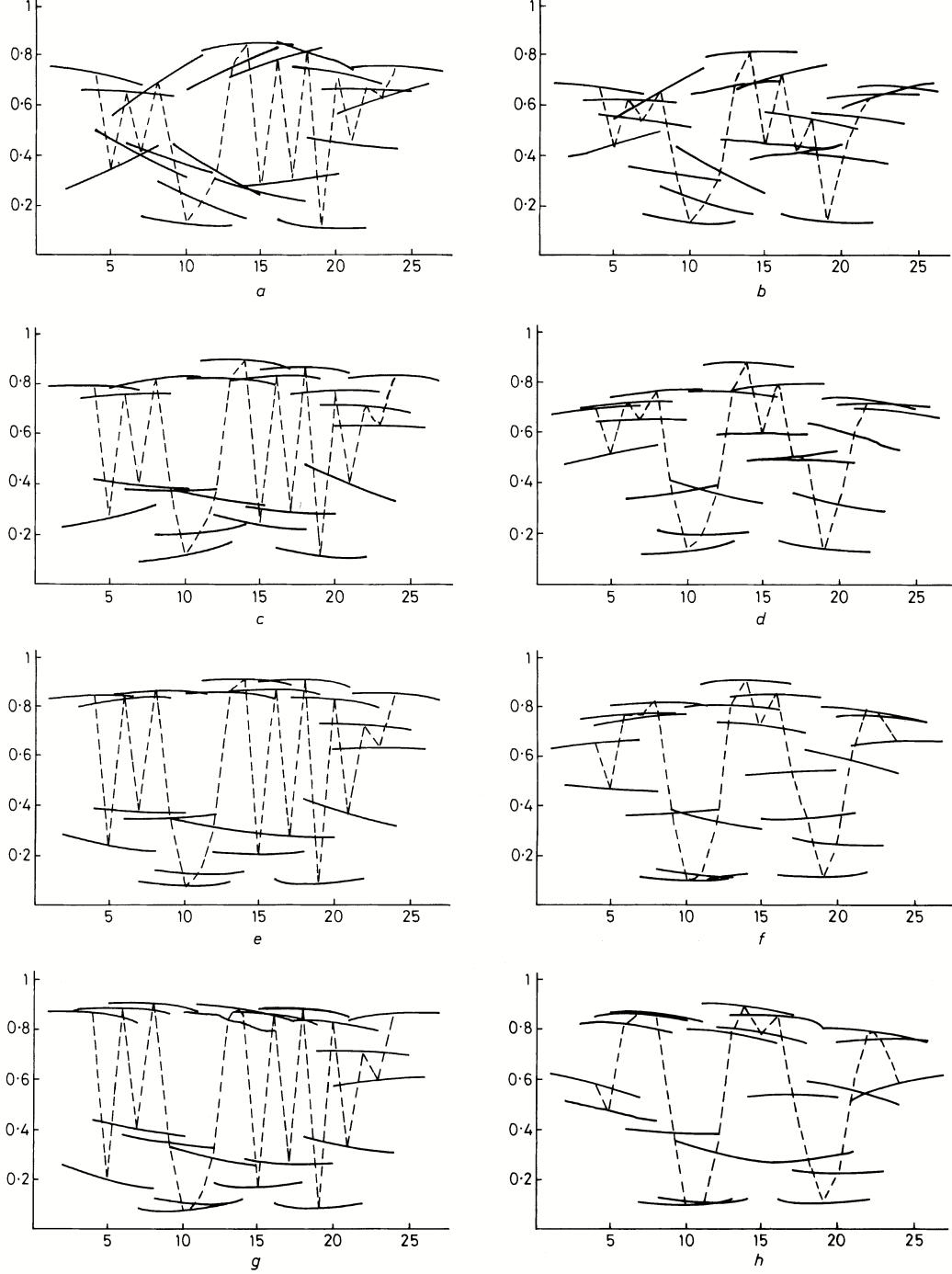


Figure 8: In this simulation the PMD is a 1-dimensional array of 21 nodes with a rruxture window width of 7 and an input window width of 7. The likelihood functions are Gaussian, each with a constant variance of 0.1, and a trainable mean whose components are initialised to be random numbers uniformly drawn from the interval [0, 1]. Each training vector was a 27 dimensional (=21+3+3 to prevent the input window from failing off the ends of the input vector) vector comprising a sampled sinusoid advancing at 0.1 radians per sample. The starting phase of the sinusoid was randomly chosen for each training vector, and the height of sinusoid was rescaled to the interval [0,1] (i.e. add 1 and divide by 2). Each successive row of the figure shows the Gaussian means after one more iteration of the EM algorithm; 4 iterations are shown. The left hand column is a standard PMD, whereas the right hand column has node activity leakage (i.e. it is topographic) such that 10% of a nodes activity leaks onto each of its immediate left and right hand neighbours. The dotted lines joint the centre of each Gaussian mean to its neighbours' means.

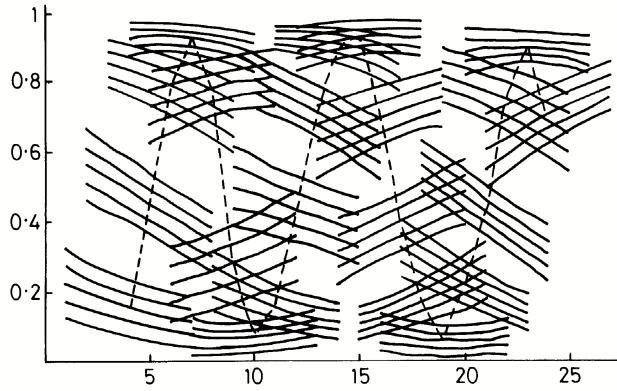


Figure 9: This is the result of 10 iterations of the EM algorithm. The parameters are the same as for the simulation with leakage in Figure 8. Also the variances were updated as well as the means. Each mean and variance is shown as 5 curves: mean,  $\text{mean} \pm 0.25\sqrt{\text{variance}}$ ,  $\text{mean} \pm 0.5\sqrt{\text{variance}}$ .

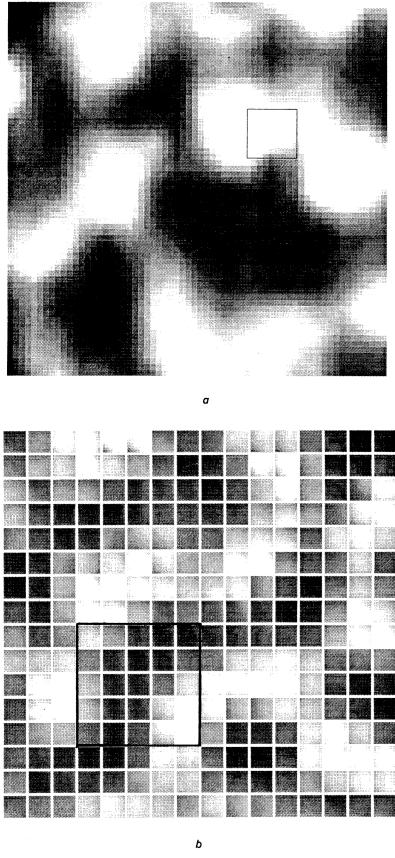


Figure 10: The result of a 2-dimensional PMD simulation. The PMD was a  $16 \times 16$  array of nodes, each input window was  $9 \times 9$ , and each mixture window was  $5 \times 5$ . (a) Example of a training image, where the superimposed square indicates the size of an input window. (b) A montage formed from the Gaussian means after training where the superimposed square indicates the size of a mixture window.

### XIII. CONCLUSIONS

Mixture distributions are a natural candidate for approximating the probability density of data, but they are not appropriate for high-dimensional applications such as image processing. The partitioned mixture distribution (PMD) networks introduced in this paper solve the problem of high-dimensional data, also their theoretical properties naturally extend those of standard mixture distributions, and finally they exhibit behaviours that have not been observed in standard mixture distributions. Two interesting behaviours of PMDs have been demonstrated. Lateral inhibition arises such that an active node tends to suppress activity in its neighbouring nodes. Furthermore, the job of computing the probability density of the data is divided up between the nodes of a PMD in such a way that each mixture window contains a complete repertoire of the machinery that is needed to compute a mixture distribution. So, although the PMD does break symmetry at a node by node level, it nevertheless does not break symmetry at the level of complete mixture distributions. In this paper only a few properties of PMDs have been explored. However, these properties emerge naturally, and are very similar to properties that are observed in experiments on the mammalian low-level visual cortex. It is encouraging that such results can be derived from Bayesian principles. Finally, PMD networks have a structure that is amenable to hardware implementation. This opens up the possibility of constructing a fast low-level vision engine based entirely on rigorous Bayesian principles.

- [1] R T Cox, *Probability, frequency and reasonable expectation*, American Journal of Physics **14** (1946), no. 1, 1–13.
- [2] S P Luttrell, *A trainable texture anomaly detector using the adaptive cluster expansion (ACE) method*, Memorandum 4437, Royal Signals and Radar Establishment, Malvern, 1990.
- [3] \_\_\_\_\_, *A hierarchical network for clutter and texture modelling*, Proceedings of SPIE conference on adaptive signal processing (San Diego), SPIE, 1991, pp. 518–528.
- [4] \_\_\_\_\_, *Partitioned mixture distributions: an introduction*, Memorandum 4671, Defence Research Agency, 1992.



# The Cluster Expansion: A Hierarchical Density Model \*

Stephen P Luttrell<sup>†</sup>

*Defence Research Agency, St Andrews Rd, Malvern, Worcestershire, WR14 3PS, United Kingdom*

Density modelling in high-dimensional spaces is a difficult problem. In this paper a new model, called the cluster expansion, is proposed and discussed. The cluster expansion scales well to high-dimensional spaces, and it allows the integrals over model parameters that arise in Bayesian predictive distributions to be evaluated explicitly.

## I. INTRODUCTION

The basic idea behind the cluster expansion is as follows. Density models in subspaces (or clusters of pixels) of a high-dimensional input space are first built, and these are then linked together to form clusters-of-clusters, which are further linked, etc. This type of hierarchical approach is computationally very efficient.

The purpose of this paper is to present a Bayesian derivation of the cluster expansion model. This supplements the rather non-Bayesian discussions presented in [1, 3, 4].

## II. NOTATION

The following notation is used in this paper.  $M$  = model,  $D$  = training set of data,  $D'$  = training set of data plus one extra sample (i.e. the test sample),  $N$  = number of samples in the training set,  $\mathbf{x}$  = input vector,  $\mathbf{s}$  = parameter vector of the model,  $\mathbf{y}(\dots)$  = transformation function (or mapping),  $\delta(\dots)$  = Dirac delta function,  $L$  = number of layers in the model,  $l$  = layer index,  $c$  = cluster index,  $\Gamma(\dots)$  = Gamma function,  $\nu$  = order parameter,  $m$  = number of counts in a histogram

\*Typeset in LATEX on May 21, 2019.

This appeared in Maximum Entropy and Bayesian Methods, Kluwer, J. Skilling and S. Sibisi (eds.), 269-278, 1996. © British Crown Copyright 1994 / DRA. Published with the permission of

bin,  $n$  = number of histogram bins per dimension,  $Y$  = whole transformation function,  $[dY]$  = integration measure over transformation functions.

## III. THE CLUSTER EXPANSION MODEL

The standard Bayesian expansion for the predictive distribution  $\Pr(\mathbf{x}|D, M)$  is

$$\Pr(\mathbf{x}|D, M) = \int d\mathbf{s} \Pr(\mathbf{x}|\mathbf{s}, M) \Pr(\mathbf{s}|D, M) \quad (3.1)$$

where the integral over  $\mathbf{s}$  is usually difficult to evaluate. The main purpose of this paper is to present a model that has the following two properties: (1)  $M$  scales sensibly to high dimensional  $\mathbf{x}$ , (2)  $M$  and  $\Pr(\mathbf{x}|D, M)$  factorise into a product of loosely coupled pieces. These two goals must be achieved *without* simply making an independence assumption. This type of  $M$  is called a “cluster expansion”, and it was briefly discussed in a rather non-Bayesian way in [1].

The simplest non-trivial example of a cluster expansion is

the Controller of Her Britannic Majesty's Stationery Office.

<sup>†</sup>Electronic address: luttrell@signal.dra.hmg.gb

---


$$\Pr(\mathbf{x}|\mathbf{s}, M) = \frac{\Pr(\mathbf{x}_1|\mathbf{s}_1, M_1)}{\Pr(\mathbf{y}_1(\mathbf{x}_1)|\mathbf{s}_1, M_1)} \frac{\Pr(\mathbf{x}_2|\mathbf{s}_2, M_2)}{\Pr(\mathbf{y}_2(\mathbf{x}_2)|\mathbf{s}_2, M_2)} \Pr(\mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2)|\mathbf{s}_{12}, M_{12}) \quad (3.2)$$


---

where  $M$  is broken into 3 pieces as  $M = M_1 \times M_2 \times M_{12}$ , and the parameter vector is appropriately partitioned as  $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12})$ . The input vector is partitioned into two non-overlapping pieces as  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ , where  $M_1$  models the density of  $\mathbf{x}_1$  and  $M_2$  models the density of  $\mathbf{x}_2$ .

The input vector is then transformed to produce another pair of vectors  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$  whose joint density is modelled by  $M_{12}$ . The denominators are normalisation factors that may be obtained from their respective numerators as follows

---


$$\Pr(\mathbf{y}_k(\mathbf{x}_k)|\mathbf{s}_k, M_k) = \int d\mathbf{x}'_k \delta(\mathbf{y}_k(\mathbf{x}'_k) - \mathbf{y}_k(\mathbf{x}_k)) \Pr(\mathbf{x}'_k|\mathbf{s}_k, M_k) \quad (3.3)$$

where the delta function constrains the integral over input space to include only those points that map to

$\mathbf{y}_k(\mathbf{x}_k)$ . The normalisation of the cluster expansion can then be checked by integrating over all  $\mathbf{x}$  using

$$\begin{aligned} \int d\mathbf{x} (\dots) &= \int d\mathbf{x}_1 d\mathbf{x}_2 (\dots) \\ &= \int d\mathbf{y}_1 d\mathbf{y}_2 \int d\mathbf{x}_1 d\mathbf{x}_2 \delta(\mathbf{y}_1 - \mathbf{y}_1(\mathbf{x}_1)) \delta(\mathbf{y}_2 - \mathbf{y}_2(\mathbf{x}_2)) (\dots) \end{aligned} \quad (3.4)$$

which first integrates over the the  $\mathbf{x}_k$  that map to a particular  $\mathbf{y}_k$ , and then integrates over the  $\mathbf{y}_k$ .

The cluster expansion model may also be derived as the solution to the following maximum entropy problem [1]: (1) Assume that  $\mathbf{x}_1$ ,  $\mathbf{x}_2$ ,  $\mathbf{y}_1(\mathbf{x}_1)$ , and  $\mathbf{y}_2(\mathbf{x}_2)$  are defined as above, (2) Supply the three marginal PDFs  $\Pr(\mathbf{x}_1)$ ,  $\Pr(\mathbf{x}_2)$ , and  $\Pr(\mathbf{y}_1, \mathbf{y}_2)$  as constraints on the maximum entropy  $\Pr(\mathbf{x})$ , (3) Seek the maximum entropy  $\Pr(\mathbf{x})$  that satisfies these constraints. In the special case where  $\Pr(\mathbf{y}_1, \mathbf{y}_2)$  is *not* supplied as a constraint, the maximum entropy solution is  $\Pr(\mathbf{x}_1)\Pr(\mathbf{x}_2)$ , as expected. When  $\Pr(\mathbf{y}_1, \mathbf{y}_2)$  is supplied as a constraint, this solution acquires an extra factor  $\frac{\Pr(\mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2))}{\Pr(\mathbf{y}_1(\mathbf{x}_1))\Pr(\mathbf{y}_2(\mathbf{x}_2))}$  to account for any correlations between  $\mathbf{y}_1$  and  $\mathbf{y}_2$  that are found in  $\Pr(\mathbf{y}_1, \mathbf{y}_2)$ . Although this maximum entropy style of argument was the way in which the cluster expansion was originally introduced in [1], in this paper the main justification for the cluster expansion is that it allows Bayesian computations (e.g. integrations over model parameters) to be performed readily.

The cluster expansion model may be visualised as shown in Figure 1, which shows the model structure for a 4-dimensional input vector  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ , where  $\mathbf{x}_1 = (x_{11}, x_{12})$  and  $\mathbf{x}_2 = (x_{21}, x_{22})$ . The clusters are each 2-dimensional subspaces of the full 4-dimensional input space, and Figure 1 shows each of these subspaces discretised into bins. The bins highlighted in black in Figure 1 correspond to a single representative input vector  $\mathbf{x}$ . The transformation  $\mathbf{y}_k(\mathbf{x}_k)$  maps  $\mathbf{x}_k$  to  $\mathbf{y}_k$ , and Figure 1 shows examples of how a subset of

bins in each input subspace maps to the same bin in the corresponding part of output space. Although these subsets of bins are shown in Figure 1 as consisting of contiguous bins, this need not necessarily be the case.  $\Pr(\mathbf{x}_k|\mathbf{s}_k, M_k)$  is the likelihood that  $M_k$  computes for input vector  $\mathbf{x}_k$  using parameters  $\mathbf{s}_k$ .  $\Pr(\mathbf{y}_k(\mathbf{x}_k)|\mathbf{s}_k, M_k)$  is the sum of the likelihoods  $\Pr(\mathbf{x}_k|\mathbf{s}_k, M_k)$  over all those  $\mathbf{x}_k$  that map to the same  $\mathbf{y}_k(\mathbf{x}_k)$ , as shown in Figure 1.  $\Pr(\mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2)|\mathbf{s}_{12}, M_{12})$  is the likelihood that  $M_{12}$  computes for output vector  $(\mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2))$  using parameters  $\mathbf{s}_{12}$ . Thus  $M_1$  has responsibility for modelling the bottom left hand part of Figure 1,  $M_2$  the bottom right hand part, and  $M_{12}$  the top part.

In practice, the cluster expansion model in Figure 1 may be applied to a situation where the data arrives at a pair of sensors (as  $\mathbf{x}_1$  and  $\mathbf{x}_2$ ). One Bayesian is located at each of the two sensors, and each constructs a likelihood ( $\Pr(\mathbf{x}_1|\mathbf{s}_1, M_1)$  and  $\Pr(\mathbf{x}_2|\mathbf{s}_2, M_2)$ ) for her own data. However, they decide to pass on the responsibility for modelling the mutual correlations between their data to a third Bayesian, who receives a transformed version the data  $(\mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2))$  and constructs a likelihood  $\Pr(\mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2)|\mathbf{s}_{12}, M_{12})$  for this data. The full cluster expansion in Equation 3.2 then combines these three likelihoods to form a consistent joint probability.

The ratio  $\frac{\Pr(\mathbf{x}_k|\mathbf{s}_k, M_k)}{\Pr(\mathbf{y}_k(\mathbf{x}_k)|\mathbf{s}_k, M_k)}$  that appears in the cluster expansion is a normalised likelihood that sums to 1 over the subset of bins that map to the same  $\mathbf{y}_k(\mathbf{x}_k)$ , as seen in Figure 1. If Bayes' theorem is used in the form

$$\Pr(\mathbf{z}|\mathbf{y}(\mathbf{x})) = \frac{\Pr(\mathbf{y}(\mathbf{x})|\mathbf{z}) \Pr(\mathbf{z})}{\Pr(\mathbf{y}(\mathbf{x}))} = \delta(\mathbf{y}(\mathbf{x}) - \mathbf{y}(\mathbf{z})) \frac{\Pr(\mathbf{z})}{\Pr(\mathbf{y}(\mathbf{x}))} \quad (3.5)$$

then

$$\Pr(\mathbf{z}_k|\mathbf{y}_k(\mathbf{x}_k), \mathbf{s}_k, M_k) = \delta(\mathbf{y}_k(\mathbf{x}_k) - \mathbf{y}_k(\mathbf{z}_k)) \frac{\Pr(\mathbf{z}_k|\mathbf{s}_k, M_k)}{\Pr(\mathbf{y}_k(\mathbf{x}_k)|\mathbf{s}_k, M_k)} \quad (3.6)$$

So  $\frac{\Pr(\mathbf{x}_k|\mathbf{s}_k, M_k)}{\Pr(\mathbf{y}_k(\mathbf{x}_k)|\mathbf{s}_k, M_k)}$  corresponds to  $\Pr(\mathbf{z}_k|\mathbf{y}_k(\mathbf{x}_k), \mathbf{s}_k, M_k)$ , which is non-zero only for  $\mathbf{y}_k(\mathbf{z}_k) = \mathbf{y}_k(\mathbf{x}_k)$ . This observation leads to the “top-down” interpretation of the cluster expansion: (1) Top level:  $\Pr(\mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2)|\mathbf{s}_{12}, M_{12})$  models the joint PDF in  $\mathbf{y}$ -space, (2) Bottom level: Each of the ratios  $\frac{\Pr(\mathbf{x}_k|\mathbf{s}_k, M_k)}{\Pr(\mathbf{y}_k(\mathbf{x}_k)|\mathbf{s}_k, M_k)}$  is a conditional PDF that generates the possible  $\mathbf{x}_k$  that correspond to  $\mathbf{y}_k(\mathbf{x}_k)$ .

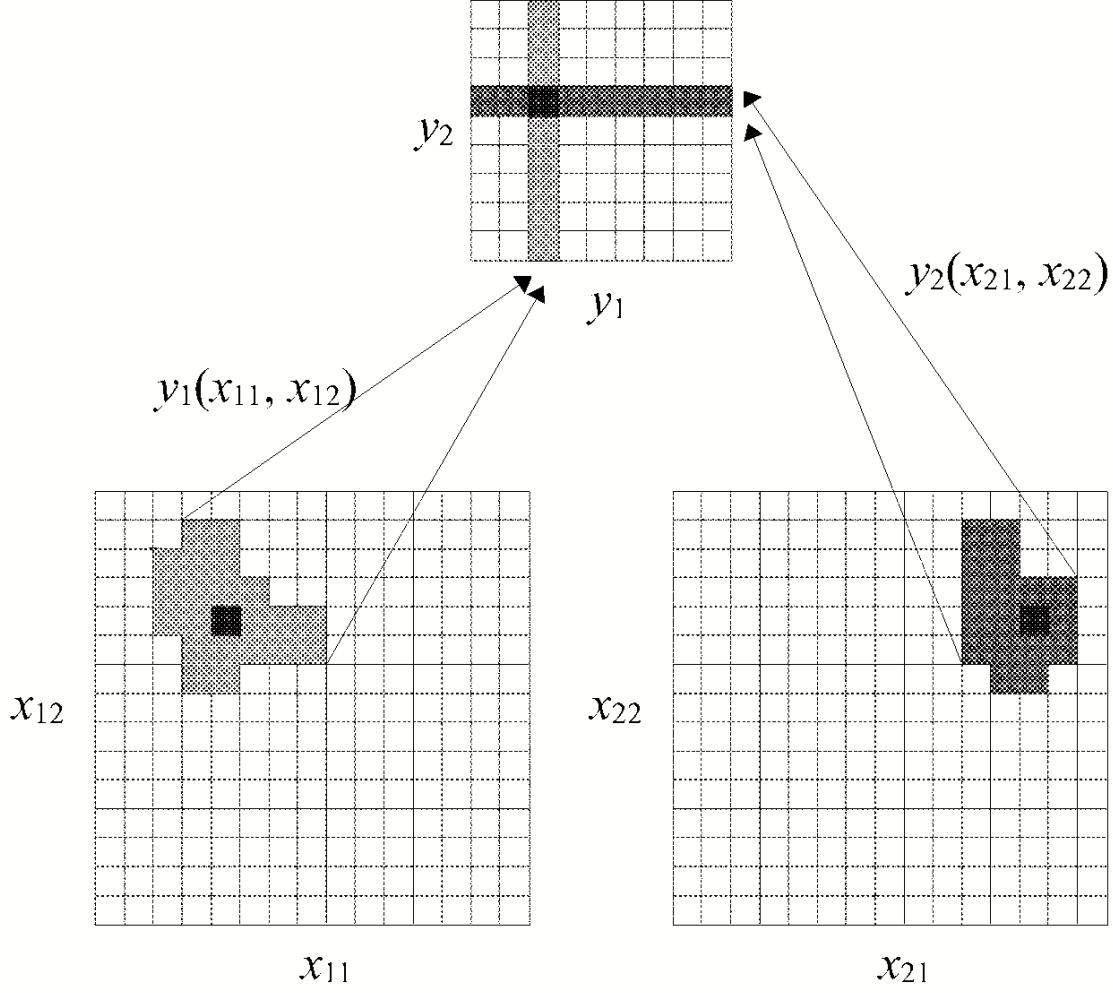


Figure 1: The cluster expansion for a 4-dimensional input vector.

Using Equation 3.1, the cluster expansion yields the following predictive distribution after integration over the parameters  $\mathbf{s}_1$ ,  $\mathbf{s}_2$  and  $\mathbf{s}_{12}$

$$\begin{aligned} \Pr(\mathbf{x}|D, M) &= \int d\mathbf{s}_1 d\mathbf{s}_2 d\mathbf{s}_{12} \Pr(\mathbf{x}|\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12}, M_1, M_2, M_{12}) \\ &\quad \times \frac{\Pr(D|\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12}, M_1, M_2, M_{12}) \Pr(\mathbf{s}_1|M_1) \Pr(\mathbf{s}_2|M_2) \Pr(\mathbf{s}_{12}|M_{12})}{\Pr(D|M)} \end{aligned} \quad (3.7)$$

where the second line is  $\Pr(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12}|D, M)$ , and  $\Pr(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12}|M)$  is assumed to be a product of three independent pieces. Assuming  $\mathbf{x}$  and  $D$  are independent, Equation 3.7 can be rewritten as

$$\Pr(\mathbf{x}|D, M) = \frac{\int d\mathbf{s}_1 d\mathbf{s}_2 d\mathbf{s}_{12} \Pr(\mathbf{s}_1|M_1) \Pr(\mathbf{s}_2|M_2) \Pr(\mathbf{s}_{12}|M_{12}) \Pr(D'|\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12}, M)}{\int d\mathbf{s}_1 d\mathbf{s}_2 d\mathbf{s}_{12} \Pr(\mathbf{s}_1|M_1) \Pr(\mathbf{s}_2|M_2) \Pr(\mathbf{s}_{12}|M_{12}) \Pr(D|\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_{12}, M)} \quad (3.8)$$

where  $D'$  is the training data  $D$  plus one extra sample  $\mathbf{x}$ .

Because the numerator and denominator have the same structure, only the denominator needs to be evaluated in

detail. For an  $N$  sample training set the denominator reduces to

$$\Pr(D|M) = \int d\mathbf{s}_1 \Pr(\mathbf{s}_1|M_1) \prod_{i=1}^N \frac{\Pr(\mathbf{x}_{i,1}|\mathbf{s}_1, M_1)}{\Pr(\mathbf{y}_1(\mathbf{x}_{i,1})|\mathbf{s}_1, M_1)} \int d\mathbf{s}_2 \Pr(\mathbf{s}_2|M_2) \prod_{i=1}^N \frac{\Pr(\mathbf{x}_{i,2}|\mathbf{s}_2, M_2)}{\Pr(\mathbf{y}_2(\mathbf{x}_{i,2})|\mathbf{s}_2, M_2)} \\ \times \int d\mathbf{s}_{12} \Pr(\mathbf{s}_{12}|M_{12}) \prod_{i=1}^N \Pr(\mathbf{y}_1(\mathbf{x}_{i,1}), \mathbf{y}_2(\mathbf{x}_{i,2})|\mathbf{s}_{12}, M_{12}) \quad (3.9)$$

Note that Equation 3.9 is a product of three separate integrals, which makes it relatively simple to evaluate.

#### IV. A DETAILED MODEL

During the derivation of the predictive distribution (both here and in Appendix A) a somewhat cavalier notation will be used in order to avoid overloading the equations with indices. However, the final result will be stated with all indices correctly included (as defined in Appendix B).

There are three separate pieces of information that are needed to specify a model so that a predictive distribution can be computed. (1) The model  $M$ : Assume that all of the variables have been discretised into bins (see Figure 1), and assume that each part ( $M_1$ ,  $M_2$ , and  $M_{12}$  in Figure 1) of the overall model is parameterised by a separate set of bin occupation probabilities  $s_1, s_2, \dots, s_n$ . (2) The priors  $\Pr(\mathbf{s}_1|M_1)$ ,  $\Pr(\mathbf{s}_2|M_2)$  and  $\Pr(\mathbf{s}_{12}|M_{12})$ : Assume that the  $\mathbf{s}_i$  each have a Dirichlet prior distribution (see Appendix A). (3) The training set  $D$ : Assume that the training data is supplied as a histogram of observed bin occupancies  $m_1, m_2, \dots, m_n$ . The probability of producing these observed occupancies from the underlying occupation probabilities is  $s_1^{m_1} s_2^{m_2} \cdots s_n^{m_n}$  (times a combinatorial factor that cancels out when computing the predictive distribution, see e.g. Equation 3.8).

With these assumptions Equation 3.9 reduces to a product of two types of factor

$$\text{Type 1} = \int d\mathbf{s} \delta\left(\sum_{i=1}^n s_i - 1\right) \prod_{i=1}^n s_i^{m_i + \nu_i - 1} \quad (4.1)$$

$$\text{Type 2} = \int d\mathbf{s} \delta\left(\sum_{i=1}^n s_i - 1\right) \prod_{i=1}^n \frac{s_i^{m_i + \nu_i - 1}}{\left(\sum_{j:y(j)=y(i)} s_j\right)^{m_i}}$$

where  $m_i$ ,  $\nu_i$  and  $s_i$  are physically different variables in the type 1 and type 2 expressions. The numerator in Equation 3.8 is effectively the same as the denominator with the addition of one extra training sample. If this extra sample falls into bin  $k$ , then the numerator can be obtained by modifying the factors in Equation 4.1 as follows: include an extra  $s_k$  (type 1), or include an extra  $\frac{s_k}{\sum_{j:y(j)=y(k)} s_j}$  (type 2). Both of these cases may readily be implemented by appropriately modifying the powers inside the product term in Equation 4.1.

When the type 1 and type 2 terms are evaluated, and the ratio of the numerator and corresponding denominator contributions in Equation 3.8 is taken, the following simple results emerge (see Appendix A).  $\hat{A}$

$$\begin{aligned} \text{Type 1} &= \frac{m_k + \nu_k}{\sum_{i=1}^n (m_i + \nu_i)} \\ \text{Type 2} &= \frac{m_k + \nu_k}{\sum_{i:y(i)=y(k)}^n (m_i + \nu_i)} \end{aligned} \quad (4.2)$$

These results may be gathered together to yield

$$\Pr(i, j, k, l|D, M) = \frac{m_{1;i,j}^1 + \nu_{1;i,j}^1}{m_{1;y_1(i,j)}^2 + \nu_{1;y_1(i,j)}^2} \frac{m_{2;k,l}^1 + \nu_{2;k,l}^1}{m_{2;y_2(k,l)}^2 + \nu_{2;y_2(k,l)}^2} \frac{m_{1,2;y_1(i,j),y_2(k,l)}^2 + \nu_{1;y_1(i,j)}^2 \nu_{2;y_2(k,l)}^2}{\sum_{y_1, y_2} (m_{1,2;y_1,y_2}^2 + \nu_{1;y_1}^2 \nu_{2;y_2}^2)} \quad (4.3)$$

where detailed indices have been used to ensure that this result is unambiguously expressed (see Appendix B).

#### V. INTEGRATION OVER THE LAYER TO LAYER TRANSFORMATIONS

The predictive distribution in Equation 3.1 (and Equation 3.7) assumes that the layer-to-layer transforma-

tions are held fixed, i.e. they are not variable parameters of the model. In effect, the predictive distribution  $\Pr(\mathbf{x}|D, M)$  thus far calculated should strictly be written as  $\Pr(\mathbf{x}|D, Y, M)$ , where the  $Y$  denotes the set of layer-to-layer transformations that is used. The full predictive distribution may now be written as

$$\Pr(\mathbf{x}|D, M) = \int [dY] \Pr(\mathbf{x}|D, Y, M) \Pr(Y|D, M) \quad (5.1)$$


---

$$\Pr(\mathbf{x}|D, M) = \int d\mathbf{y}_1 d\mathbf{y}_2 \int [dY] \delta(\mathbf{y}_1 - \mathbf{y}_1(\mathbf{x}_1)) \delta(\mathbf{y}_2 - \mathbf{y}_2(\mathbf{x}_2)) \Pr(\mathbf{x}|D, Y, M) \Pr(Y|D, M) \quad (5.2)$$


---

If all points in output space are a priori equivalent, then the inner integral over  $Y$  yields a result that is independent of  $\mathbf{y}_1$  and  $\mathbf{y}_2$ . The outer integral (over  $\mathbf{y}_1$  and  $\mathbf{y}_2$ ) merely introduces a constant overall factor, so henceforth  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$  will be held constant in the  $Y$  integration in Equation 5.1. In other words, it does not matter to which particular  $\mathbf{y}_1$  and  $\mathbf{y}_2$  one decides to map  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , respectively.

There are two cases of Equation 5.2 where the  $Y$  integral may readily be simplified further: (1) The  $\Pr(\mathbf{x}|D, Y, M)$  factor dominates, or (2) The  $\Pr(Y|D, M)$  factor dominates (as is generally the case). These two cases will be considered below.

### A. Case 1

Equation 3.8 (which is now to be read as computing  $\Pr(\mathbf{x}|D, Y, M)$ ) has all of its  $\mathbf{x}$  dependence in the numerator in the  $\Pr(D'|s_1, s_2, s_{12}, M)$  factor. The transformations appear in three separate places: (1)  $\mathbf{y}_1(\mathbf{x}_1)$  appears in  $\frac{\Pr(\mathbf{x}_1|s_1, M_1)}{\Pr(\mathbf{y}_1(\mathbf{x}_1)|s_1, M_1)}$ .  $\mathbf{y}_1(\mathbf{x}_1)$  is the point in output space to which  $\mathbf{x}_1$  maps. The output PDF  $\Pr(\mathbf{y}_1(\mathbf{x}_1)|s_1, M_1)$  is the integral of the input PDF  $\Pr(\mathbf{x}_1|s_1, M_1)$  over all those input vectors that map to  $\mathbf{y}_1(\mathbf{x}_1)$ . In the  $Y$  integration this term is large for those  $Y$  that map only a small number of input vectors to the given  $\mathbf{y}_1(\mathbf{x}_1)$ , because then the PDF  $\Pr(\mathbf{y}_1(\mathbf{x}_1)|s_1, M_1)$  is small. (2) There is a similar term for  $\mathbf{y}_2(\mathbf{x}_2)$ . (3)  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$  both appear in  $\Pr(\mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2)|s_{12}, M_{12})$ . Typically (see the example below), the output PDF  $\Pr(\mathbf{y}_1(\mathbf{x}_1), \mathbf{y}_2(\mathbf{x}_2)|s_{12}, M_{12})$  is large for those  $Y$  that map a large number of members of the training set to the given  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$ , which are assumed to be held constant as discussed after Equation 5.25.2

These observations lead to two opposing effects. (1) and (2) above require that *few* other input vectors map to  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$ , whereas (3) above typically requires that *many* members of the training set map to  $\mathbf{y}_1(\mathbf{x}_1)$

where  $\Pr(\mathbf{x}|D, Y, M)$  is computed by the right hand side of Equation 3.1. It is important to determine which particular transformations dominate in Equation 5.1 so that the integral can be approximated. No attempt will be made to analyse Equation 5.1 in detail, rather, an attempt will be made to extract some rules of thumb which will assist in its interpretation.

A simplification can be made by rewriting Equation 5.1 as follows

and  $\mathbf{y}_2(\mathbf{x}_2)$ .

Suppose that the training data is drawn from a distribution in which  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are related to each other as  $\mathbf{x}_1 = \mathbf{x}_1(\mathbf{x}_2) + \epsilon_1(\mathbf{x}_2)$  and  $\mathbf{x}_2 = \mathbf{x}_2(\mathbf{x}_1) + \epsilon_2(\mathbf{x}_1)$ , where  $\epsilon_1(\mathbf{x}_2)$  and  $\epsilon_2(\mathbf{x}_1)$  are localised noise processes. If  $\mathbf{x}_1$  is known then  $\mathbf{x}_2$  is also known approximately, and vice versa. The dominant contributions to the  $Y$  integral then come from transformations that have the general form shown in Figure 1. The set of input vectors that map to  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$  typically occupy patches of input space in the neighbourhoods of  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , respectively, whereas all other regions of input space typically map to other output values (i.e. *not* to  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$ ). In effect,  $Y$  is unconstrained, apart from in the vicinity of  $\mathbf{x}_1$  and  $\mathbf{x}_2$  where  $Y$  is constrained to map to the given  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$ . Actually, it is a little more complicated than this, because the border between the constrained and unconstrained regions is somewhat blurred, and it depends on how much training data there is, but the general behaviour is typically as described.

### B. Case 2

Using Bayes' theorem,  $\Pr(Y|D, M)$  may be written as

$$\Pr(Y|D, M) = \frac{\Pr(D|Y, M) \Pr(Y|M)}{\Pr(D|M)} \quad (5.3)$$

The  $Y$  dependence is contained entirely in the numerator, and it will be assumed to be dominated by the training set likelihood term  $\Pr(D|Y, M)$ . The problem of choosing a single dominant mapping thus may be written as

$$Y_0 = \arg \max_Y \sum_{i=1}^N \log \Pr(\mathbf{x}_i|Y, M) \quad (5.4)$$

For the cluster expansion shown in Figure 1, Equation 4.3 may be used to reduce this to

$$Y_0 = \arg \max_Y \sum_{\text{training set}} \log \left( \frac{m_{1,2;y_1(i,j),y_2(k,l)}^2 + \nu_{1;y_1(i,j)}^2 \nu_{2;y_2(k,l)}^2}{\left( m_{1;y_1(i,j)}^2 + \nu_{1;y_1(i,j)}^2 \right) \left( m_{2;y_2(k,l)}^2 + \nu_{2;y_2(k,l)}^2 \right)} \right) \quad (5.5)$$


---

where each training vector has been expressed in bin notation  $((i,j),(k,l))$  (as described in Appendix B). In the limit where the number of histogram counts dominates the size of the order parameters this reduces to

$$Y_0 \simeq \arg \max_Y \sum_{\text{training set}} \log \left( \frac{m_{1,2;y_1(i,j),y_2(k,l)}^2}{m_{1;y_1(i,j)}^2 m_{2;y_2(k,l)}^2} \right) \quad (5.6)$$

which is equivalent to maximising the mutual information between the outputs  $\mathbf{y}_1(\mathbf{x}_1)$  and  $\mathbf{y}_2(\mathbf{x}_2)$ .

## VI. EXTENSIONS TO THE CLUSTER EXPANSION MODEL

In [3] a similar mutual information result was obtained for an  $L$  layer cluster expansion model. A global optimisation criterion was used, which was equivalent to the sum of all of the transverse mutual informations throughout the network. In effect, the network actively adjusted its layer-to-layer transformations to “lock onto” correlations between different subspaces of the input data.

In [4] the optimisation of an  $L$  layer cluster expansion model was investigated in detail. Because the optimisation criterion depended simultaneously on what each layer was doing, the layer-to-layer mappings had to co-operate in order to find their optimum configuration. Two types of information flowed through the network: (1) Bottom-up flow of data from layer to layer, as already described above, (2) Top-down flow of control signals from

layer to layer, required to implement the optimisation. In effect the higher layers of the cluster expansion model controlled the lower layers during the optimisation process. This effect is called “self-supervision” because it is exactly like standard “supervision” of a multilayer neural network, except that here the backward propagating signals are internally generated within the network itself.

In [2] it was shown how to embed many cluster expansion models into a single translation invariant layered network structure. This is very useful for image processing applications.

## VII. CONCLUSIONS

This paper presents a Bayesian treatment of the cluster expansion model. It shows that the cluster expansion approach to density modelling allows Bayesian predictive distributions to be derived in closed form, if it is assumed that the underlying histograms follow Dirichlet prior distributions. The form of the results thus obtained is essentially the same as those previously derived using more ad hoc approaches (see e.g. [1]).

## VIII. ACKNOWLEDGEMENT

I am indebted to David MacKay for useful comments on this paper.

## Appendix A: The Dirichlet distribution

A more detailed discussion on the use of Dirichlet prior distributions is given by Skilling and Sibisi in these proceedings.

The Dirichlet distribution for  $n$  bins is defined as

$$\Pr_{\text{Dirichlet}}(\mathbf{s}|M) \equiv \delta \left( \sum_{i=1}^n s_i - 1 \right) \Gamma \left( \sum_{i=1}^n \nu_i \right) \prod_{i=1}^n \frac{s_i^{\nu_i-1}}{\Gamma(\nu_i)} \quad (A1)$$

where  $s_i$  and  $\nu_i$  are the occupation probability and the order parameter assigned to bin  $i$ , respectively. Note that  $\int d\mathbf{s} \Pr_{\text{Dirichlet}}(\mathbf{s}|M) = 1$  and  $\int d\mathbf{s} \Pr_{\text{Dirichlet}}(\mathbf{s}|M) s_i = \frac{\nu_i}{\nu_1 + \dots + \nu_n}$ .

The cluster expansion requires that the following integrals be evaluated (see Equation 4.1)

$$\begin{aligned} \text{Type 1} &= \int d\mathbf{s} \delta \left( \sum_{i=1}^n s_i - 1 \right) \prod_{i=1}^n s_i^{m_i + \nu_i - 1} \\ \text{Type 2} &= \int d\mathbf{s} \delta \left( \sum_{i=1}^n s_i - 1 \right) \prod_{i=1}^n \frac{s_i^{m_i + \nu_i - 1}}{\left( \sum_{j:y(j)=y(i)} s_j \right)^{m_i}} \end{aligned} \quad (A2)$$

The normalisation property of the Dirichlet distribution may be used to evaluate the type 1 integral to give the results (for both the numerator and denominator contributions to Equation 3.8)

$$\begin{aligned} \text{Type 1 denominator} &= \frac{\prod_{i=1}^n \Gamma(m_i + \nu_i)}{\Gamma(\sum_{i=1}^n (m_i + \nu_i))} \\ \text{Type 1 numerator} &= \frac{\Gamma(m_k + \nu_k + 1)}{\Gamma(m_k + \nu_k)} \prod_{i=1}^n \frac{\Gamma(m_i + \nu_i)}{\Gamma(\sum_{i=1}^n (m_i + \nu_i) + 1)} \end{aligned} \quad (\text{A3})$$

The type 2 integral requires rather more effort to evaluate it. Use the result

$$1 = \int dT_y \delta \left( \sum_{i:y(i)=y} s_i - T_y \right) \quad (\text{A4})$$

and introduce scaled coordinates

$$t_i \equiv \frac{s_i}{T_{y(i)}} \quad (\text{A5})$$

to transform Equation A4 into the form

$$1 = \int dT_y \frac{1}{T_y} \delta \left( \sum_{i:y(i)=y} t_i - 1 \right) \quad (\text{A6})$$

The type 2 integral then becomes

$$\text{Type 2} = \int \frac{dT_1}{T_1} \frac{dT_2}{T_2} \cdots \delta \left( \sum_y T_y - 1 \right) \left( \prod_{i=1}^n T_{y(i)}^{\nu_i} \right) \int d\mathbf{t} \prod_y \left( \delta \left( \sum_{i:y(i)=y} t_i - 1 \right) \prod_{i:y(i)=y} t_i^{m_i + \nu_i - 1} \right) \quad (\text{A7})$$

where  $\delta(\sum_{i=1}^n s_i - 1) = \delta(\sum_y T_y - 1)$  and  $ds_i = dt_i T_{y(i)}$  have been used. Note that the first and second lines of this result are separate integrals. The first line cancels between the numerator and denominator contributions in Equation 3.8, so only the second line needs to be retained, and it may be evaluated by making use of the normalisation property of the Dirichlet distribution, to yield the results

$$\begin{aligned} \text{Type 2 denominator} &= \prod_y \frac{\prod_{i:y(i)=y} \Gamma(m_i + \nu_i)}{\Gamma(\sum_{i:y(i)=y} (m_i + \nu_i))} \\ \text{Type 2 numerator} &= \frac{\Gamma(m_k + \nu_k + 1)}{\Gamma(m_k + \nu_k)} \frac{\Gamma(\sum_{i:y(i)=y(k)} (m_i + \nu_i))}{\Gamma(\sum_{i:y(i)=y(k)} (m_i + \nu_i) + 1)} \prod_y \frac{\prod_{i:y(i)=y} \Gamma(m_i + \nu_i)}{\Gamma(\sum_{i:y(i)=y} (m_i + \nu_i))} \end{aligned} \quad (\text{A8})$$

## Appendix B: Coupling the layers of the cluster expansion together

The notation that will be used is

$$\begin{aligned} m_{c;k}^l &= \text{counts in bin } k \text{ of cluster } c \text{ in layer } l \\ \nu_{c;k}^l &= \text{order parameter corresponding to bin } k \text{ of cluster } c \text{ in layer } l \end{aligned} \quad (\text{B1})$$

So, for instance, the notation used for the cluster expansion in Figure 1 consists of  $m_{1;i,j}^1$ ,  $m_{2;k,l}^1$  (layer 1, clusters 1 and 2, bins  $(i, j)$  and  $(k, l)$  respectively), and  $m_{1,2;y_1,y_2}^2$  (layer 2, cluster  $(1, 2)$ , bin  $(y_1, y_2)$ ). There is an analog-

ous notation for the order parameters. Note that the histogram counts and order parameters in the two layers of the cluster expansion in Figure 1 are related.

The histogram counts in layer 2 are determined as fol-

lows

$$m_{1,2;y_1,y_2}^2 = \sum_{\substack{i,j:y_1(i,j)=y_1 \\ k,l:y_2(k,l)=y_2}} m_{1,2;i,j,k,l}^1 \quad (\text{B2})$$

where  $m_{1,2;i,j,k,l}^1$  is the full joint histogram in layer 1, from which the histogram counts in, layer 1 may be determined by marginalisation

$$\begin{aligned} m_{1;i,j}^1 &= \sum_{k,l} m_{1,2;i,j,k,l}^1 \\ m_{2;k,l}^1 &= \sum_{i,j} m_{1,2;i,j,k,l}^1 \end{aligned} \quad (\text{B3})$$

Thus  $m_{1,2;y_1,y_2}^2$  is related to  $m_{1;i,j}^1$  and  $m_{2;k,l}^1$  via  $m_{1,2;i,j,k,l}^1$ .

The order parameters in layers 1 and 2 are related in a less obvious way than the histogram counts. Referring to Figure 1, regard the transformation from layer 1 to layer 2 as concatenation of the following two operations: (1) Each cluster histogram is rebinned into coarser bins, (2) A joint histogram is formed from 2 or more coarse binned histograms. The Dirichlet distribution has the property that when two or more bins are combined to create a

## The Cluster Expansion: A Hierarchical Density Model

larger bin, the resulting distribution is still Dirichlet, but with an order parameter equal to the sum of the order parameters of the original bins. Thus step 1 above produces the following summed order parameters

$$\begin{aligned} \text{Cluster 1} &= \sum_{i,j:y_1(i,j)=y_1} \nu_{1;i,j}^1 \\ \text{Cluster 2} &= \sum_{k,l:y_2(k,l)=y_2} \nu_{2;k,l}^1 \end{aligned} \quad (\text{B4})$$

Step 2 forces these summed order parameters to refer to the two marginalised versions of layer 2 of Figure 1, i.e. summing the layer 2 bins down the columns or along the rows, respectively. In order for the order parameters for layers 1 and 2 to be consistent with each other they must therefore satisfy the constraints

$$\begin{aligned} \text{Cluster 1: } \sum_{y_2} \nu_{1,2;y_1,y_2}^2 &= \sum_{i,j:y_1(i,j)=y_1} \nu_{1;i,j}^1 \\ \text{Cluster 2: } \sum_{y_1} \nu_{1,2;y_1,y_2}^2 &= \sum_{k,l:y_2(k,l)=y_2} \nu_{2;k,l}^1 \end{aligned} \quad (\text{B5})$$

Note that these are insufficient constraints to completely determine  $\nu_{1,2;y_1,y_2}^2$  from  $\nu_{1;i,j}^1$  and  $\nu_{2;k,l}^1$ .

- 
- [1] S P Luttrell, *Maximum entropy and Bayesian methods*, ch. The use of Bayesian and entropic methods in neural network theory, pp. 363–370, Kluwer, Dordrecht, 1989.
  - [2] ———, *A trainable texture anomaly detector using the adaptive cluster expansion (ACE) method*, Memorandum 4437, Royal Signals and Radar Establishment, Malvern, 1990.
  - [3] ———, *A hierarchical network for clutter and texture modelling*, Proceedings of SPIE conference on adaptive signal processing (San Diego), SPIE, 1991, pp. 518–528.
  - [4] ———, *Adaptive Bayesian networks*, Proceedings of SPIE conference on adaptive signal processing (Orlando), SPIE, 1992, pp. 140–151.

# The Partitioned Mixture Distribution: Multiple Overlapping Density Models \*

Stephen P Luttrell<sup>†</sup>

Defence Research Agency, St Andrews Rd, Malvern, Worcestershire, WR14 3PS, United Kingdom

In image processing problems density models are often used to characterise the local image statistics. In this paper a layered network structure is proposed, which consists of a large number of overlapping mixture distributions. This type of network is called a partitioned mixture distribution (PMD), and it may be used to apply mixture distribution models simultaneously to many different patches of an image.

## I. INTRODUCTION

A partitioned mixture distribution (PMD) [3] is a set of overlapping mixture distributions, which is used in the simultaneous density modelling of many low-dimensional subspaces of a high-dimensional dataset. This type of problem can arise in image processing, for instance. The theory of standard mixture distributions is discussed, and then extended to encompass PMDs. An expectation-maximisation (EM) optimisation scheme is derived.

## II. NOTATION

The following notation is used in this paper.  $M$  = model,  $D$  = training set of data,  $N$  = number of samples in the training set,  $\mathbf{x}$  = input vector,  $\mathbf{s}$  = parameter vector,  $Q(\mathbf{x}|\mathbf{t})$  = parametric PDF used for fitting a Bayesian predictive distribution,  $\mathbf{t}$  = parameter vector,  $\sum Q(\mathbf{x}|\mathbf{t}_c)Q(c)$  = mixture distribution form of  $Q(\mathbf{x}|\mathbf{t})$ ,  $\sum_c Q(\mathbf{x}|\mathbf{t}_c)$  = class PDF,  $Q(c)$  = prior PDF,  $c$  = class label,  $G$  = relative entropy between the fitting PDF and the Bayesian predictive distribution,  $G_0$  = relative entropy between the fitting PDF and the training data,  $\delta(\dots)$  = Dirac delta function,  $\nu$  = noise parameter,  $n$  = (odd) number of components in a mixture distribution,  $\frac{n}{2}$  = (rounded down to nearest integer) half width of a mixture window in PMD,  $n_0$  = size of PMD (or number of embedded mixture distributions),  $\mathbf{S}$  = entire set of PMD parameters,  $Q(\mathbf{x}, c|\mathbf{S})$  = joint PDF of input and class in a PMD,  $Q_c(\mathbf{x}|\mathbf{S})$  = mixture distribution centred at location  $c$  in a PMD,  $Q_c(c'|\mathbf{x}, \mathbf{S})$  = posterior probability for class  $c'$  in the mixture distribution centred at location  $c$  in a PMD,  $\tilde{Q}(c|\mathbf{x}, \mathbf{S})$  = average over overlapping mixture distributions of the posterior probability of class  $c$  in a PMD,  $\mathbf{m}_c$  = mean of a Gaussian class PDF,  $\mathbf{A}_c$  = covariance of a Gaussian class PDF,  $M_0$  = zeroth moment of the training data for class  $c$ ,  $M_1$  = first moment of

the training data for class  $c$ ,  $M_c$  = second moment of the training data for class  $c$ ,  $\tilde{Q}(c|\mathbf{S})$  = version of  $Q(c|\mathbf{S})$  converted to PMD posterior probability form (i.e. the same functional form as  $Q(c|\mathbf{x}, \mathbf{S})$ ),  $\epsilon$  = leakage factor,  $\Delta G$  = difference in relative entropy for two different choices of parameter values,  $\Delta G_0$  = lower bound for  $\Delta G$ .

## III. FITTING A PREDICTIVE DISTRIBUTION WITH A MIXTURE DISTRIBUTION

In general a Bayesian predictive distribution can be written as

$$\Pr(\mathbf{x}|D, M) = \int d\mathbf{s} \Pr(\mathbf{x}|\mathbf{s}, M) \Pr(\mathbf{s}|D, M) \quad (3.1)$$

where the integral over parameters  $\mathbf{s}$  is usually difficult to do exactly. There are many ways to alleviate this  $\mathbf{s}$  integral problem, such as the cluster expansion model discussed by Luttrell in these proceedings. However, one possible approach would be to approximate  $\Pr(\mathbf{x}|D, M)$  by a simple parametric PDF  $Q(\mathbf{x}|\mathbf{t})$ , whose parameter vector  $\mathbf{t}$  is adjusted so that  $Q(\mathbf{x}|\mathbf{t}) \simeq \Pr(\mathbf{x}|D, M)$  according to some appropriate fitting criterion. Note that it is not appropriate to think of  $Q(\mathbf{x}|\mathbf{t})$  as based on a model, because it is used solely as a numerical trick to speed up computations at the cost of some loss of accuracy. This is the reason that the notation  $Q$  is used in preference to  $\Pr$ .

The goodness of fit criterion that will be used here is the relative entropy  $G$  defined as

$$G \equiv - \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \log \left( \frac{\Pr(\mathbf{x}|D, M)}{Q(\mathbf{x}|\mathbf{t})} \right) \quad (3.2)$$

where  $G < 0$ , and  $G = 0$  iff  $Q(\mathbf{x}|\mathbf{t}) = \Pr(\mathbf{x}|D, M)$  for all  $\mathbf{x}$ . The goal is to find the parameter vector  $\mathbf{t}$  that maximises  $G$ , and which yields an optimal approximation  $Q(\mathbf{x}|\mathbf{t})$  to  $\Pr(\mathbf{x}|D, M)$ . During the fitting process itself the predictive distribution  $\Pr(\mathbf{x}|D, M)$  must be evaluated many times, which can be computationally expensive. However, once the optimum parameter vector has been located, the approximation  $Q(\mathbf{x}|\mathbf{t})$  is used thereafter in place of  $\Pr(\mathbf{x}|D, M)$ .

In this paper  $Q(\mathbf{x}|\mathbf{t})$  will be a mixture distribution [4]

$$Q(\mathbf{x}|\mathbf{t}) = \sum_{c=1}^n Q(\mathbf{x}|\mathbf{t}_c) Q(c) \quad (3.3)$$

\*Typeset in LATEX on May 22, 2019.

This appeared in Maximum Entropy and Bayesian Methods, Kluwer, J. Skilling and S. Sibisi (eds.), 279-286, 1996. © British Crown Copyright 1994 / DRA. Published with the permission of the Controller of Her Britannic Majesty's Stationery Office.

<sup>†</sup>Electronic address: luttrell@signal.dra.hmg.gb

where  $Q(\mathbf{x}|\mathbf{t}_c)$  is a normalised PDF, and the coefficients  $Q(c)$  sum up to 1. The relative entropy defined in Equation 3.2 may then be written as

$$G = \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \log \left( \sum_{c=1}^n Q(\mathbf{x}|\mathbf{t}_c) Q(c) \right) + \text{constant} \quad (3.4)$$

If the constant term is ignored, then  $G$  has the following frequentist interpretation. It is proportional to the logarithmic likelihood of drawing from the mixture distribution a large number of samples with a distribution given by the Bayesian predictive distribution.

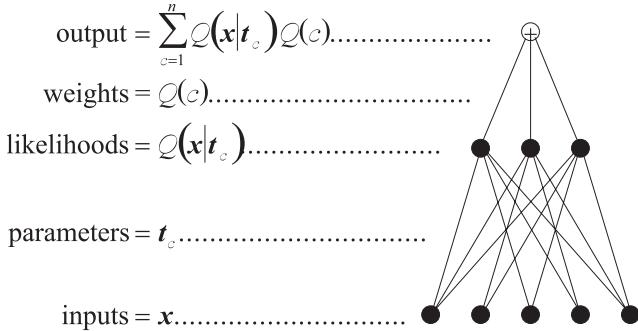


Figure 1: A mixture distribution represented as a simple type of neural network.

Figure 1 shows a three layer network representation of a mixture distribution. The nodes in the input layer hold the components of the input vector, and they are fully connected to the hidden layer by “links” that make use of the parameters  $\mathbf{t}_c$  to compute the likelihoods  $Q(\mathbf{x}|\mathbf{t}_c)$ , which are then stored in the hidden layer. The nodes in the hidden layer are then connected to the node in the output layer via conventional links weighted by the  $Q(c)$ , which do a weighted sum of the likelihoods to produce the mixture probability  $Q(\mathbf{x}|\mathbf{t})$ , which is the required output. This “neural network” is trained to maximise the relative entropy criterion in Equation 3.4. Note that the hidden-to-output weights sum up to one.

#### IV. FREQUENTIST VERSUS BAYESIAN MIXTURE OPTIMISATION

It is useful to note that the frequentist way of optimising a mixture distribution is to maximise the logarithm of the likelihood that the mixture distribution can generate the training data, which can be written as

$$G_0 = \frac{1}{N} \sum_{i=1}^N \log \left( \sum_{c=1}^n Q(\mathbf{x}_i|\mathbf{t}_c) Q(c) \right) \quad (4.1)$$

where  $i$  ranges over the training set.

Note that  $G_0$  and  $G$  are very similar except for the choice of measure to use in the integral over  $\mathbf{x}$ -space.

These measures are, respectively

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^N \delta(\mathbf{x} - \mathbf{x}_i) & \quad \text{frequentist} \\ \Pr(\mathbf{x}|D, M) & \quad \text{Bayesian} \end{aligned} \quad (4.2)$$

The Bayesian method uses the model to “fill in” the gaps between the delta functions, and in the limit of a sufficiently large amount of appropriately distributed training data will produce the same result as the frequentist method.

If a non-parametric model with a Dirichlet prior is used, then the predictive distribution reduces to

$$\Pr(\mathbf{x}|D, M) = \frac{1 - \nu}{N} \sum_{i=1}^N \delta(\mathbf{x} - \mathbf{x}_i) + \nu \quad (4.3)$$

The procedure for maximising  $G$  is then very similar to maximising  $G_0$ , whatever the size of the training set. The constant  $\nu$  term corresponds to injecting a flat background distribution of training vectors into the training set.

#### V. THE PARTITIONED MIXTURE DISTRIBUTION

A set of overlapping mixture distributions is called a partitioned mixture distribution (PMD). In order to avoid unnecessary complications, the PMD will be assumed to have periodic boundary conditions. The derivations may readily be modified to account for other types of boundary condition.

Figure 2 shows the PMD that is suggested by the mixture distribution shown in Figure 1. It is simplest to inspect Figure 2 starting from one of the output nodes. Each output node depends on a number of hidden nodes (three in this case), each of which in turn depends on a number of input nodes (five in this case). The nodes on which the highlighted output node depends are also highlighted in Figure 1. The computations performed by each layer and by the links between the layers is the same as in a standard mixture distribution (see Figure 1). Thus the highlighted output node computes a mixture distribution approximation to the marginal PDF of the highlighted nodes in the input layer. Overall, the PMD computes separate mixture distribution approximations to each of the marginal PDFs that can be obtained by laterally shifting the highlighted region in the input layer. Because these overlapping mixture distributions share parameters they are not independent of each other. This is the price that has to be paid for embedding a large number of mixture distributions in a single-layered network structure.

The generalisation of Equation 3.4 is ( $\frac{n}{2}$  is rounded down to the nearest integer)

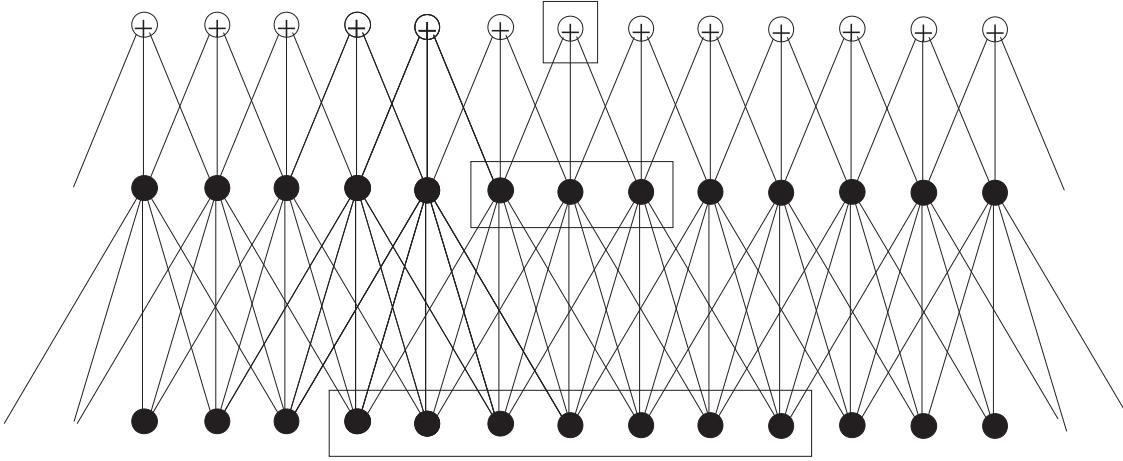


Figure 2: A partitioned mixture distribution viewed as a neural network.

$$G = \sum_{c=1}^{n_0} \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \log \left( \frac{\sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(\mathbf{x}|\mathbf{t}_{c'}) Q(c')}{\sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c')} \right) + \text{constant} \quad (5.1)$$


---

where the outer summation over  $c$  scans across the PMD, and the inner summations over  $c'$  compute the mixture distribution that is centred at location  $c$  in the PMD. For simplicity, boundary effects have been ignored (in effect, circular boundary conditions have been assumed). A normalisation factor has been included at each location  $c$  to ensure that the hidden-to-output layer weights sum up to unity for each mixture distribution. Note that each mixture distribution “sees” only a marginal PDF of the full predictive distribution  $\Pr(\mathbf{x}|D, M)$ .

The relative entropy in Equation 5.1 is then maximised with respect to the  $\mathbf{t}_c$  parameters and the  $Q(c)$  weights. The eventual result of this is a set of mixture distributions

that approximate the marginal PDFs of the predictive distribution  $\Pr(\mathbf{x}|D, M)$ .

## VI. OPTIMISING A PARTITIONED MIXTURE DISTRIBUTION

The relative entropy in Equation 5.1 can be maximised by using the so-called expectation maximisation (EM) method [1]. After some algebra (see Appendix A and [2, 3]) this yields the following iterative update algorithm for modifying the PMD parameters

---


$$\mathbf{S}_{new} = \arg \max_{\mathbf{S}} \sum_{c=1}^{n_0} \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \left( \tilde{Q}(c|\mathbf{x}, \mathbf{S}_{old}) \log Q(\mathbf{x}, c|\mathbf{S}) - \log \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'|\mathbf{S}) \right) \quad (6.1)$$

where  $\mathbf{S}$  stands for the entire set of PMD parameters  $\mathbf{S} \equiv \{\mathbf{t}_c, Q(c) : c = 1, 2, \dots, n_0\}$ , and the subscripts “old” and “new” refer to the values of these parameters on successive iterations of the EM algorithm.

There are several additional pieces of notation used in this EM update equation.

$$\begin{aligned} Q(c|\mathbf{S}) &\equiv Q(c) \text{ as specified in the parameter set } \mathbf{S} \\ Q(\mathbf{x}|c, \mathbf{S}) &\equiv Q(\mathbf{x}|\mathbf{t}_c) \text{ with } \mathbf{t}_c \text{ as specified in the parameter set } \mathbf{S} \\ \tilde{Q}(c|\mathbf{x}, \mathbf{S}) &\equiv Q(\mathbf{x}, c|\mathbf{S}) \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} \frac{1}{\sum_{c''=c'-\frac{n}{2}}^{c'+\frac{n}{2}} Q(\mathbf{x}, c''|\mathbf{S})} \end{aligned} \quad (6.2)$$


---

The first two of these definitions are straightforward, but

the definition of  $\tilde{Q}(c|\mathbf{x}, \mathbf{S})$  requires some clarification. It

is the sum of the  $n$  different posterior probabilities of  $c$  given  $\mathbf{x}$  that are computed by the mixture distributions that overlap node  $c$  in the PMD.

Apart from these details, Equation 6.1 is identical to the EM update equation for a standard mixture distribution.

## VII. GAUSSIAN PARTITIONED MIXTURE DISTRIBUTION

A Gaussian PMD uses class likelihood functions defined as follows

$$Q(\mathbf{x}|c, \mathbf{S}) \equiv \frac{1}{\sqrt{\det 2\pi \mathbf{A}_c}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m}_c)^T \mathbf{A}_c^{-1} (\mathbf{x} - \mathbf{m}_c)\right) \quad (7.1)$$

where  $\mathbf{m}_c$  is the class mean and  $\mathbf{A}_c$  is the class covariance.

The EM update equation can readily be implemented by defining the following moments

$$\begin{aligned} M_c^0(\mathbf{S}_{old}) &\equiv \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_{old}) \\ M_c^1(\mathbf{S}_{old}) &\equiv \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_{old}) \mathbf{x} \\ M_c^2(\mathbf{S}_{old}) &\equiv \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \tilde{Q}(c|\mathbf{x}, \mathbf{S}_{old}) \mathbf{x} \mathbf{x}^T \end{aligned} \quad (7.2)$$

whence the update equations reduce to

$$\begin{aligned} \tilde{Q}(c|\mathbf{S}_{new}) &= M_c^0(\mathbf{S}_{old}) \\ \mathbf{m}_c(\mathbf{S}_{new}) &= \frac{M_c^1(\mathbf{S}_{old})}{M_c^0(\mathbf{S}_{old})} \\ \mathbf{A}_c(\mathbf{S}_{new}) &= \frac{M_c^2(\mathbf{S}_{old})}{M_c^0(\mathbf{S}_{old})} - \mathbf{m}_c(\mathbf{S}_{new}) \mathbf{m}_c(\mathbf{S}_{new})^T \end{aligned} \quad (7.3)$$

where  $\tilde{Q}(c|\mathbf{S}_{new})$  is defined as

$$\tilde{Q}(c|\mathbf{S}_{new}) \equiv Q(c|\mathbf{S}_{new}) \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} \frac{1}{\sum_{c''=c'-\frac{n}{2}}^{c'+\frac{n}{2}} Q(c''|\mathbf{S}_{new})} \quad (7.4)$$

The second and third update equations yield  $\mathbf{m}_c(\mathbf{S}_{new})$  and  $\mathbf{A}_c(\mathbf{S}_{new})$  directly, but the first update equation must be solved iteratively to obtain  $Q(c|\mathbf{S}_{new})$ .

## VIII. CONCLUSIONS

A PMD is a convenient way of embedding many mixture distributions into a single-layered network structure. For instance, it may be used to build density models of a large number of separate patches of an image simultaneously. This convenience is bought at the cost of forcing neighbouring mixture distributions to share parameters. The EM method of optimising a standard mixture distribution may readily be extended to a PMD.

## Appendix A: The Expectation-Maximisation (EM) Algorithm for PMDs

The maximisation of the relative entropy in Equation 5.1 needs to be converted into an explicit training algorithm. There are two basic types of algorithm: the expectation-maximisation (EM) method [1], and various gradient ascent methods. The former is suitable for batch training, whereas the latter is suitable for on-line training. In this paper only batch training is considered.

Use the notation  $\mathbf{S}$  to denote the entire set of PMD parameters  $\mathbf{S} \equiv \{\mathbf{t}_c, Q(c) : c = 1, 2, \dots, n_0\}$ , and the subscripts “old” and “new” denote the values of these parameters on successive iterations of the EM algorithm.

First of all define the difference  $\Delta G(\mathbf{S}, \mathbf{S}_{old})$  between the relative entropy evaluated for two different choices of the parameters  $\mathbf{S}$  and  $\mathbf{S}_{old}$

---


$$\Delta G(\mathbf{S}, \mathbf{S}_{old}) = \sum_{c=1}^{n_0} \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \log \left( \frac{\sum_{c'=\frac{n}{2}-\frac{n}{2}}^{c+\frac{n}{2}} Q(\mathbf{x}|c', \mathbf{S}) Q(c'|\mathbf{S})}{\sum_{c'=\frac{n}{2}-\frac{n}{2}}^{c+\frac{n}{2}} Q(\mathbf{x}|c', \mathbf{S}_{old}) Q(c'|\mathbf{S}_{old})} \frac{\sum_{c'=\frac{n}{2}-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'|\mathbf{S}_{old})}{\sum_{c'=\frac{n}{2}-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'|\mathbf{S})} \right) \quad (A1)$$


---

Define some convenient notation. The mixture distribution evaluated at location  $c$  in the PMD has a density

$$Q_c(\mathbf{x}|\mathbf{S}) \equiv \frac{\sum_{c'=\frac{n}{2}-\frac{n}{2}}^{c+\frac{n}{2}} Q(\mathbf{x}|c', \mathbf{S}) Q(c'|\mathbf{S})}{\sum_{c'=\frac{n}{2}-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'|\mathbf{S})} \quad (A2)$$

given by

$$Q_c(\mathbf{x}|\mathbf{S}) \equiv \frac{\sum_{c'=\frac{n}{2}-\frac{n}{2}}^{c+\frac{n}{2}} Q(\mathbf{x}|c', \mathbf{S}) Q(c'|\mathbf{S})}{\sum_{c'=\frac{n}{2}-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'|\mathbf{S})} \quad (A2)$$

and the posterior probability for class  $c'$  in the mixture distribution centred at location  $c$  in the PMD is

$$Q_c(c'|\mathbf{x}, \mathbf{S}) \equiv \begin{cases} \frac{Q(\mathbf{x}|c', \mathbf{S}) Q(c'|\mathbf{S})}{\sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(\mathbf{x}|c'', \mathbf{S}) Q(c''|\mathbf{S})} & |c' - c| \leq \frac{n}{2} \\ 0 & |c' - c| > \frac{n}{2} \end{cases} \quad (\text{A3})$$

Then the EM update prescription can be derived as follows

---

$$\begin{aligned} \Delta G(\mathbf{S}, \mathbf{S}_{old}) &= \sum_{c=1}^{n_0} \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \log \left( \frac{\sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(\mathbf{x}|c', \mathbf{S}) Q(c'|\mathbf{S})}{Q_c(\mathbf{x}|\mathbf{S}_{old}) \sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c''|\mathbf{S})} \right) \\ &= \sum_{c=1}^{n_0} \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \log \left( \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{S}_{old}) \frac{Q(\mathbf{x}|c', \mathbf{S}) Q(c'|\mathbf{S})}{Q_c(\mathbf{x}, c'|\mathbf{S}_{old}) \sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c''|\mathbf{S})} \right) \\ &\geq \sum_{c=1}^{n_0} \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{S}_0) \log \left( \frac{Q(\mathbf{x}|c', \mathbf{S}) Q(c'|\mathbf{S})}{Q_c(\mathbf{x}|c', \mathbf{S}_{old}) Q(c'|\mathbf{S}_{old})} \frac{\sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c''|\mathbf{S}_{old})}{\sum_{c''=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c''|\mathbf{S})} \right) \\ &= \Delta G_0(\mathbf{S}, \mathbf{S}_{old}) \end{aligned} \quad (\text{A4})$$

where  $\Delta G_0(\mathbf{S}, \mathbf{S}_{old})$  is given by

$$\begin{aligned} \Delta G(\mathbf{S}, \mathbf{S}_{old}) &= \sum_{c=1}^{n_0} \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \left( \tilde{Q}(c|\mathbf{x}, \mathbf{S}_{old}) \log Q(\mathbf{x}, c|\mathbf{S}) - \log \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'|\mathbf{S}) \right) \\ &\quad + \text{a term that is independent of } \mathbf{S} \end{aligned} \quad (\text{A5})$$

The penultimate step of the derivation in Equation A4 was obtained by using Jensen's inequality for convex functions, and the last step was obtained by using the following results

$$\sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{S}) = 1 \quad (\text{A6})$$

and

$$\begin{aligned} \sum_{c=1}^{n_0} \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{S}) (\dots) &= \sum_{c'=1}^{n_0} \sum_{c=c'-\frac{n}{2}}^{c'+\frac{n}{2}} Q_c(c'|\mathbf{x}, \mathbf{S}) (\dots) \\ &= \sum_{c'=1}^{n_0} \tilde{Q}(c'|\mathbf{x}, \mathbf{S}) (\dots) \end{aligned} \quad (\text{A7})$$

where the term  $(\dots)$  does not depend on  $c$ . Combining the above results yields

$$G(\mathbf{S}) \geq \Delta G(\mathbf{S}, \mathbf{S}_{old}) + G(\mathbf{S}_{old}) \quad (\text{A8})$$

where  $\Delta G(\mathbf{S}, \mathbf{S}) = 0$ . Thus  $\Delta G(\mathbf{S}, \mathbf{S}_{old}) + G(\mathbf{S}_{old})$  can be used to locate the greatest lower bound of  $G(\mathbf{S})$ , so a single update of the EM algorithm is implemented as

$$\mathbf{S}_{new} = \arg \max_{\mathbf{S}} \Delta G(\mathbf{S}, \mathbf{S}_{old}) \quad (\text{A9})$$

which may be written out in full as

$$\mathbf{S}_{new} = \arg \max_{\mathbf{S}} \sum_{c=1}^{n_0} \int d\mathbf{x} \Pr(\mathbf{x}|D, M) \left( \tilde{Q}(c|\mathbf{x}, \mathbf{S}_{old}) \log Q(\mathbf{x}, c|\mathbf{S}) - \log \sum_{c'=c-\frac{n}{2}}^{c+\frac{n}{2}} Q(c'|\mathbf{S}) \right) \quad (\text{A10})$$

- [1] A P Dempster, N M Laird, and D B Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society B **39** (1977), no. 1, 1–38.
- [2] S P Luttrell, *An adaptive Bayesian network for low-level image processing*, Proceedings of IEE conference on artificial neural networks (Brighton), IEE, 1993, pp. 61–65.
- [3] ———, *Partitioned mixture distribution: an adaptive Bayesian network for low-level image processing*, IEE Proceedings on Vision, Image, and Signal Processing **141** (1994), no. 4, 251–260.
- [4] D M Titterington, A F Smith, and U E Makov, *Statistical analysis of finite mixture distributions*, Wiley, New York, 1985.

# Designing Analysable Networks \*

Stephen P Luttrell<sup>†</sup>  
Defence Research Agency, Malvern, UK

**Abstract:** In this section a unified theoretical model of unsupervised neural networks based on the theoretical ideas in [10] is presented. The analysis starts with a probabilistic model of the discrete neuron firing events that occur when a set of neurons is exposed to an input vector, and then uses Bayes' theorem to build a probabilistic description of the input vector from knowledge of the firing events. This sets the scene for unsupervised training of the network, by minimisation of the expected value of a distortion measure between the true input vector and the input vector inferred from the firing events. Various models of this type are investigated. For instance, if the model of the neurons permits firing to occur only within a defined cluster of neurons, and further, that only one firing event is observed, then the theory approximates the well-known topographic mapping network [2].

## I. INTRODUCTION

The purpose of this section is to present a probabilistic analysis of an unsupervised neural network model. In Section II the neural network model is presented together with its probabilistic description. In Section III the network optimisation criterion (i.e. an objective function) is presented and analysed, and in Section IV a useful upper bound to the objective function is derived that is much easier to optimise than the full objective function. In Section V a very simple neural network model is discussed in which only one neuron is permitted to fire in response to the input vector; this is equivalent to a vector quantiser [4]. In Section VI a related neural network model is discussed in which neurons in a single cluster fire in response to the input vector; this is equivalent to the well-known topographic mapping network [2], as was shown in [7, 10]. The theory provides a natural interpretation of the topographic neighbourhood function. In Section VII a neural network model is discussed in which a single neuron in each of many clusters of neurons fires in response to the input; this is equivalent to the “self-supervised” network that was discussed in [9, 10]. In Section VIII various pieces of research that are related to the theory presented in this section are briefly mentioned.

## II. PROBABILISTIC NEURAL NETWORK MODEL

The basic neural network model will describe the behaviour of a pair of layers of neurons, called the “input” and “output” layer. The locations of the neurons that “fire” in the output layer will be described probabilistically. Denote the rates of firing of the neurons in the

input layer by the vector  $\mathbf{x}$ , where  $\dim \mathbf{x}$  is equal to the number of neurons in the input layer. Denote the location of a neuron that fires in the output layer by the vector  $\mathbf{y}$ , which is assumed to sit on a  $d$ -dimensional rectangular lattice of size  $\mathbf{m}$  (where  $\dim \mathbf{m} = d$ ), so  $\dim \mathbf{m} = 2$  for a 2-dimensional sheet of output neurons.

The answer to the question “Which output neuron will fire next?” is then  $\Pr(\mathbf{y}|\mathbf{x})$ , which is the probability distribution over possible locations  $\mathbf{y}$  of the next neuron that fires, given that the input  $\mathbf{x}$  is known. More generally, the answer to the question “Which  $n$  neurons will fire next?” is then  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n|\mathbf{x})$ , which is a joint probability distribution over the possible locations  $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  of the next  $n$  neurons that fire. Note that the  $\mathbf{y}_i$  are not restricted to being different from each other, so a given neuron might fire more than once. Marginal probabilities may be derived from  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n|\mathbf{x})$  to give the probability of occurrence of a subset of the events in  $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$ . Thus, to obtain a marginal probability, the locations of the unobserved firing events must be summed over. Care has to be taken when forming marginal probabilities. For instance, in the  $n = 3$  case the marginal probabilities for  $(?, \mathbf{y}_1, \mathbf{y}_2)$ ,  $(\mathbf{y}_1, ?, \mathbf{y}_2)$  and  $(\mathbf{y}_1, \mathbf{y}_2, ?)$  are all *different* (where the  $?$  denotes the unobserved event). However, if the *order* in which the neurons fire is *not* observed, then  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n|\mathbf{x})$  is the sum of the probabilities for all  $n!$  permutations of the sequence of firings, in which case  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n|\mathbf{x})$  is a symmetric function of  $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$ , and in the  $n = 3$  case the marginal probabilities for  $(?, \mathbf{y}_1, \mathbf{y}_2)$ ,  $(\mathbf{y}_1, ?, \mathbf{y}_2)$  and  $(\mathbf{y}_1, \mathbf{y}_2, ?)$  are all the *same*. If the number of firings is itself known only probabilistically (i.e. as  $\Pr(n)$ ) then an appropriate average  $\sum_{n=0}^{\infty} \Pr(n)$  must be formed.

It is important to distinguish between the neural network *itself*, whose input-output state after  $n$  neurons have fired is described by the vector  $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n; \mathbf{x})$ , and the *knowledge of* the network input-output relationship, which is written as  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n|\mathbf{x})$ . For instance, a piece of software that is written to compute quantities like  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n|\mathbf{x})$  is not really a “neural network” program, rather, it is a program that makes probabilistic statements about how a neural network be-

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This paper appeared in Handbook of Neural Computation, 1995, chapter B5.3. © 1995 IOP Publishing Ltd and Oxford University Press.

<sup>†</sup>Electronic address: luttrell@signal.dra.hmg.gb

haves. The utility of  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x})$  it that it allows average properties of the neural network to be computed. One particular property that is of great interest is the network objective function; this is the quantity that measures the network's average performance. This is the subject of the next subsection.

### III. OPTIMISATION CRITERION

A neural network is trained by minimising a suitably defined objective function, which will be chosen to be the average Euclidean distortion  $D$  defined as [10]

$$D \equiv \sum_{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n=1}^m \int d\mathbf{x} d\mathbf{x}' \Pr(\mathbf{x}) \Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x}) \Pr(\mathbf{x}' | \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (1)$$

where  $\mathbf{x}$  and  $\mathbf{x}'$  are both vectors in input space, the  $\mathbf{y}_i$  are vectors in output space,  $\|\mathbf{x} - \mathbf{x}'\|^2$  is the square of the Euclidean distance between  $\mathbf{x}$  and  $\mathbf{x}'$ ,  $\int d\mathbf{x} \Pr(\mathbf{x})(\dots)$  is the average over input space using probability density  $\Pr(\mathbf{x})$ . It will be assumed that  $\int d\mathbf{x} \Pr(\mathbf{x})(\dots)$  is accurately approximated by an average over a suitable training set. Thus, if samples  $\mathbf{x}$  are drawn from the training set and plotted in input space, then after a large number of samples has been drawn the density of plotted points approximates  $\Pr(\mathbf{x})$ .

$\sum_{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n=1}^m \Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x})(\dots)$  is the average over output space as specified by the probabilistic neural network model, and  $\int d\mathbf{x}' \Pr(\mathbf{x}' | \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)(\dots)$  is the average over input space as specified by the *inverse* of the probabilistic neural network, i.e. the probability density of input vectors given that the location of the firing neurons is known. This is determined entirely by the other probabilities already defined, and may be written as

$$\Pr(\mathbf{x}' | \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) = \frac{\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x}') \Pr(\mathbf{x}')}{\int d\mathbf{x}'' \Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x}'') \Pr(\mathbf{x}'')} \quad (2)$$

which is an application of Bayes' theorem. This may be used to eliminate  $\Pr(\mathbf{x}' | \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  from the expression for  $D$  in Equation 1 to obtain

$$D \equiv 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n=1}^m \Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)\|^2 \quad (3)$$

where the  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  are defined as  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \equiv \int d\mathbf{x} \Pr(\mathbf{x} | \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \mathbf{x}$ . The  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  will be called "reference vectors". This means that there is a separate reference vector for each possible set of locations for the  $n$  neurons that fire. Thus the total number of reference vectors increases exponentially with  $n$ , which soon leads to an unacceptably large number of reference vectors. The next subsection introduces a theoretical trick for circumventing this difficulty.

### IV. LEAST UPPER BOUND TRICK

The exponential increase with  $n$  of the number of reference vectors  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  in Equation 3 can be

avoided [10] by minimising not  $D$ , but a suitably defined upper bound to  $D$  that depends on simplified reference vectors with the functional form  $\mathbf{x}'(\mathbf{y})$ , rather than  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$ . When this upper bound is minimised it yields a *least* upper bound on  $D$ , rather than its ideal lower bound. This is the price that has to be paid for not using the full reference vectors  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$ . The upper bound is derived as follows. Use the following identity, which holds for all  $\mathbf{x}'(\mathbf{y}_i)$

$$\mathbf{x} - \mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \equiv \frac{1}{n} \sum_{i=1}^n (\mathbf{x} - \mathbf{x}'(\mathbf{y}_i)) + \frac{1}{n} \sum_{i=1}^n (\mathbf{x}'(\mathbf{y}_i) - \mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)) \quad (4)$$

to separate  $\mathbf{x}$  from  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  and assume that  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x})$  is a symmetric function of  $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$ , to write  $D$  in Equation 3 in the form  $D = D_1 + D_2 - D_3$ , where

$$\begin{aligned} D_1 &\equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}=1}^m \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \\ D_2 &\equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1, y_2=1}^m \Pr(\mathbf{y}_1, \mathbf{y}_2 | \mathbf{x}) (\mathbf{x} - \mathbf{x}'(\mathbf{y}_1)) \cdot (\mathbf{x} - \mathbf{x}'(\mathbf{y}_2)) \\ D_3 &\equiv 2 \sum_{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n=1}^m \Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \left\| \mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) - \frac{1}{n} \sum_{i=1}^n \mathbf{x}'(\mathbf{y}_i) \right\|^2 \end{aligned} \quad (5)$$


---

$D_1$  is  $\frac{1}{n}$  times the average Euclidean distortion that would occur if only 1 out of the  $n$  neuron firing events is observed (assuming that  $\mathbf{x}'(\mathbf{y})$  is chosen to be  $\int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ ).  $D_2$  is a new type of term that cannot be interpreted as a simple Euclidean distortion. Suppose that the locations  $\mathbf{y}_1$  and  $\mathbf{y}_2$  of 2 out of the  $n$  neuron firing events are observed (which 2 does not matter, because it is assumed that the order in which the events occur is not observed), and an attempt is made to reconstruct the input vector *independently* from each of these firing events. This produces 2 vectors  $\mathbf{x}'(\mathbf{y}_1)$  and  $\mathbf{x}'(\mathbf{y}_2)$ , and 2 error vectors  $(\mathbf{x} - \mathbf{x}'(\mathbf{y}_1))$  and  $(\mathbf{x} - \mathbf{x}'(\mathbf{y}_2))$ . The covariance of these error vectors is the average of their outer product  $\int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1, y_2=1}^m (\mathbf{x} - \mathbf{x}'(\mathbf{y}_1))(\mathbf{x} - \mathbf{x}'(\mathbf{y}_2))^T$ , and  $D_2$  is  $\frac{2(n-1)}{n}$  times the trace of this covariance matrix (i.e. the sum of its eigenvalues). Because  $D_3 > 0$ , it follows that  $D \leq D_1 + D_2$ , so minimisation of  $D_1 + D_2$  yields a least upper bound to  $D$ , as required. Note that  $D_2$  and  $D_3$  contribute only for  $n \geq 2$ . In the  $n \rightarrow \infty$  limit the contribution of  $D_1$  vanishes, and then  $D_2$  is the value that  $D$  would take if  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  were approximated by the expression  $\frac{1}{n} \sum_{i=1}^n \mathbf{x}'(\mathbf{y}_i)$  and the error term  $D_3$  were ignored. Many useful results can be obtained by minimising  $D_1 + D_2$  as defined in Equation 3 when  $n > 2$  (or minimising  $D$  itself when  $n = 1$ ), and some of these will be discussed in the following subsections.

## V. VECTOR QUANTISER MODEL - SINGLE NEURON APPROXIMATION

In the expression for  $D$  in Equation 3 assume that only a single neuron fires  $n$  times, so that  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x})$  is given by  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x}) = \delta_{\mathbf{y}_1, \mathbf{y}_1(\mathbf{x})} \delta_{\mathbf{y}_2, \mathbf{y}_2(\mathbf{x})} \dots \delta_{\mathbf{y}_n, \mathbf{y}_n(\mathbf{x})}$ , where  $\delta_{\mathbf{y}, \mathbf{y}(\mathbf{x})} = 1$  if  $\mathbf{y} = \mathbf{y}(\mathbf{x})$ , and 0 otherwise. The role of the “encoding function”  $\mathbf{y}(\mathbf{x})$  is to convert the input vector  $\mathbf{x}$  into the index of the “winning” neuron (i.e. the one that fires). This

allows  $D$  to be simplified to the form

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y}(\mathbf{x}))\|^2 \quad (6)$$

where the  $n$  argument reference vector  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  has been written using an abbreviated notation  $\mathbf{x}'(\mathbf{y}(\mathbf{x}))$ .  $D$  in Equation 6 can be minimised with respect to  $\mathbf{y}(\mathbf{x})$  to give

$$\mathbf{y}(\mathbf{x}) = \arg \min_{\mathbf{y}} \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (7)$$

where “a” means “the value of  $\mathbf{y}$  that minimises  $\dots$ ”. This is a “nearest neighbour” encoding rule because the winning neuron  $\mathbf{y}$  has reference vector that is closest to the input vector, in the Euclidean distance sense.  $D$  in Equation 6 can be minimised with respect to  $\mathbf{x}'(\mathbf{y})$  to give

$$\begin{aligned} \mathbf{x}'(\mathbf{y}) &= \frac{\int d\mathbf{x} \Pr(\mathbf{x}) \delta_{\mathbf{y}, \mathbf{y}(\mathbf{x})} \mathbf{x}}{\int d\mathbf{x} \Pr(\mathbf{x}) \delta_{\mathbf{y}, \mathbf{y}(\mathbf{x})}} \\ &= \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x} \end{aligned} \quad (8)$$

where the second line has been obtained by using Bayes’ theorem.  $\mathbf{x}'(\mathbf{y})$  is the centroid of the input vectors  $\mathbf{x}$  that are permitted given that the location  $\mathbf{y}$  of the firing neuron is known. In effect,  $\mathbf{x}'(\mathbf{y})$  is the decoder corresponding to the encoder  $\mathbf{y}(\mathbf{x})$ . Because the optimisation of  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  are mutually coupled, these 2 results (i.e. Equation 7 and Equation 8) must be iterated in order to obtain a consistent solution. This is essentially the LBG algorithm [4] for training a vector quantiser, which may be summarised as follows:

(i) Initialise the reference vectors  $\mathbf{x}'(\mathbf{y})$ , e.g. set them to different randomly selected vectors chosen from the training set.

(ii) Encode each vector  $\mathbf{x}$  in the training set using the nearest neighbour rule  $\mathbf{y}(\mathbf{x})$  in Equation 7.

- (iii) Compute the centroids on the right hand side of Equation 8.
- (iv) Update the reference vectors  $\mathbf{x}'(\mathbf{y})$  as in Equation 8.
- (v) Test if the reference vectors  $\mathbf{x}'(\mathbf{y})$  have converged, and if not then go to step(ii), otherwise stop. There are many possible convergence tests. For instance have all the reference vectors moved by less than some predefined fraction of the diameter of the volume of input space that they live in? Another possibility is, has  $D$  decreased by less than some predefined fraction of its value on the previous iteration? There is no method that is guaranteed to avoid premature termination.

The LBG algorithm is a “batch” training algorithm. An “on-line” training algorithm can be obtained by updating the  $\mathbf{x}'(\mathbf{y})$  in the direction of  $-\frac{\partial D}{\partial \mathbf{x}'(\mathbf{y})}$  (i.e. gradient descent), which yields the update prescription

$$\Delta \mathbf{x}'(\mathbf{y}(\mathbf{x})) = \varepsilon (\mathbf{x} - \mathbf{x}'(\mathbf{y}(\mathbf{x}))) \quad (9)$$

which operates as follows:

- (i) Initialise the reference vectors  $\mathbf{x}'(\mathbf{y})$ , e.g. set them to different randomly selected vectors chosen from the training set.
- (ii) Encode a vector  $\mathbf{x}$  from the training set using the nearest neighbour rule  $\mathbf{y}(\mathbf{x})$  in Equation 7.

- (iii) Move the corresponding reference vector  $\mathbf{x}'(\mathbf{y}(\mathbf{x}))$  a small amount towards the input vector  $\mathbf{x}$  as in Equation 9.

- (iv) Test if the reference vectors  $\mathbf{x}'(\mathbf{y})$  have converged, and if not then go to step(ii), otherwise stop.

Neither the batch nor the on-line training algorithms can avoid the problem of becoming trapped in a local minimum. It is prudent to run these algorithms several times on each training set, but starting from a different initial configuration of reference vectors on each run.

## VI. TOPOGRAPHIC MAPPING MODEL - SINGLE CLUSTER APPROXIMATION

Generalise the vector quantiser case studied in the Section V, so that the neurons that fire are not all forced to be the same neuron. Thus, in the expression for  $D$  in Equation 3 assume that the neurons that fire are located in a single cluster, so that  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x})$  is given by  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x}) = \Pr(\mathbf{y}_1 | \mathbf{y}(\mathbf{x})) \Pr(\mathbf{y}_2 | \mathbf{y}(\mathbf{x})) \dots \Pr(\mathbf{y}_n | \mathbf{y}(\mathbf{x}))$ , where the “shape” of the cluster is modelled by  $\Pr(\mathbf{y} | \mathbf{y}(\mathbf{x}))$ . The results for  $D_1$ , and  $D_2$  in Equation 5 then permit an upper bound on  $D$  to be obtained as

---


$$D \leq \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}=1}^m \Pr(\mathbf{y} | \mathbf{y}(\mathbf{x})) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 + \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{\mathbf{y}=1}^m \Pr(\mathbf{y} | \mathbf{y}(\mathbf{x})) \mathbf{x}'(\mathbf{y}) \right\|^2 \quad (10)$$


---

In the special case  $n = 1$ , this inequality reduces to an equality, and the second term on the right hand side of vanishes. The first term of Equation 10 is  $\frac{1}{n}$  times the average Euclidean error that occurs when only 1 neuron firing event is observed. The second term of Equation 10 is  $\frac{2(n-1)}{n}$  times the average Euclidean error that occurs when an attempt is made to reconstruct the input vector from the weighted average  $\sum_{\mathbf{y}=1}^m \Pr(\mathbf{y} | \mathbf{y}(\mathbf{x})) \mathbf{x}'(\mathbf{y})$  of the reference vectors. This term dominates when  $n \gg 1$ .

It is possible to interpret the second term of Equation 10 in terms of a “radial basis function” network. The  $\Pr(\mathbf{y} | \mathbf{y}(\mathbf{x}))$  are a set of non-linear functions that connect the input layer to a hidden layer,  $\mathbf{x}'(\mathbf{y})$  is the set of weights connecting the  $\mathbf{y}^{th}$  hidden neuron to the output layer, and  $\mathbf{x} - \sum_{\mathbf{y}=1}^m \Pr(\mathbf{y} | \mathbf{y}(\mathbf{x})) \mathbf{x}'(\mathbf{y})$  is the error vector between the input and output layers. This use of a non-linear input-to-hidden transformation plus a linear hidden-to-output transformation is the same as is used in a radial basis function network, except that here the non-linear basis functions add up to 1, and the error is measured between the input and output, rather than between a target and the output.

### A. Optimise the $n = 1$ case

$D$  itself in Equation 3 (and not merely its upper bound in Equation 10) may be minimised with respect to  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  to give [7, 10]

$$\begin{aligned} \mathbf{y}(\mathbf{x}) &= \arg \min_{\mathbf{y}} \sum_{y'=1}^m \Pr(y' | \mathbf{y}) \|\mathbf{x} - \mathbf{x}'(y')\|^2 \\ \mathbf{x}'(\mathbf{y}) &= \frac{\int d\mathbf{x} \Pr(\mathbf{x}) \Pr(\mathbf{y} | \mathbf{y}(\mathbf{x})) \mathbf{x}}{\int d\mathbf{x} \Pr(\mathbf{x}) \Pr(\mathbf{y} | \mathbf{y}(\mathbf{x}))} \\ \Delta \mathbf{x}'(\mathbf{y}) &= \varepsilon \Pr(\mathbf{y} | \mathbf{y}(\mathbf{x})) (\mathbf{x} - \mathbf{x}'(\mathbf{y})) \end{aligned} \quad (11)$$

$\mathbf{y}(\mathbf{x})$  is no longer a nearest neighbour encoding rule as it was in the vector quantiser case in Equation 7. It is a “minimum distortion” encoding rule where the winning neuron is the one that leads to the minimum expected Euclidean error. Note that the phrase “winning neuron” is used loosely in this context; it is actually the neuron that determines where the cluster of firing neurons is located. When  $n = 1$  the neuron that actually fires is somewhere in the cluster located around the winning neuron.  $\mathbf{x}'(\mathbf{y})$  is a straightforward generalisation of

the vector quantiser case in Equation 8. Both the batch and on-line versions of the training algorithm are implemented as a straightforward generalization of the batch and on-line vector quantiser training algorithms, so they will not be repeated here. In the on-line training algorithm, an important change is that each training vector  $\mathbf{x}$  causes each reference vector  $\mathbf{x}'(\mathbf{y})$  to be updated by an

amount that is proportional to  $\Pr(\mathbf{y}|\mathbf{y}(\mathbf{x}))$ . In the vector quantiser case in Equation 9 only the winning reference vector  $\mathbf{x}'(\mathbf{y}(\mathbf{x}))$  was updated.

It is useful to approximate  $\mathbf{y}(\mathbf{x})$  [7] by doing a Taylor expansion of  $\|\mathbf{x} - \mathbf{x}'(\mathbf{y}')\|^2$  in Equation 11 in powers of  $(\mathbf{y}' - \mathbf{y})$  to obtain

$$\mathbf{y}(\mathbf{x}) = \arg \min_{\mathbf{y}} \left( \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 + \sum_{i=1}^d \frac{\partial \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2}{\partial y_i} \sum_{y'=1}^m \Pr(y'|\mathbf{y}) (y'_i - y_i) + \dots \right) \quad (12)$$

where the derivatives are evaluated as finite difference expressions on the lattice of points on which  $\mathbf{y}$  sits. If the “arg min” operation is applied to the first term in isolation, then it returns a  $\mathbf{y}$  that guarantees that a  $\frac{\partial \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2}{\partial y_i} = 0$ , which ensures that the first order term in the Taylor series vanishes. So  $\mathbf{y}(\mathbf{x})$  reduces to  $\mathbf{y}(\mathbf{x}) = \arg \min_{\mathbf{y}} (\|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2) + \text{second order terms}$ , which is a nearest neighbour encoding rule. Using this approximation, the on-line training algorithm is the same as the well-known topographic mapping training algorithm [2], and  $\Pr(y'|\mathbf{y})$  plays the role of the “neighbourhood function” around the  $\mathbf{y}^{th}$  neuron.

### B. Optimise the $n > 1$ case

If  $n \gg 1$  in Equation 10 then  $D_1 \ll D_2$ , so  $D_1$  can be ignored, in which case the upper bound for  $D$  in Equation 10 can be approximately minimised with respect to  $\mathbf{y}(\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  to give

$$\begin{aligned} \mathbf{y}(\mathbf{x}) &\approx \arg \min_{\mathbf{y}} \left\| \mathbf{x} - \bar{\mathbf{x}}'(\mathbf{y}) \right\|^2 \\ \Delta \mathbf{x}'(\mathbf{y}) &\approx \varepsilon \Pr(\mathbf{y}|\mathbf{y}(\mathbf{x})) \left( \mathbf{x} - \bar{\mathbf{x}}'(\mathbf{y}(\mathbf{x})) \right) \end{aligned} \quad (13)$$

where  $\bar{\mathbf{x}}'(\mathbf{y})$  is a weighted average of the reference vectors  $\mathbf{x}'(\mathbf{y})$  defined as  $\bar{\mathbf{x}}'(\mathbf{y}) = \sum_{y'=1}^m \Pr(y'|\mathbf{y}) \mathbf{x}'(y')$ . These results may also be obtained directly from the original definition of  $D$  in Equation 3 for  $n > 1$  by making the approximation  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \approx \frac{1}{n} \sum_{i=1}^n \mathbf{x}'_i(\mathbf{y}_i)$  (i.e. ignoring  $D_3$ ) and noting that  $\frac{1}{n} \sum_{i=1}^n \mathbf{x}'_i(\mathbf{y}_i) \approx \sum_{y'=1}^m \Pr(y'|\mathbf{y}(\mathbf{x})) \mathbf{x}'(y')$  (i.e. the  $n$  neurons that fire allow a good estimate of the cluster shape  $\Pr(y'|\mathbf{y}(\mathbf{x}))$  to be made).

### VII. TOPOGRAPHIC MAPPING MODEL - MULTIPLE CLUSTER APPROXIMATION

In the expression for  $D$  in Equation 3 assume that 1 neuron located in each of  $c$  clusters fires, so that  $\Pr(\mathbf{y}|\mathbf{x})$  has the form  $\Pr(\mathbf{y}|\mathbf{x}) = \Pr(\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c | \mathbf{y}^1(\mathbf{x}), \mathbf{y}^2(\mathbf{x}), \dots, \mathbf{y}^c(\mathbf{x}))$ , where superscripts have been used for cluster indices, and the encoding function  $\mathbf{y}(\mathbf{x})$  has been partitioned as  $\mathbf{y}(\mathbf{x}) \equiv (\mathbf{y}^1(\mathbf{x}), \mathbf{y}^2(\mathbf{x}), \dots, \mathbf{y}^c(\mathbf{x}))$  to separate the pieces that locate each cluster. This allows  $D$  to be written as

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c=1}^m \Pr(\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c | \mathbf{y}^1(\mathbf{x}), \mathbf{y}^2(\mathbf{x}), \dots, \mathbf{y}^c(\mathbf{x})) \left\| \mathbf{x} - \mathbf{x}'(\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c) \right\|^2 \quad (14)$$

Partition the input space into  $c$  non-overlapping subspaces, so that the input vector  $\mathbf{x}$  is written as  $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^c)$ , and use the following identity, which holds for all  $\mathbf{x}'^i(\mathbf{y}^i)$

$$\begin{aligned} \mathbf{x} - \mathbf{x}'(\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c) &\equiv \left( (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^c) - \left( \mathbf{x}'^1(\mathbf{y}^1), \mathbf{x}'^2(\mathbf{y}^2), \dots, \mathbf{x}'^c(\mathbf{y}^c) \right) \right) \\ &\quad - \left( \mathbf{x}'(\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c) - \left( \mathbf{x}'^1(\mathbf{y}^1), \mathbf{x}'^2(\mathbf{y}^2), \dots, \mathbf{x}'^c(\mathbf{y}^c) \right) \right) \end{aligned} \quad (15)$$

where  $\mathbf{x}'^i(\mathbf{y}^i)$  lies in input subspace  $i$ , to write  $D$  in the form  $D = D_1 - D_3$ , where

$$\begin{aligned} D_1 &\equiv 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{i=1}^c \sum_{\mathbf{y}^i=\mathbf{1}}^{\mathbf{m}} \Pr(\mathbf{y}^i | \mathbf{y}^1(\mathbf{x}), \mathbf{y}^2(\mathbf{x}), \dots, \mathbf{y}^c(\mathbf{x})) \left\| \mathbf{x}^i - \mathbf{x}'^i(\mathbf{y}^i) \right\|^2 \\ D_3 &\equiv 2 \sum_{\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c=\mathbf{1}}^{\mathbf{m}} \Pr(\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c) \left\| \mathbf{x}'(\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c) - (\mathbf{x}'^1(\mathbf{y}^1), \mathbf{x}'^2(\mathbf{y}^2), \dots, \mathbf{x}'^c(\mathbf{y}^c)) \right\|^2 \end{aligned} \quad (16)$$

which should be compared with the results in Equation 5. Note that in  $D_1$  the  $i^{th}$  cluster contributes only to the average Euclidean error in the  $i^{th}$  input subspace; this was enforced by the assumed functional dependence in  $(\mathbf{x}'^1(\mathbf{y}^1), \mathbf{x}'^2(\mathbf{y}^2), \dots, \mathbf{x}'^c(\mathbf{y}^c))$ . Because  $D_3 \geq 0$  it follows that  $D \leq D_1$ , so minimisation of  $D_1$  leads to a least upper bound on  $D$ . Minimisation of  $D_1$  with respect to  $\mathbf{y}^i(\mathbf{x})$  and  $\mathbf{x}'^i(\mathbf{y}^i)$  then gives

$$\begin{aligned} (\mathbf{y}^1(\mathbf{x}), \mathbf{y}^2(\mathbf{x}), \dots, \mathbf{y}^c(\mathbf{x})) &= \arg \min_{\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c} \sum_{i=1}^c \sum_{\mathbf{y}^i=\mathbf{1}}^{\mathbf{m}} \Pr(\mathbf{y}'^i | \mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^c) \left\| \mathbf{x}^i - \mathbf{x}'^i(\mathbf{y}'^i) \right\|^2 \\ \mathbf{x}'^i(\mathbf{y}^i) &= \frac{\int d\mathbf{x} \Pr(\mathbf{x}) \Pr(\mathbf{y}^i | \mathbf{y}^1(\mathbf{x}), \mathbf{y}^2(\mathbf{x}), \dots, \mathbf{y}^c(\mathbf{x})) \mathbf{x}^i}{\int d\mathbf{x} \Pr(\mathbf{x}) \Pr(\mathbf{y}^i | \mathbf{y}^1(\mathbf{x}), \mathbf{y}^2(\mathbf{x}), \dots, \mathbf{y}^c(\mathbf{x}))} \\ \Delta \mathbf{x}'^i(\mathbf{y}^i) &= \varepsilon \Pr(\mathbf{y}^i | \mathbf{y}^1(\mathbf{x}), \mathbf{y}^2(\mathbf{x}), \dots, \mathbf{y}^c(\mathbf{x})) (\mathbf{x}^i - \mathbf{x}'^i(\mathbf{y}^i)) \end{aligned} \quad (17)$$

which is equivalent to the “self-supervised” network training algorithm that was discussed in [9, 10]. If the  $c$  subspaces were treated completely separately, then in Equation 17 the results for the  $i^{th}$  subspace would read the same as the  $n = 1$  topographic mapping case in Equation 11, with a superscript  $i$  inserted where appropriate. Now examine Equation 17 in detail. When there is more than 1 cluster of firing neurons, the effective shape of each cluster is modified by the locations of the other clusters, i.e.  $\Pr(\mathbf{y}'^i | \mathbf{y}^i(\mathbf{x})) \rightarrow \Pr(\mathbf{y}'^i | \mathbf{y}^1(\mathbf{x}), \mathbf{y}^2(\mathbf{x}), \dots, \mathbf{y}^c(\mathbf{x}))$ . So, the cluster shapes determine the winning neurons, which, in turn, determine the cluster shapes. Note, as in

the single cluster case in Section VI, that the phrase “winning neurons” refers to the neurons that determine the cluster locations  $(\mathbf{y}^1(\mathbf{x}), \mathbf{y}^2(\mathbf{x}), \dots, \mathbf{y}^c(\mathbf{x}))$ . This feedback makes the determination of which neurons are the winners a non-trivial coupled optimisation problem, in which the  $\mathbf{y}^i(\mathbf{x})$  affect each other, so they must be *jointly* optimised. In particular, the optimal  $\mathbf{y}^i(\mathbf{x})$  is a function of the whole input vector  $\mathbf{x}$ , and not merely a function of the part of  $\mathbf{x}$  that lies in the  $i^{th}$  subspace (i.e.  $\mathbf{x}^i$ ), as it would be if the subspaces were considered separately. In practice, the problem of optimizing the  $\mathbf{y}^i(\mathbf{x})$  could be solved by iterating the following set of equations

$$\mathbf{y}^i(\mathbf{x}) = \arg \min_{\mathbf{y}^i} \sum_{\mathbf{y}'^i=\mathbf{1}}^{\mathbf{m}} \Pr(\mathbf{y}'^i | \mathbf{y}^i, \{\mathbf{y}^j(\mathbf{x}) : j \neq i\}) \left\| \mathbf{x}^i - \mathbf{x}'^i(\mathbf{y}'^i) \right\|^2 \quad (18)$$

where the  $\{\mathbf{y}^j(\mathbf{x}) : j \neq i\}$  on the right hand side is obtained from the previous iteration of the equation. If this converges, then it solves the coupled optimisation problem. Although only one neuron was permitted to fire in each of the  $c$  clusters, it is straightforward to generalise these results to the case where any number of neurons may fire in each cluster. It is also possible to generalise to the more realistic case where the input subspaces overlap with each other.

## VIII. VARIOUS RELATED PIECES OF RESEARCH

In Section VI the density of reference vectors can be derived for an optimised network [8], and the result obtained is independent of the topographic neighbourhood function. This contrasts with the result obtained for a standard topographic network in [11] where the density is dependent on the topographic neighbourhood function. This difference arises from the choice of encoding prescriptions used in the two approaches; minimum distortion in [8], and nearest neighbour in [11]. The results

of Section VI may also be used to derive a hierarchical vector quantiser [5] for encoding high dimensional vectors in easy-to-implement stages. An example of the use of this approach in image compression can be found in [6]. The results of Section VI may also be interpreted

as vector quantisation for communication along a noisy channel [9]. This type of coding problem was analysed in [1, 3], but the connection with neural networks was not made.

- [1] N Farvardin, *A study of vector quantisation for noisy channels*, IEEE Transactions on Information Theory **36** (1990), no. 4, 799–809.
- [2] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [3] H Kumazawa, M Kasahara, and T Namekawa, *A construction of vector quantisers for noisy channels*, Electronics and Engineering in Japan **67** (1984), no. 4, 39–47.
- [4] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [5] S P Luttrell, *Hierarchical vector quantisation*, Proceedings of the IEE I **136** (1989), no. 6, 405–413.
- [6] ———, *Image compression using a multilayer neural network*, Pattern Recognition Letters **10** (1989), no. 1, 1–7.
- [7] ———, *Derivation of a class of training algorithms*, IEEE Transactions on Neural Networks **1** (1990), no. 2, 229–232.
- [8] ———, *Code vector density in topographic mappings: scalar case*, IEEE Transactions on Neural Networks **2** (1991), no. 4, 427–436.
- [9] ———, *Self-supervised adaptive networks*, Proceedings of the IEE F **139** (1992), no. 6, 371–377.
- [10] ———, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
- [11] H Ritter, *Asymptotic level density for a class of vector quantisation processes*, IEEE Transactions on Neural Networks **2** (1991), no. 1, 173–175.



# Using Self-Organising Maps to Classify Radar Range Profiles \*

S P Luttrell  
Defence Research Agency, U.K.

A model-based approach to radar range profile classification is presented, and it is shown to be equivalent to training a topographic mapping neural network (see Kohonen (1)) on each of the range profile categories to be classified. The topographic mapping method is basically a Euclidean distance method of classifying range profiles. However, because it is model-based, it offers much more flexibility, and will perform better in situations where there is little training data.

## I. INTRODUCTION

This paper describes a neural network that adaptively learns the properties of a set of radar range profiles. Each set of profiles is derived from radar returns from a target presented at all aspect angles to the radar, and thus comprises a complete set of possible returns from that target.

The neural network is a special kind of topographic mapping [1], which encodes the input vector (i.e. the range profile) to produce a code index (i.e. the winning neuron in a competition for which neuron has the greatest activity).

A novel feature is the use of topological prior knowledge (see Luttrell [2]) to constrain the behaviour of the neural network. The data that derives from a single target lies on a (1-dimensional) circular subspace of the input space, because the only degree of freedom is the aspect angle of the target to the radar line of sight. This constraint is imposed on the neural network to enhance its ability to learn the structure of the training data.

One such neural network may be trained on each of the training sets. Overall, the effect is to reduce the large quantity of data in each training set (about 500,000 bytes per target, typically) to a much smaller quantity of data (about 20,000 bytes for the network size reported on here) that is needed to specify the neural network parameters. The training time is of the order of minutes per training set.

The target data is divided into training and testing sets in order to compute the confusion matrix for determining the target class. The results obtained from the topographic mapping method are compared with those obtained from a simple Euclidean distance classifier.

## II. THEORY

### A. Model Of Radar Range Profiles

A radar range profile is an example of a vector of samples (i.e. radar return received in each range gate) that depends on several types of underlying variable, which are: (1) Target class. (2) Aspect angle of target, which is in the interval  $[0, 2\pi]$ . (3) Ideal range profile for a given aspect angle, which is a vector with one component for each range gate. (4) Noise process, which is also a vector with one component for each range gate. (5) Range of target. (6) Overall normalisation.

Variables (5) and (6) will be assumed to have been removed from the problem by preprocessing each range profile in a standard way. Any residual uncertainties in variables (5) and (6) will therefore be absorbed into variables (4). For each target class (variable (1)) there is thus a continuum of choices for the aspect angle (variable (2)), each of which can produce an ideal range profile (variables (3)). This is subsequently corrupted by noise processes (variables (4)) to produce the observed range profile.

This sequence of operations will now be described mathematically. Denote the target class by a discrete variable  $c$ , its aspect angle by  $\theta$ , the ideal range profile by the function  $r_c(\theta)$ , the observed range profile by  $\mathbf{x}$ , and the noise process by the conditional probability density function  $\Pr(\mathbf{x}|\mathbf{r}, c)$ . In addition to these quantities, the prior probability density function  $\Pr(\theta|c)$  over aspect angles for target class  $c$  is required.

The overall probability of the observed range profiles  $\mathbf{x}$  (which are vectors of dimension  $n$ ) is then given by

$$\Pr(\mathbf{x}|c) = \int_0^{2\pi} d\theta \Pr(\theta|c) \Pr(\mathbf{x}|r_c(\theta), c) \quad (2.1)$$

where the unknown aspect angle is integrated out.

### B. Optimising The Radar Range Profile Model

Ideally, the full Bayesian approach which computes the posterior probability over model parameters given training data should now be used. However, in this paper, a computationally cheap approximation to this approach

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

This appeared in Proceedings of the 4<sup>th</sup> International Conference on Artificial Neural Networks, Cambridge, 335-350, 1995. © British Crown Copyright 1995/DRA. Published with the permission of the Controller of Her Britannic Majesty's Stationery Office.

will be taken, in which the parameters are optimised to take sharply defined values.

The free parameters ( $\Pr(\theta|c)$ ,  $r_c(\theta)$ , and  $\Pr(\mathbf{x}|\mathbf{r}, c)$  in  $\Pr(\mathbf{x}|c)$ ) will be optimised to fit the observed distribution in a training set of range profiles derived from target class  $c$ . A convenient optimisation criterion is to maximise the logarithmic likelihood of the training set given the model, which is defined as

$$L(c) \equiv \sum_{\mathbf{x} \text{ in training set}} \log \Pr(\mathbf{x}|c) \quad (2.2)$$


---

In order to maximise  $L(c)$  it must be differentiated with respect to each variable parameter. Infinitesimal variation of  $L(c)$  yields

$$\delta L(c) \equiv \sum_{\mathbf{x} \text{ in training set}} \int_0^{2\pi} d\theta \Pr(\theta|\mathbf{x}, c) \delta \log(\Pr(\theta|c) \Pr(\mathbf{x}|\mathbf{r}_c(\theta), c)) \quad (2.3)$$

where

$$\Pr(\theta|\mathbf{x}, c) = \frac{\Pr(\theta|c) \Pr(\mathbf{x}|\mathbf{r}_c(\theta), c)}{\int_0^{2\pi} d\theta' \Pr(\theta'|c) \Pr(\mathbf{x}|\mathbf{r}_c(\theta'), c)} \quad (2.4)$$

which is the posterior probability of aspect angle given that the observed radar range profile and the target class are known.

This expression for  $\delta L(c)$  is used as follows: (1) Select a range profile  $\mathbf{x}$  at random from the training set for class  $c$ . (2) Compute the posterior probability density  $\Pr(\theta|\mathbf{x}, c)$  over aspect angles  $\theta$  given the range profile  $\mathbf{x}$ . This will be discussed below. (3) Compute the gradients of  $\log \Pr(\theta|c)$  and  $\log \Pr(\mathbf{x}|\mathbf{r}_c(\theta), c)$  with respect to their underlying parameters. (4) Increment all the parameters thus

$$\text{parameter} \longrightarrow \text{parameter} + \varepsilon \Pr(\theta|\mathbf{x}, c) \frac{\partial (\log \Pr(\theta|c) \text{ or } \log \Pr(\mathbf{x}|\mathbf{r}_c(\theta), c))}{\partial \text{parameter}} \quad (2.5)$$


---

where  $\varepsilon$  is a small update step size parameter that controls the rate of learning. Note that more sophisticated update prescriptions, such as the conjugate gradient method, could be used. The above sequence of steps is repeated many times until the parameters settle down to approximately constant values.

A simple Gaussian form for the noise model  $\Pr(\mathbf{x}|\mathbf{r}, c)$  is

$$\Pr(\mathbf{x}|\mathbf{r}, c) = \left( \frac{1}{\sqrt{2\pi}\sigma} \right)^n \exp \left( -\frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{r}\|^2 \right) \quad (2.6)$$

where  $\sigma$  is the variance of each component of the observed range profile  $\mathbf{x}$  given that the ideal range profile  $\mathbf{r}$  is known. This noise model is introduced, because its logarithm has a particularly simple form; it is possible to derive analogous results for other types of noise model. Note that the  $c$ -dependence has now been dropped, for simplicity.

For simplicity, it will also be assumed that the prior probability density of the aspect angle is uniform, so that  $\Pr(\theta|c) = \frac{1}{2\pi}$ . Note that typical training data does not correspond to targets being observed an equal number of

times at each aspect angle, which invalidates this assumption. However, it is not difficult to extend the analysis to allow for a  $\Pr(\theta|c)$  that is optimised to fit the training data.

The  $\delta \log(\Pr(\theta|c) \Pr(\mathbf{x}|\mathbf{r}_c(\theta), c))$  term in  $\delta L(c)$  thus reduces to

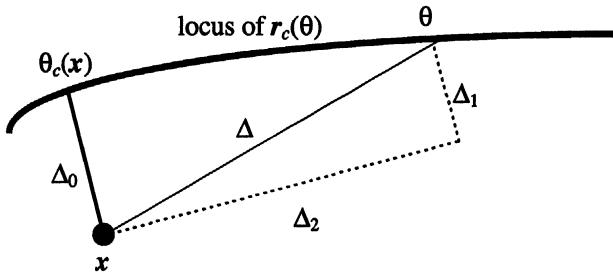
$$\delta \log(\Pr(\theta|c) \Pr(\mathbf{x}|\mathbf{r}_c(\theta), c)) = \frac{1}{\sigma^2} (\mathbf{x} - \mathbf{r}_c(\theta)) \cdot \delta \mathbf{r}_c(\theta) \quad (2.7)$$

and the  $\Pr(\theta|\mathbf{x}, c)$  term in  $\delta L(c)$  reduces to

$$\Pr(\theta|\mathbf{x}, c) = \frac{\Pr(\mathbf{x}|\mathbf{r}_c(\theta), c)}{\int_0^{2\pi} d\theta' \Pr(\mathbf{x}|\mathbf{r}_c(\theta'), c)} \quad (2.8)$$

where  $\Pr(\mathbf{x}|\mathbf{r}_c(\theta), c)$  is a Gaussian, as above.

In order to make progress this expression for  $\Pr(\theta|\mathbf{x}, c)$  must be approximated, so it is necessary to visualise clearly what it says. This is shown in Figure 1. The various pieces of notation in Figure 1 have the following meaning: (1) The filled circle is a particular choice of training vector  $\mathbf{x}$ . (2) The bold curve is a parametric curve of  $r_c(\theta)$ . (3) The point on this curve marked

Figure 1: Geometry used to approximate  $\Pr(\theta|\mathbf{x}, c)$ .

$\theta_c(\mathbf{x})$  is the point that is closest to  $\mathbf{x}$ , i.e. it minimises the norm  $\|\mathbf{x} - \mathbf{r}_c(\theta)\|^2$ . The vector joining it to  $\mathbf{x}$  (i.e.  $\mathbf{x} - \mathbf{r}_c(\theta_c(\mathbf{x}))$ ) is denoted as  $\Delta_0$ . (4) The point on the curve marked  $\theta$  is an arbitrarily chosen point. The vector joining it to  $\mathbf{x}$  (i.e.  $\mathbf{x} - \mathbf{r}_c(\theta)$ ) is denoted as  $\Delta$ . Using Pythagoras' theorem this may be written as  $\|\Delta\|^2 = \|\Delta_1\|^2 + \|\Delta_2\|^2$ , where  $\Delta_1 \propto \Delta_0$  and  $\Delta_1 \cdot \Delta_2 = 0$ .

Two approximations will be made, both of which assume that the radius of curvature of the locus of  $\mathbf{r}_c(\theta)$  is large compared with the typical values for the norm  $\|\Delta\|^2$ .

$$\begin{aligned}\|\Delta_1\|^2 &\approx \|\Delta_0\|^2 \\ \|\Delta_2\|^2 &\approx (\theta - \theta_c(\mathbf{x}))^2 \left( \frac{\partial \mathbf{r}_c(\theta)}{\partial \theta} \Big|_{\theta=\theta_c(\mathbf{x})} \right)^2\end{aligned}\quad (2.9)$$

In the second approximation the derivative is evaluated at the point  $\theta = \theta_c(\mathbf{x})$ ; it is the rate at which  $\mathbf{r}_c(\theta)$  changes as its parameter  $\theta$  is varied. For convenience its absolute value will be denoted as  $J_c(\mathbf{x})$ .  $\Pr(\theta|\mathbf{x}, c)$  may then be approximated as

$$\Pr(\theta|\mathbf{x}, c) \approx \frac{J_c(\mathbf{x})}{\sqrt{2\pi}\sigma} \exp \left( -\frac{J_c(\mathbf{x})^2}{2\sigma^2} (\theta - \theta_c(\mathbf{x}))^2 \right) \quad (2.10)$$

These results may now be combined to obtain a useful approximation to  $\delta L(c)$

---


$$\delta L(c) \approx \frac{1}{\sigma^2} \frac{J_c(\mathbf{x})}{\sqrt{2\pi}\sigma} \sum_{\mathbf{x} \text{ in training set}} \int_0^{2\pi} d\theta \exp \left( -\frac{J_c(\mathbf{x})^2}{2\sigma^2} (\theta - \theta_c(\mathbf{x}))^2 \right) (\mathbf{x} - \mathbf{r}_c(\theta)) \cdot \delta \mathbf{r}_c(\theta) \quad (2.11)$$

A gradient ascent training algorithm for  $\mathbf{r}_c(\theta)$  may be read off this result to yield: (1) Select a range profile  $\mathbf{x}$  at random from the training set. (2) Locate the closest point to  $\mathbf{x}$  on the parametric curve of  $\mathbf{r}_c(\theta)$ , and call this point  $\theta_c(\mathbf{x})$ . (3) For each  $\theta$  adjust the corresponding  $\mathbf{r}_c(\theta)$  in the following fashion

$$\mathbf{r}_c(\theta) \longrightarrow \mathbf{r}_c(\theta) + \varepsilon \frac{J_c(\mathbf{x})}{\sqrt{2\pi}\sigma} \exp \left( -\frac{J_c(\mathbf{x})^2}{2\sigma^2} (\theta - \theta_c(\mathbf{x}))^2 \right) (\mathbf{x} - \mathbf{r}_c(\theta)) \quad (2.12)$$


---

The above sequence of steps is repeated many times until the locus of  $\mathbf{r}_c(\theta)$  settles down to move in small steps about a stable configuration.

In practice it is necessary to quantise the aspect angle  $\theta$  into a large number of finite width ‘‘bins’’ that tile the interval  $[0, 2\pi]$ , each of which then has a corresponding ideal range profile  $\mathbf{r}_c(\theta)$  that has to be optimised as above. The number of bins must be chosen to be large enough that the errors due to quantisation are not too great, and small enough that the number of ideal range profiles is not too great to prevent them from being accurately optimised using the supplied training set.

### C. Relationship Between Radar Range Profile Model And Topographic Mappings

In a topographic mapping, the reference vector  $\mathbf{r}(\theta)$  attached to neuron  $\theta$  is updated in response to presentation

of an input vector  $\mathbf{x}$  from the training set thus

$$\mathbf{r}(\theta) \longrightarrow \mathbf{r}(\theta) + \varepsilon \pi(\theta - \theta(\mathbf{x})) (\mathbf{x} - \mathbf{r}(\theta)) \quad (2.13)$$

where  $\pi(\theta - \theta(\mathbf{x}))$  is the topographic mapping neighbourhood function [1], which must be defined as

$$\pi(\theta - \theta(\mathbf{x})) \equiv \sqrt{\frac{J(\mathbf{x})}{2\pi\sigma}} \exp \left( -\frac{J(\mathbf{x})^2}{2\sigma^2} (\theta - \theta(\mathbf{x}))^2 \right) \quad (2.14)$$

in order to bring this result into one-to-one correspondence with the prescription for optimising the radar range profile model. There is a separate topographic mapping for each class  $c$ .

By making a few simple approximations, the use of a topographic mapping neural network in the modelling of range profiles has thus been derived from first principles. The only unusual factor is  $\frac{J(\mathbf{x})}{\sigma}$ , which is the ratio of the distance between reference vectors to the characteristic

Table I: Table of range profile file information. They are divided into training data (11 files) and test data (8 files).

Target Class	No. of Profiles	Train/Test File
4	4966	Test 1
4	8879	Train 1
5	3643	Test 2
5	3872	Train 2
3	2604	Train 3
6	1696	Train 4
6	2216	Test 3
2	2116	Test 4
2	2636	Train 5
2	4128	Train 6
7	1839	Train 7
3	2248	Train 8
3	2476	Test 5
3	2923	Train 9
4	2560	Test 6
4	3516	Train 10
1	2085	Test 7
1	3334	Train 11
3	3619	Test 8

“radius” of the Gaussian noise process  $\Pr(\mathbf{x}|\mathbf{r}, c)$ . In this paper a crude bump-shaped neighbourhood function will be used

$$\pi(\Delta\theta) = \begin{cases} 1 - 2\nu & \Delta\theta = 0 \\ \nu & |\Delta\theta| = 1 \\ 0 & |\Delta\theta| > 1 \end{cases} \quad (2.15)$$

When using a set of topographic mappings to classify a range profile it is necessary to determine which loop of reference vectors lies closest to the observed range profile. In this paper a simple linear interpolation scheme is used to approximate the loop by a series of line segments running between neighbouring reference vectors. The distance between the observed range profile and each of these loops of line segments is then calculated.

### III. EXPERIMENT

#### A. Radar Range Profiles

A radar range profile is a set of radar returns stored by range gate. The range profile may be an incoherent average over profiles obtained at different transmission frequencies. The range profiles used in this work were obtained from 19 files. Useful information about the data files is summarised in Table I.

The range profiles required some correction in order to standardise them. Each profile was shifted by an integral number of range gates until its centroid lay as close as

possible to range gate 65 (i.e. the middle range gate). The range profiles were also normalised to unit total energy (i.e. the sum of each profile over its range gates was adjusted to be unity). These standardised range profiles then became the input vectors to the topographic mappings used in the simulations.

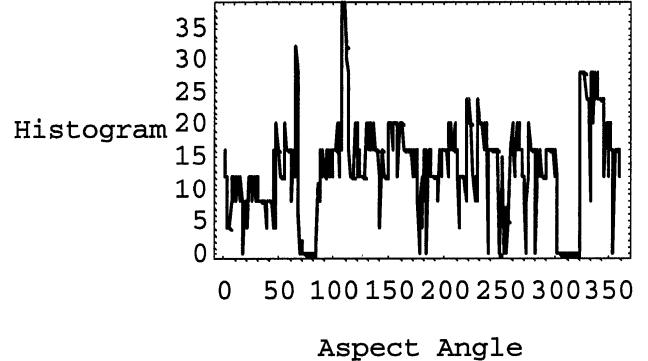


Figure 2: Aspect angle histograms for file 1. Note that the amount of data varies greatly with aspect angle. Some aspect angles have no data, and other aspect angles have a very large amount of data.

A preliminary survey of these files revealed that each had a histogram of aspect angles which was non-uniform. A typical example is shown in Figure 2. The main caveat that may be derived by inspecting Figure 2 is that one cannot assume that the frequency with which each aspect angle occurs in the training data is the same as occurs in the “real world”. The non-uniformity of the aspect angle histograms is ignored in this paper, because useful results can be obtained nevertheless.

#### B. Training The Topographic Mappings

The 11 training sets listed in Table I were each used to train a circular topographic mapping. This led to some classes having more than 1 associated topographic mapping. Other ways of using the training data are possible, corresponding to different ways of weighting the various classes. Each topographic mapping had 40 reference vectors which were initialised to 40 aspect angles selected at uniform intervals from its training set (i.e. intervals of  $\frac{1}{40}$  of the total number of profiles in the file). This guarantees that the topographic mapping has a reasonably good starting configuration before training is commenced. The training parameters used were  $\varepsilon = 0.05$  (update step size parameter) and  $\nu = 0.1$  (neighbourhood function width parameter). For each topographic mapping 1200 training vectors (i.e. 30 per reference vector) were selected at random from the corresponding training set, and the training algorithm was implemented using the above training parameters. Each topographic mapping took approximately 2 minutes to train on a DECstation 5000/260 workstation (i.e. approximately 60 SpecMarks

of computing power).

It should be noted that if the reference vectors are not initialised to have a reasonably good starting configuration, then the topographic mapping can become folded. A neat way to solve this type of problem is to augment the input space to include the aspect angle as an extra dimension, and to correspondingly augment the reference vectors. A variable weighting can then be applied to the aspect angle dimension to determine how much it should be allowed to influence the topographic mapping.

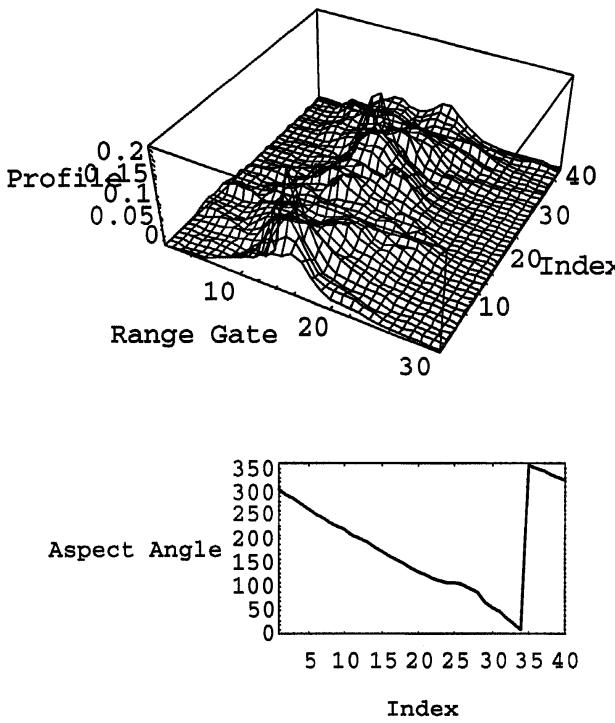


Figure 3: The top figure is the 40 reference vectors (only range gates 50 to 80 are displayed) in the topographic mapping trained on file 1. The bottom figure shows the corresponding aspect angles.

The reference vectors that are obtained by using file 1 as the training set are shown as a 3D plot in Figure 3. The aspect angles in Figure 3 show an almost linear variation with index. There is a slight departure at around an aspect angle of 100 degrees. This probably corresponds to the large peak and trough in the aspect angle histogram in Figure 2. These results are typical of topographic mappings trained on these data files.

### C. Testing The Topographic Mappings

The topographic mappings corresponding to the 11 training files are now used to classify hitherto unseen range profiles (i.e. drawn from the 8 test files, where the orientation of the ship is in the range  $\pm 40$  degree bow-on or stern-on to the radar) into one of 7 classes of target as

Table II: Confusion matrix corresponding to the results shown in Figure 4.

	Classifier			Decision				
	1	2	3	4	5	6	7	
1	72.0	11.3	9.3	0.6	0.2	2.5	4.2	
2	4.4	70.7	10.0	3.0	2.3	5.9	3.6	
True	3	2.9	10.1	67.7	1.8	3.3	9.8	4.4
Class	4	5.2	4.8	10.0	59.6	9.0	4.5	6.7
	5	1.7	2.8	24.6	4.4	57.9	6.2	2.3
	6	6.0	8.5	20.2	0.7	2.3	59.7	2.5
	7	No Test Data Available						

Table III: Confusion matrix corresponding to the results shown in Figure 5.

	Classifier			Decision				
	1	2	3	4	5	6	7	
1	68.8	15.1	7.2	0.9	0.0	3.0	5.1	
2	4.2	68.3	10.0	3.9	2.1	5.7	5.8	
True	3	3.2	11.3	66.7	2.9	3.6	8.2	4.1
Class	4	3.1	5.2	11.8	65.4	6.7	2.4	5.4
	5	2.1	3.6	24.8	4.9	55.6	7.3	1.6
	6	5.3	7.8	20.2	1.1	2.9	59.5	3.1
	7	No Test Data Available						

defined in Table II. The results in Figure 4 and Table II show the classification performance.

### D. Comparison With Standard Methods

The purpose of this paper is to demonstrate how the topographic mapping method of range profile classification may be derived from first principles, and to compare it with an existing technique. Of all the existing techniques a Euclidean distance classifier is the closest in spirit to the topographic mapping method used here.

In a simple implementation of a Euclidean distance classifier, a set of 40 average range profiles (which are effectively reference vectors) is computed over each interval of 9 degrees of aspect angle. This is repeated for each training set to yield 11 sets of 40 range profiles. Whereas, in the topographic mapping method, the range profiles (i.e. reference vectors) are carefully optimised using the topographic mapping training algorithm to maximise the likelihood of each training set given the training data.

The results in Figure 5 and Table III show the classification performance of the Euclidean distance classifier. Note that the confusion matrices produced by the Euclidean distance classifier and the topographic mapping method have a similar structure, as can be seen by comparing Figure 4 with Figure 5, and Table II with Table III.

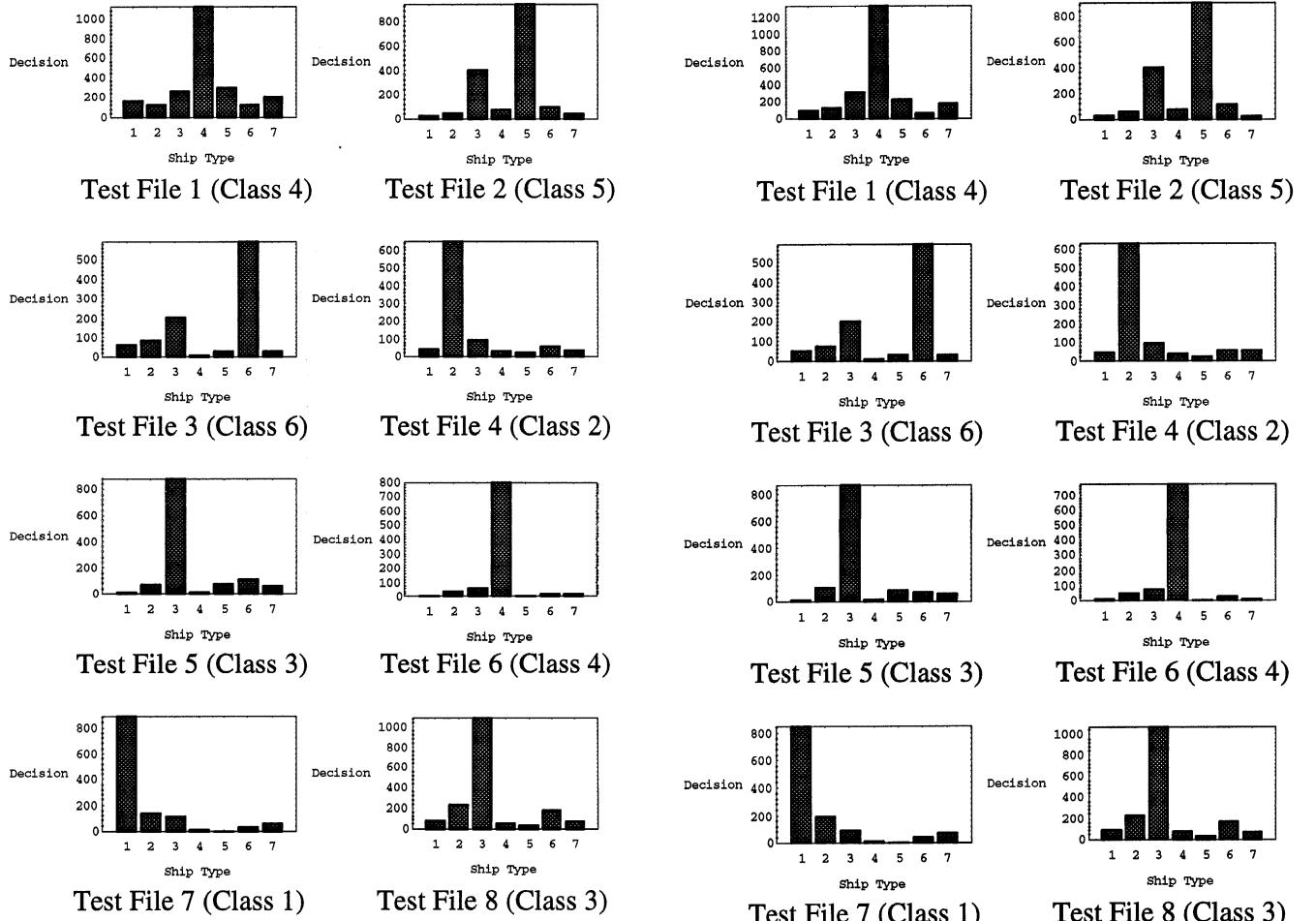


Figure 4: The histograms of classification decisions using the nearest neighbour rule. The sum of the counts in each histogram is equal to the number of range profiles in the corresponding test file.

Although the two methods lead to similar results in this case, the topographic mapping method is much to be preferred because it has been derived from an explicit model. The topographic mapping method fits a smooth loop (parameterised by a set of reference vectors) through the training data, whereas the simple Euclidean classifier method merely groups the training data into predefined bins.

#### IV. CONCLUSIONS

Maximum likelihood optimisation of the parameters in a simple range profile model has been shown to be equivalent to training a topographic mapping neural network. This is an example of the derivation of a neural network algorithm from first principles, rather than using it as a “black box”, as is more usually the case. For classifying target range profiles, the topographic mapping method has a similar classification performance to a simple Eu-

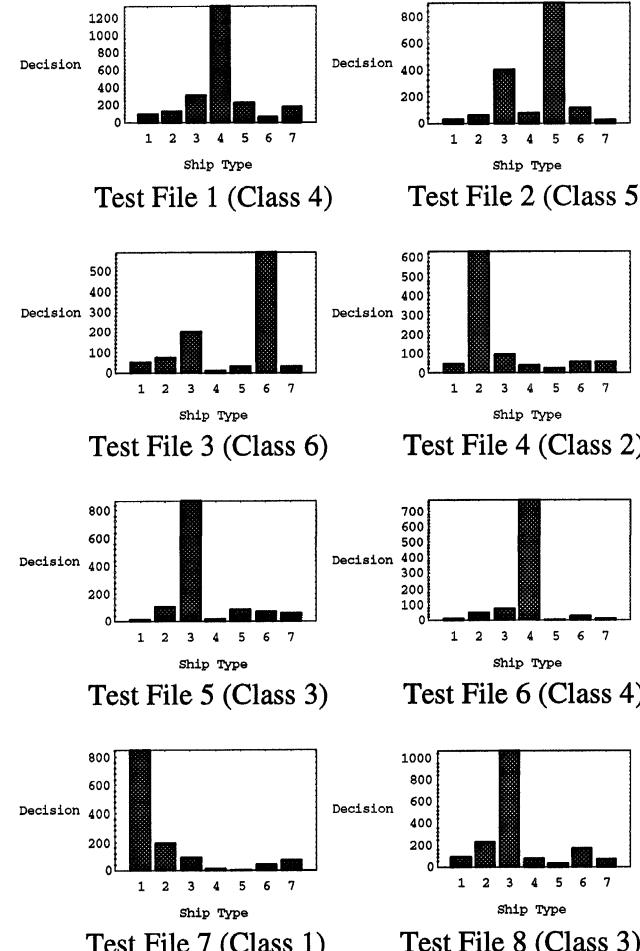


Figure 5: The same as Figure 4, but using a simple Euclidean distance classifier method.

clidean distance classifier. The reason for the similarity of these results is that the methods differ only in their choice of reference vectors. The explicit model used in the topographic mapping method will make it a more powerful technique than the simple Euclidean distance classifier method in cases where there is limited training data.

The most important lesson to be learnt from the results contained in this paper is that a simple Euclidean distance classifier method of range profile classification is at best empirical, and should be replaced by the suggested model-based approach. This gives a much more flexible framework for modelling range profiles, and it is equivalent to the Euclidean classifier method in simple cases.

#### V. ACKNOWLEDGEMENTS

The author thanks Robert Tough and Alistair Jolly for useful discussions.

- [1] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [2] S P Luttrell, *Self organising multilayer topographic map-pings*, Proceedings of IEEE conference on neural networks (San Diego), IEEE, 1988, pp. 93–100.



# A THEORY OF SELF-ORGANISING NEURAL NETWORKS \*

S P Luttrell<sup>†</sup>

*Defence Research Agency, Malvern, Worcs, WR14 3PS, UK.*

The purpose of this paper is to present a probabilistic theory of self-organising networks based on the results published in [4]. This approach allows vector quantisers and topographic mappings to be treated as different limiting cases of the same theoretical framework. The full theoretical machinery allows a visual cortex-like network to be built.

## I. INTRODUCTION

The purpose of this paper is to present a generalisation of the probabilistic approach to the static analysis of self-organising neural networks that appeared in [4]. In the simplest case the network has two layers: an input and an output layer. An input vector is used to clamp the pattern of activity of the nodes in the input layer, and the resulting pattern of individual “firing” events of the nodes in the output layer is described probabilistically. Finally, an attempt is made to reconstruct the pattern of activity in the input layer from knowledge of the location of the firing events in the output layer. This inversion from output to input is achieved by using Bayes’ theorem to invert the probabilistic feed-forward mapping from input to output. A network objective function is then introduced in order to optimise the overall network performance. If

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This appeared in Mathematics of Neural Networks: Models, Algorithms and Applications, Kluwer, Ellacott S W, Mason J C and Anderson I J (eds.), 240-244, 1997. © British Crown Copyright

the average Euclidean error between an input vector and its corresponding reconstruction is used as the objective function, then many standard self-organising networks emerge as special cases [4, 6].

In Section II the network objective function is introduced, in Section III a simpler form is derived which is an upper bound to the true objective function, and in Section IV the derivatives with respect to various parameters of this upper bound are derived. Finally, in Section V various standard neural networks are analysed within this framework.

## II. OBJECTIVE FUNCTION

The basic mathematical object is the objective function  $D$ , which is defined as

1995/DRA. Published with the permission of the Controller of Her Britannic Majesty’s Stationery Office.

<sup>†</sup>Electronic address: luttrell@signal.dra.hmg.gb

$$D \equiv \sum_{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n=1}^m \int d\mathbf{x} d\mathbf{x}' \Pr(\mathbf{x}) \Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x}) \Pr(\mathbf{x}' | \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (1)$$

where  $\mathbf{x}$  is the input vector and  $\mathbf{x}'$  is its reconstruction,  $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  are the locations in the output layer of  $n$  firing events, and  $\|\mathbf{x} - \mathbf{x}'\|^2$  is the Euclidean distance between the input vector and its reconstruction. The various probabilities arise as follows:  $\int d\mathbf{x} \Pr(\mathbf{x})(\dots)$  integrates over the training set of input vectors,  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x})$  is the joint probability of  $n$  firing events at locations  $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$ ,  $\Pr(\mathbf{x}' | \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  is the Bayes’ inverse probability that input vector  $\mathbf{x}'$  is inferred as the cause of the  $n$  firing events,  $\sum_{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n=1}^m (\dots)$  sums over all possible locations (on an assumed rectangular lattice of size  $m$ ) of the  $n$  firing events. The order in which the  $n$  firing events occurs will be assumed not to be observed, so that  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x})$  is a symmetric function of  $(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$ .

A simplifying assumption will be made where the observed firing events are assumed to be statistically inde-

pendent, so that

$$\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x}) = \Pr(\mathbf{y}_1 | \mathbf{x}) \Pr(\mathbf{y}_2 | \mathbf{x}) \cdots \Pr(\mathbf{y}_n | \mathbf{x}) \quad (2)$$

Normally, a simple form for  $\Pr(\mathbf{y} | \mathbf{x})$  is used such as

$$\Pr(\mathbf{y} | \mathbf{x}) = \frac{Q(\mathbf{x} | \mathbf{y})}{\sum_{\mathbf{y}'=1}^m Q(\mathbf{x} | \mathbf{y}')} \quad (3)$$

where  $Q(\mathbf{x} | \mathbf{y}) \geq 0$ , which guarantees that the normalisation condition  $\sum_{\mathbf{y}=1}^m \Pr(\mathbf{y} | \mathbf{x}) = 1$  holds. However, a more general expression will be used here based on the definition

$$\Pr(\mathbf{y} | \mathbf{x}; \mathbf{y}') \equiv \frac{Q(\mathbf{x} | \mathbf{y}) \delta_{\mathbf{y} \in \mathcal{N}(\mathbf{y}')}}{\sum_{\mathbf{y}'' \in \mathcal{N}(\mathbf{y}')} Q(\mathbf{x} | \mathbf{y}'')} \quad (4)$$

which implies that  $\sum_{\mathbf{y} \in \mathcal{N}(\mathbf{y}')} \Pr(\mathbf{y} | \mathbf{x}; \mathbf{y}') = 1$ , where

$\mathcal{N}(\mathbf{y}')$  is the set of node locations that lie in a predefined “neighbourhood” of  $\mathbf{y}'$ . This neighbourhood can be used to introduce “lateral inhibition” between the firing neurons if the expression for  $\Pr(\mathbf{y}|\mathbf{x})$  is written as a sum over  $\Pr(\mathbf{y}|\mathbf{x}; \mathbf{y}')$  as follows

$$\begin{aligned}\Pr(\mathbf{y}|\mathbf{x}) &= \frac{1}{M} \sum_{\mathbf{y}' \in \mathcal{N}^{-1}(\mathbf{y})} \Pr(\mathbf{y}|\mathbf{x}; \mathbf{y}') \\ &= \frac{1}{M} Q(\mathbf{x}|\mathbf{y}) \sum_{\mathbf{y}' \in \mathcal{N}^{-1}(\mathbf{y})} \frac{1}{\sum_{\mathbf{y}'' \in \mathcal{N}(\mathbf{y}')} 1}\end{aligned}\quad (5)$$

where  $\mathcal{N}^{-1}(\mathbf{y})$  is the “inverse neighbourhood” of  $\mathbf{y}$  defined as  $\mathcal{N}^{-1}(\mathbf{y}) \equiv \{\mathbf{y}' : \mathbf{y} \in \mathcal{N}(\mathbf{y}')\}$ , and  $M$  is the total number of locations in the output layer that have a non-zero neighbourhood size ( $M \equiv \sum_{\mathbf{y}: \mathcal{N}(\mathbf{y}) \neq \emptyset}$ ). This expression for  $\Pr(\mathbf{y}|\mathbf{x})$  is identical to one that arises in the context of optimising a special class of mixture distributions [5], and it satisfies  $\sum_{\mathbf{y}=1}^m \Pr(\mathbf{y}|\mathbf{x}) = 1$ . The expression for  $\Pr(\mathbf{y}|\mathbf{x})$  in Equation 5 may readily be generalised to the case where the neighbourhood is non-uniformly weighted. The expression for  $D$  in Equation 1 and the expression for  $\Pr(\mathbf{y}|\mathbf{x})$  in Equation 5 are the two basic ingredients in the theory of self-organising neural networks presented

in this paper.

There is one further ingredient that proves to be very useful.  $\Pr(\mathbf{y}|\mathbf{x})$  will be allowed to “leak” as follows [4, 5]

$$\Pr(\mathbf{y}|\mathbf{x}) \longrightarrow \sum_{\mathbf{y}' \in \mathcal{L}^{-1}(\mathbf{y})} \Pr(\mathbf{y}|\mathbf{y}') \Pr(\mathbf{y}'|\mathbf{x}) \quad (6)$$

where  $\Pr(\mathbf{y}|\mathbf{y}')$  is the amount of probability that leaks from location  $\mathbf{y}'$  to location  $\mathbf{y}$ , and  $\mathcal{L}^{-1}(\mathbf{y})$  is the “inverse leakage neighbourhood” of  $\mathbf{y}$  defined as  $\mathcal{L}^{-1}(\mathbf{y}) \equiv \{\mathbf{y}' : \mathbf{y} \in \mathcal{L}(\mathbf{y}')\}$ , where  $\mathcal{L}(\mathbf{y}')$  is the “leakage neighbourhood” of  $\mathbf{y}'$ . The purpose of leakage is to allow the network output to be “damaged” in a controlled way, so that when the network is optimised it automatically becomes robust with respect to such damage. For instance, if each node is allowed to leak probability onto its neighbours in the output layer, then when the network is optimised the node properties become topographically ordered.

### III. SIMPLIFY THE OBJECTIVE FUNCTION

The network objective function can be simplified to yield [4]

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n=1}^m \Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)\|^2 \quad (7)$$

where  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  is a “reference vector” defined as  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \mathbf{x}$ . If  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  is treated as an independent variable (i.e. its definition is ignored) then minimisation of  $D$  with respect to it yields the result  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) = \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \mathbf{x}$ , which is consistent with how it should have been defined anyway. This convenient trick allows the inverse probability  $\Pr(\mathbf{x}|\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  to be eliminated henceforth, provided that  $D$  in Equation 7 is tacitly assumed to be minimised with respect to  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$ .

$D$  can be further simplified to the form  $D = D_1 + D_2 - D_3$  [6], where

$$\begin{aligned}D_1 &\equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}=1}^m \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \\ D_2 &\equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}_1, \mathbf{y}_2=1}^m \Pr(\mathbf{y}_1, \mathbf{y}_2 | \mathbf{x}) (\mathbf{x} - \mathbf{x}'(\mathbf{y}_1)) \cdot (\mathbf{x} - \mathbf{x}'(\mathbf{y}_2)) \\ D_3 &\equiv 2 \sum_{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n=1}^m \Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) \left\| \mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n) - \frac{1}{n} \sum_{i=1}^n \mathbf{x}'(\mathbf{y}_i) \right\|^2\end{aligned}\quad (8)$$

To obtain this result  $\Pr(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n | \mathbf{x})$  has been assumed to be symmetric under permutation of the locations  $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n$  (e.g. the locations, but not the order of occurrence of the  $n$  firing events is known). If the independence assumption in Equation 2 is now invoked, then

$D_2$  may be simplified to

$$D_2 = \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{\mathbf{y}=1}^m \Pr(\mathbf{y}|\mathbf{x}) \mathbf{x}'(\mathbf{y}) \right\|^2 \quad (9)$$

The dependence of  $D_3$  on the  $n$ -argument reference vectors  $\mathbf{x}'(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  is inconvenient, because the

total number of such reference vectors is  $\mathcal{O}(|\mathbf{m}|^n)$ , where  $|\mathbf{m}|$  is the total number of output nodes ( $|\mathbf{m}| = m_1 m_2 \cdots m_d$  for a  $d$ -dimensional rectangular lattice of size  $\mathbf{m}$ ). However, the positivity of  $D_3$ , together with  $D = D_1 + D_2 - D_3$ , will be used to obtain an upper bound to  $D$  as  $D \leq D_1 + D_2$ , which depends only on 1-argument reference vectors  $\mathbf{x}'(\mathbf{y})$ . The total number of 1-argument reference vectors is equal to  $|\mathbf{m}|$ .

#### IV. DIFFERENTIATE THE OBJECTIVE FUNCTION

In order to implement an optimisation algorithm the derivatives of  $D_1$  and  $D_2$  with respect to the various pa-

rameters must be obtained. The expressions that are encountered when differentiating are rather cumbersome, but they have a simple structure which can be made clear by the introduction of the following notation

$$\begin{aligned}
 L_{\mathbf{y}, \mathbf{y}'} &\equiv \Pr(\mathbf{y}' | \mathbf{y}) & P_{\mathbf{y}, \mathbf{y}'} &\equiv \Pr(\mathbf{y}' | \mathbf{x}; \mathbf{y}) \\
 p_{\mathbf{y}} &\equiv \sum_{\mathbf{y}' \in \mathcal{N}^{-1}(\mathbf{y})} P_{\mathbf{y}', \mathbf{y}} & (L^T p)_{\mathbf{y}} &\equiv \sum_{\mathbf{y}' \in \mathcal{L}^{-1}(\mathbf{y})} L_{\mathbf{y}', \mathbf{y}} p_{\mathbf{y}'} \\
 \mathbf{d}_{\mathbf{y}} &\equiv \mathbf{x} - \mathbf{x}'(\mathbf{y}) & (L \mathbf{d})_{\mathbf{y}} &\equiv \sum_{\mathbf{y}' \in \mathcal{L}(\mathbf{y})} L_{\mathbf{y}, \mathbf{y}'} \mathbf{d}_{\mathbf{y}'} \\
 (P L \mathbf{d})_{\mathbf{y}} &\equiv \sum_{\mathbf{y}' \in \mathcal{N}(\mathbf{y})} P_{\mathbf{y}, \mathbf{y}'} (L \mathbf{d})_{\mathbf{y}'} & (P^T P L \mathbf{d})_{\mathbf{y}} &\equiv \sum_{\mathbf{y}' \in \mathcal{N}^{-1}(\mathbf{y})} P_{\mathbf{y}', \mathbf{y}} (P L \mathbf{d})_{\mathbf{y}'} \\
 e_{\mathbf{y}} &\equiv \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 & (L e)_{\mathbf{y}} &\equiv \sum_{\mathbf{y}' \in \mathcal{L}(\mathbf{y})} L_{\mathbf{y}, \mathbf{y}'} e_{\mathbf{y}'} \\
 (P L e)_{\mathbf{y}} &\equiv \sum_{\mathbf{y}' \in \mathcal{N}(\mathbf{y})} P_{\mathbf{y}, \mathbf{y}'} (L e)_{\mathbf{y}'} & (P^T P L e)_{\mathbf{y}} &\equiv \sum_{\mathbf{y}' \in \mathcal{N}^{-1}(\mathbf{y})} P_{\mathbf{y}', \mathbf{y}} (P L e)_{\mathbf{y}'} \\
 \bar{\mathbf{d}} &\equiv \sum_{\mathbf{y}=1}^m (L^T p)_{\mathbf{y}} \mathbf{d}_{\mathbf{y}} & \text{or} & \quad \bar{\mathbf{d}} \equiv \sum_{\mathbf{y}=1}^m (P L \mathbf{d})_{\mathbf{y}} & (10)
 \end{aligned}$$

which allows Equation 6 to be written as  $\Pr(\mathbf{y} | \mathbf{x}) \rightarrow \frac{1}{M} (L^T p)_{\mathbf{y}}$ .  $D_1$  and  $D_2$  may be differentiated with respect to  $\mathbf{x}'(\mathbf{y})$  to obtain

$$\begin{aligned}
 \frac{\partial D_1}{\mathbf{x}'(\mathbf{y})} &= -\frac{4}{nM} \int d\mathbf{x} \Pr(\mathbf{x}) (L^T p)_{\mathbf{y}} \mathbf{d}_{\mathbf{y}} \\
 \frac{\partial D_2}{\mathbf{x}'(\mathbf{y})} &= -\frac{4(n-1)}{nM^2} \int d\mathbf{x} \Pr(\mathbf{x}) (L^T p)_{\mathbf{y}} \bar{\mathbf{d}}
 \end{aligned} \tag{11}$$

$D_1$  and  $D_2$  may be functionally varied with respect to  $\log Q(\mathbf{x} | \mathbf{y})$  to obtain

$$\begin{aligned}
 \delta D_1 &= \frac{2}{nM} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}=1}^m \delta \log Q(\mathbf{x} | \mathbf{y}) \left( p_{\mathbf{y}} (L e)_{\mathbf{y}} - (P^T P L e)_{\mathbf{y}} \right) \\
 \delta D_2 &= \frac{4(n-1)}{nM^2} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}=1}^m \delta \log Q(\mathbf{x} | \mathbf{y}) \left( p_{\mathbf{y}} (L \mathbf{d})_{\mathbf{y}} - (P^T P L \mathbf{d})_{\mathbf{y}} \right) \cdot \bar{\mathbf{d}}
 \end{aligned} \tag{12}$$

If  $Q(\mathbf{x} | \mathbf{y})$  is assumed to be a sigmoid function

$$Q(\mathbf{x} | \mathbf{y}) = \frac{1}{1 + \exp(-\mathbf{w}(\mathbf{y}) \cdot \mathbf{x} - b(\mathbf{y}))} \tag{13}$$

then the derivatives of  $D_1$  and  $D_2$  with respect to the “weight” vector  $\mathbf{w}(\mathbf{y})$  and “bias”  $b(\mathbf{y})$  may be written as

$$\begin{aligned}\frac{\partial D_1}{\partial \begin{pmatrix} b(\mathbf{y}) \\ \mathbf{w}(\mathbf{y}) \end{pmatrix}} &= \frac{2}{nM} \int d\mathbf{x} \Pr(\mathbf{x}) \left( p_{\mathbf{y}}(L e)_{\mathbf{y}} - (P^T P L e)_{\mathbf{y}} \right) (1 - Q(\mathbf{x}|\mathbf{y})) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \\ \frac{\partial D_2}{\partial \begin{pmatrix} b(\mathbf{y}) \\ \mathbf{w}(\mathbf{y}) \end{pmatrix}} &= \frac{4(n-1)}{nM^2} \int d\mathbf{x} \Pr(\mathbf{x}) \left( p_{\mathbf{y}}(L \mathbf{d})_{\mathbf{y}} - (P^T P L \mathbf{d})_{\mathbf{y}} \right) \cdot \bar{\mathbf{d}} (1 - Q(\mathbf{x}|\mathbf{y})) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}\end{aligned}\quad (14)$$

The gradients in Equation 11 and Equation 14 may be used to implement a gradient descent algorithm for optimising  $D_1 + D_2$ , which then leads to a least upper bound on the full objective function  $D$  ( $= D_1 + D_2 - D_3$ ).

## V. SPECIAL CASES

Various standard results that are special cases of the model presented above are discussed in the following subsections.

### A. Vector Quantiser and Topographic Mapping

Assume  $n = 1$  so that only 1 firing event is observed so that  $D_2 = D_3 = 0$ ,  $\mathbf{y} \in \mathcal{N}(\mathbf{y}')$   $\forall \mathbf{y}, \mathbf{y}'$  so that the neighbourhood embraces all of the output nodes, and probability leakage of the type given in Equation 6 is allowed. Then  $D$  reduces to  $D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}=1}^m \Pr(\mathbf{y}|\mathbf{y}(\mathbf{x})) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2$  [4], which leads to a behaviour that is very similar to the topographic mappings described in [2], where  $\Pr(\mathbf{y}|\mathbf{y}')$  now corresponds to the topographic neighbourhood function. In the limit where  $\Pr(\mathbf{y}|\mathbf{y}(\mathbf{x})) = \delta_{\mathbf{y}, \mathbf{y}(\mathbf{x})}$  this reduces to the criterion for optimising a vector quantiser [3].

### B. Visual Cortex Network

A “visual cortex”-like network can be built if the full theoretical machinery presented earlier is used. This network has many of the emergent properties of the mammalian visual cortex, such as orientation maps, centre-on/surround-off detectors, dominance stripes, etc (see e.g. [1] for a review of these phenomena). There is an input layer with a pattern of activity representing

the input vector, an output layer with nodes firing in response to feed-forward connections (i.e. a “recognition” model), and a mechanism for reconstructing the input from the firing events via feed-back connections (i.e. a “generative” model). The output layer has lateral inhibition implemented as in Equation 5; the neighbourhood of node  $\mathbf{y}'$  has an associated inhibition factor  $\frac{1}{\sum_{\mathbf{y}'' \in \mathcal{N}(\mathbf{y}')} Q(\mathbf{x}|\mathbf{y}'')}$ , and the overall inhibition factor for node  $\mathbf{y}$  is  $\sum_{\mathbf{y}' \in \mathcal{N}^{-1}(\mathbf{y})} \frac{1}{\sum_{\mathbf{y}'' \in \mathcal{N}(\mathbf{y}')} Q(\mathbf{x}|\mathbf{y}'')}$ , which is the sum of the inhibition factors over all nodes  $\mathbf{y}'$  that have node  $\mathbf{y}$  in their neighbourhood. This scheme for introducing lateral inhibition is discussed in greater detail in [5]. The leakage introduced by  $\Pr(\mathbf{y}|\mathbf{y}')$  induces topographical ordering as usual.

In the limit  $n \rightarrow 1$  where  $D_1$  is dominant, this network behaves like a topographic mapping network, except that the output layer splits up into a number of “domains” each of which is typically a lateral inhibition length in size, and each of which forms a separate topographic mapping. These domains are seamlessly joined together, so no domain boundaries are actually visible. In the limit  $n \rightarrow \infty$  where  $D_2$  is dominant, this network approximates its input as a superposition of reference vectors  $\sum_{\mathbf{y}=1}^m \Pr(\mathbf{y}|\mathbf{x}) \mathbf{x}'(\mathbf{y})$  (see Equation 9). Thus the network is capable of explaining the input in terms of multiple causes.

## VI. CONCLUSIONS

A single theoretical framework has been shown to describe a number of standard self-organising neural networks. This makes it easy to understand the relationship between these neural networks, and it provides a useful framework for analysing their properties.

- [1] G Goodhill, *Correlations, competition and optimality: modelling the development of topography and ocular dominance*, Technical Report CSRP 226, Sussex University, 1992.
- [2] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [3] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [4] S P Luttrell, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
- [5] ———, *Partitioned mixture distribution: an adaptive*

- Bayesian network for low-level image processing*, IEE Proceedings on Vision, Image, and Signal Processing **141** (1994), no. 4, 251–260.
- [6] ———, *Handbook of neural computation*, ch. Designing analysable networks, pp. B5.3:1–B5.3:8, IOP and OUP, Oxford, 1995.



# Partitioned Mixture Distributions: The Dynamical Case \*

S P Luttrell<sup>†</sup>

*Defence Research Agency, St Andrews Road, Malvern, Worcs, WR14 3PS, United Kingdom*

The theory of static partitioned mixture distribution networks is generalised to the dynamical case. A simulation is presented to demonstrate how this type of network may be used to track moving objects in a cluttered background.

## I. INTRODUCTION

In [3, 14] the factorial vector quantiser (FVQ) was introduced, which generalised the standard VQ [5] so that it reconstructed its input as a linear combination of more than one code vector. That extension allows an FVQ to decompose its input into a superposition of more than one factor, which enables it to discover very efficient ways of encoding inputs that arise from multiple causes. In [6–8] a network of connected VQs split up the input space into a set of disjoint subspaces, which led to a factorial interpretation of the code. However, this factorial ability was constrained by the fact that the subspaces were hard-wired in advance; this is a special case of the FVQ where the various terms in the linear combination live in separate subspaces, so no terms in the linear combination ever superpose.

In a separate line of research, into adaptive density models, the partitioned mixture distribution (PMD) was introduced in [9, 11], allowing multiple mixture models of overlapping subspaces to be embedded into a neural network architecture, where they shared parameters which could be simultaneously optimised using a variant of the standard expectation-maximisation (EM) algorithm [1].

In a development of the VQ research reported in [8] the encoder was chosen to be probabilistic, so that the relationship between the input and its encoded version was not deterministic. The theory of this type of network was further developed in [10, 13], and it was successfully applied to the problem of modelling the visual cortex in [12], where the probability over codes-given-input was chosen to be the posterior probability computed in a PMD. A PMD posterior probability allows a VQ to produce mul-

\*Typeset in LATEX on May 21, 1997.  
This appeared in Proceedings of the 5th International Conference on Artificial Neural Networks, Cambridge, 59–63, 1997. © 1997

tiple codes for each input, and this type of VQ may be used to reconstruct its input as a corresponding linear combination of code vectors. This architecture includes the FVQ as a special case.

In [2] the idea of representing the input as a linear combination of vectors (as used in the FVQ) was applied to a Gaussian mixture distribution, where the centroid of each Gaussian component was a linear combination of a set of underlying vectors, which can then be extended to a hidden (Gauss-)Markov model (HMM) which is the dynamical counterpart of the static Gaussian mixture distribution: this is called a factorial hidden Markov model (FHMM).

By using the same trick of converting a mixture distribution into an HMM, the VQ architecture using a PMD posterior probability may be generalised to become a dynamical network. This yields an architecture that is analogous to the FHMM, except that it is designed to operate as a dynamical encoder rather than as a density modeller. This work is related to the Fokker-Planck image tracking techniques used in [4].

## II. AUTOENCODER THEORY

A Bayesian theory of autoencoders was presented in [8], where an input vector  $\mathbf{x}$  was converted into a posterior probability  $\Pr(\mathbf{y}|\mathbf{x})$  over a vector  $\mathbf{y}$  of  $n$  code indices, where  $\mathbf{y} = (y_1, y_2, \dots, y_n)$  and each  $y_i$  lies in the range  $1, 2, \dots, M$  ( $y$  may be interpreted as indexing an individual code and  $\mathbf{y}$  a sequence of such codes). A suitable objective function is given by the upper bound on a Bayesian autoencoder's reconstruction error  $D$  [10, 13]

BRITISH CROWN COPYRIGHT/DRA.

<sup>†</sup>Electronic address: luttrell@signal.dra.hmg.gb

$$D \leq \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 + \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 \quad (2.1)$$

where  $\Pr(\mathbf{y}|\mathbf{x})$  is assumed to be a symmetric function of  $(y_1, y_2, \dots, y_n)$ , the marginal probability  $\Pr(y_i, y_j|\mathbf{x})$  is assumed to factorise as a pair of marginal probabilities  $\Pr(y_i|\mathbf{x}) \Pr(y_j|\mathbf{x})$  (where all marginal probabilities are

derived from  $\Pr(\mathbf{y}|\mathbf{x})$ ), and  $\mathbf{x}'(y)$  is any vector function of  $y$  (which will be called a reference vector, and which forms the basis of a factorial reconstruction of  $\mathbf{x}$ ).

### III. STATIC PMD

This theory may be used to develop an autoencoder which codes separately for separate factors in the input vector, in an analogous fashion to the factorial vector quantiser (FVQ) described in [3, 14]. Thus the vector  $\mathbf{y}$  of code indices has  $n$  components, each of which potentially could contribute a piece of information to form a reconstruction  $\mathbf{x}'(\mathbf{x})$  of the input vector  $\mathbf{x}$ . In Equation 2.1 the reconstruction vector may be obtained by inspection as  $\mathbf{x}'(\mathbf{x}) = \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y)$ . Thus each code  $y$  contributes linearly to a superposition of reference vectors  $\mathbf{x}'(y)$ , where the  $y^{\text{th}}$  code contributes in proportion to its posterior probability  $\Pr(y|\mathbf{x})$  of being selected in the first place. This interpretation is valid only in the limit  $n \rightarrow \infty$  where the second term dominates the first term in Equation 2.1, because then the relative frequency of occurrence of  $y$  in large number  $n$  of components in the vector of code indices  $\mathbf{y}$  effectively gives an accurate estimate of the posterior probability  $\Pr(y|\mathbf{x})$ . When  $n$  is finite, the vector of  $n$  codes only allows us to estimate  $\Pr(y|\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n \delta_{y,y_i}$  which implies the reconstruction  $\mathbf{x}'(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n \mathbf{x}'(y_i)$ . This result is the basis of the FVQ interpretation of the  $n > 1$  autoencoder.

In order that the reconstruction  $\mathbf{x}'(\mathbf{x})$  given above gives useful results it is necessary that minimising the upper bound on  $D$  (as given in Equation 2.1) leads to  $\Pr(y|\mathbf{x})$  and  $\mathbf{x}'(y)$  that have FVQ-like properties. Thus  $\Pr(y|\mathbf{x})$  must potentially allow more than one  $y$  to have a large probability, so that the expression for  $\mathbf{x}'(\mathbf{x})$  as a superposition of  $\mathbf{x}'(y)$  can potentially have more than one contribution. This FVQ property of  $\Pr(y|\mathbf{x})$  could be allowed to emerge by a process of self-organisation, but in this paper a less sophisticated approach will be taken in which a special type of posterior probability model will be used to enforce the FVQ property.

The basic trick is to define a whole ensemble of posterior probability models thus

$$\Pr(y|\mathbf{x}; y') = \frac{Q(\mathbf{x}|y) \delta_{y \in \mathcal{N}(y')}}{\sum_{y'' \in \mathcal{N}(y')} Q(\mathbf{x}|y'')} \quad (3.1)$$

where  $y'$  is an index that labels the posterior probability model in the ensemble (it will be assumed that  $y'$  lies in the range  $1, 2, \dots, M$ , although this is not necessary),  $Q(\mathbf{x}|y)$  is the likelihood function (which satisfies  $Q(\mathbf{x}|y) \geq 0$ ) associated with code index  $y$ ,  $\mathcal{N}(y)$  is a subset of code indices that are deemed to be “in the neighbourhood of” code index  $y$ , and  $\delta_{y \in \mathcal{N}(y')}$  is a Kronecker delta function that forces  $y$  to be in the neighbourhood of  $y'$ . This definition of  $\Pr(y|\mathbf{x}; y')$  allows  $\Pr(y|\mathbf{x}; y') > 0$  only for those code indices that satisfy  $y \in \mathcal{N}(y')$ , and guarantees the normalisation condition  $\sum_{y'=1}^M \Pr(y|\mathbf{x}; y') = 1$ . A single posterior probability is obtained from these separate models by averaging them

### Partitioned Mixture Distributions: The Dynamical Case

together

$$\Pr(y|\mathbf{x}) = \frac{1}{M} \sum_{y' \in \mathcal{N}^{-1}(y)} \Pr(y|\mathbf{x}; y') \quad (3.2)$$

where  $\mathcal{N}^{-1}(y)$  is the “inverse neighbourhood” defined as  $\mathcal{N}^{-1}(y) = \{y'|y \in \mathcal{N}(y')\}$ . This average of the  $\Pr(y|\mathbf{x}; y')$  is a special case of the Bayesian approach to combining posterior probabilities from separate models, where each model is assigned equal prior weight. The functional form of  $\Pr(y|\mathbf{x})$  is identical to that of the posterior probability over class labels that occurs in a partitioned mixture distribution (PMD) [9].

This PMD posterior probability has all of the properties that are required to guarantee FVQ-like properties. Each contributing posterior probability  $\Pr(y|\mathbf{x}; y')$  is able to produce its own encoding and reconstruction of the input vector. When the separate posterior probabilities are combined to produce the PMD posterior probability in Equation 3.2 they can no longer be used to produce separate encodings and reconstructions of the input vector; rather, they act together as a single FVQ-like encoder.

### IV. DYNAMICAL PMD

It is convenient to rewrite Equation 3.2 using the following shorthand notation

$$\begin{aligned} \mathbf{p} &\equiv (\Pr(y=1|\mathbf{x}), \dots, \Pr(y=M|\mathbf{x})) \\ \mathbf{q} &\equiv (Q(\mathbf{x}|y=1), \dots, Q(\mathbf{x}|y=M)) \\ A_{y,y'} &\equiv \begin{cases} 1 & y' \in \mathcal{N}(y) \\ 0 & y' \notin \mathcal{N}(y) \end{cases} \end{aligned} \quad (4.1)$$

which gives

$$\mathbf{p} = \frac{1}{M} \mathbf{q} \mathbf{A}^T \cdot \left( \frac{1}{\mathbf{A} \cdot \mathbf{q}} \right) \quad (4.2)$$

where the dot notation  $\cdot$  denotes the matrix-vector inner product operation,  $\frac{1}{\mathbf{A} \cdot \mathbf{q}}$  denotes the componentwise reciprocal of the vector  $\mathbf{A} \cdot \mathbf{q}$ , and  $\mathbf{q} \mathbf{A}^T \cdot \left( \frac{1}{\mathbf{A} \cdot \mathbf{q}} \right)$  denotes the componentwise product of the vectors  $\mathbf{q}$  and  $\mathbf{A}^T \cdot \left( \frac{1}{\mathbf{A} \cdot \mathbf{q}} \right)$ . Note that any  $\mathbf{A}$  that satisfies  $A_{y,y'}$  could be used in general. This abbreviated notation for  $\Pr(y|\mathbf{x})$  may readily be used to introduce a discrete-time index by using the following shorthand notation

$$\begin{aligned} \mathbf{p}(t) &\equiv (\Pr(y(t)=1|\mathbf{x}(1), \dots, \mathbf{x}(t)), \dots) \\ \mathbf{q}(t) &\equiv (Q(\mathbf{x}(t)|y(t)=1), \dots) \\ T_{y,y'} &\equiv \Pr(y|y') \end{aligned} \quad (4.3)$$

$$\mathbf{p}(t) = \frac{1}{M} (\mathbf{q}(t) \mathbf{T} \cdot \mathbf{p}(t-1)) \mathbf{A}^T \cdot \left( \frac{1}{\mathbf{A} \cdot (\mathbf{q}(t) \mathbf{T} \cdot \mathbf{p}(t-1))} \right) \quad (4.4)$$

where  $\mathbf{T}$  is a matrix of conditional probabilities which allows the vector of posterior probabilities  $\mathbf{p}(t-1)$  at time

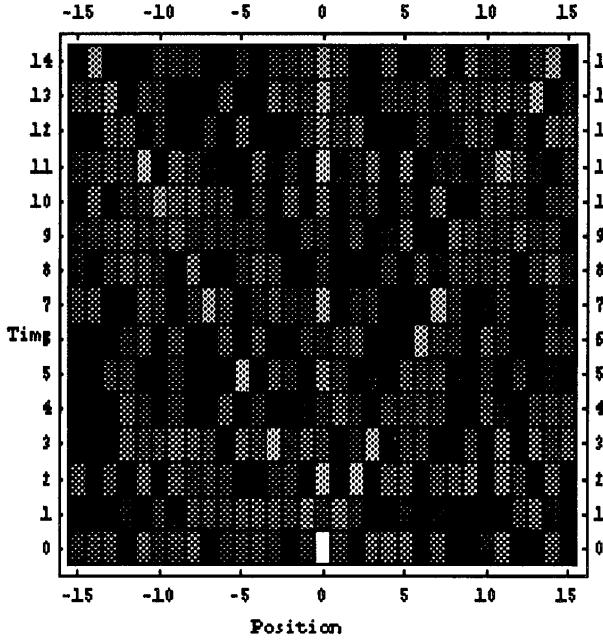


Figure 1: Example input sequence for a dynamical PMD: a sequence of 1-dimensional input vectors  $\mathbf{q}(t)$  containing 3 objects moving in a cluttered background. Light (dark) pixels correspond to high (low) input values.

$t - 1$  to be evolved into the prior probability  $\mathbf{T} \cdot \mathbf{p}(t - 1)$  at time  $t$ . Equation 4.2 describes a static PMD whereas Equation 4.4 describes a dynamical PMD. In the limit where there is only one contributing posterior probability in Equation 3.2 (i.e.  $\mathbf{A}$  is the identity matrix), Equation 4.2 reduces to a mixture distribution posterior probability, and Equation 4.4 reduces to a hidden Markov model (HMM) posterior probability. However, in this paper the focus is on autoencoders so only the posterior probabilities are used, unlike in density modelling work[2]. Note that in a dynamical PMD described by Equation 4.4 there are multiple HMMs cross-linked by the action of the transition matrix  $\mathbf{T}$ , whereas in [2] there is no such cross-linking.

## V. SIMULATION

There are many possible applications for dynamical PMDs, each of which will be characterised by the choice of transition matrix  $\mathbf{T}$ , which controls the free dynamics of the PMD posterior probability. Free dynamics occur when the likelihood functions convey no information about the input vector  $\mathbf{x}$ , i.e.  $Q(\mathbf{x}|y) = \text{constant}$ . When the likelihood functions  $Q(\mathbf{x}|y)$  have a non-trivial dependence on  $\mathbf{x}$  they convert the free dynamics into forced dynamics, as described in Equation 4.4. One application which is easy to demonstrate is the tracking of objects in a cluttered background. The dynamical PMD approach

to this problem is analogous to the Fokker-Planck approach adopted in [4].

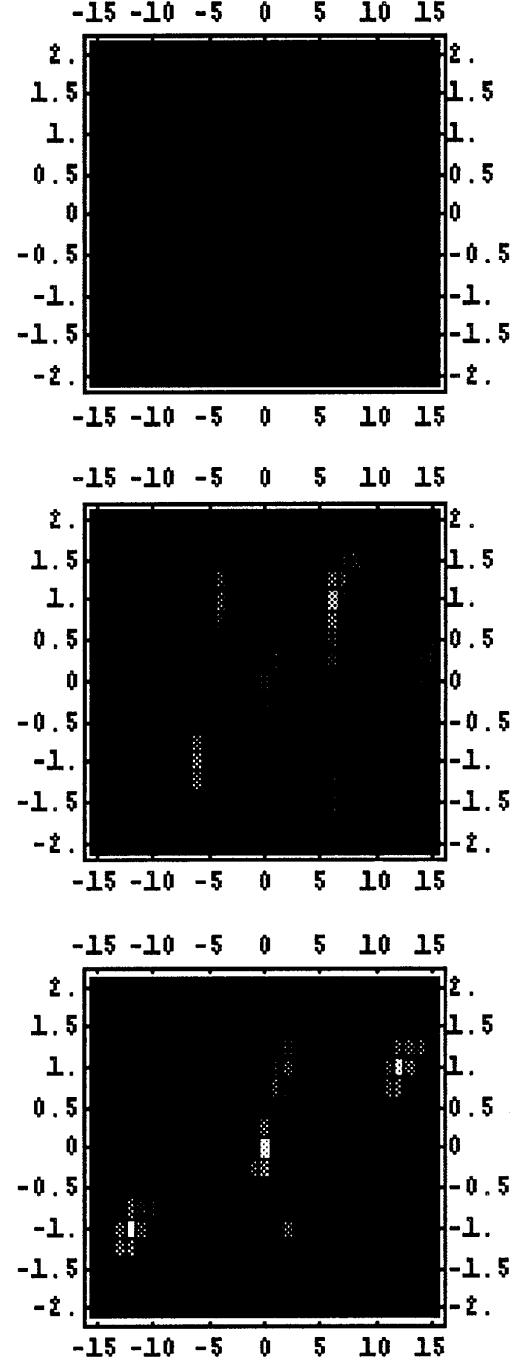


Figure 2: The PMD posterior probabilities at times  $t = 0$ ,  $t = 6$ , and  $t = 12$  (reading from left to right). In each picture the horizontal axis measures displacement along each 1-dimensional PMD, and the vertical axis measures the velocity to which each of the 17 1-dimensional PMDs is tuned. Each horizontal slice thus represents  $\mathbf{p}(t)$  for one of the 17 1-dimensional PMDs. Light (dark) pixels correspond to high (low) PMD posterior probability values.

Consider the 1-dimensional scenario shown in Figure 1, which is a sequence of input vectors  $\mathbf{q}(t)$  containing 3 objects moving in a cluttered background. A suitable 1-dimensional dynamical PMD for tracking these objects would have a transition matrix  $\mathbf{T}$  which led to free dynamics in which the PMD posterior probability propagated at constant velocity from one end to the other of the  $\mathbf{x}$ -space (usually, it is better to use a  $\mathbf{T}$  matrix which defines a narrow range of velocities, rather than a single velocity). If this velocity is matched to the velocity at which an object is moving in the input data, then the forcing induced by the effect of the likelihood function  $Q(\mathbf{x}|y)$  will reinforce the free dynamics, causing the PMD posterior probability to build up a strong peak that tracks the object across the PMD. If the  $\mathbf{A}$  matrix (that defines the FVQ-like behaviour of the PMD) is chosen appropriately, then the PMD can simultaneously track multiple objects moving at that velocity. Finally, if there is a whole set of dynamical PMDs, each tuned to a different narrow range of velocities, then a whole area of (position, velocity)-space can be covered, which allows multiple objects moving at different speeds to be tracked.

The parameters that were used in the simulation were as follows. There were 17 PMDs tuned to different velocities, and each PMD was a 1-dimensional array of 31 elements. The transition matrix  $T_{y,y'}$  for each PMD was a Gaussian function of  $y - y'$  with standard deviation 0.5. The centroid of the Gaussian was chosen to match each PMD to a particular velocity. The 17 PMDs were tuned in this way to all velocities in the interval  $[-2, +2]$  in steps of 0.25. The  $\mathbf{A}$  matrix was chosen to be a 2-dimensional Gaussian function of  $y - y'$  (i.e. the displacement across a single PMD) and  $v - v'$  (i.e. the difference in the velocity between a pair of PMDs), with a standard deviation of 3 in the position direction, and 2 in the velocity direction. There were 3 point-like objects. The object positions as a function of time were  $-t, 0, +t$ . Thus 1 object was stationary at the origin, whilst the other 2 objects start off at the origin and then move in opposite directions at unit velocity. The amplitude of each object was 2. The cluttered background was generated as a set of independently sampled uniform random numbers in the range  $[0, 2]$ . A vector  $\mathbf{q}(t)$  of likelihood functions  $Q(\mathbf{x}|y)$

## Partitioned Mixture Distributions: The Dynamical Case

was constructed at each  $t$  by adding together the above objects and clutter directly, to produce the sequence of vectors  $\mathbf{q}(t)$  shown in Figure 1. This empirical procedure for constructing the likelihood functions  $Q(\mathbf{x}|y)$  has been used for simplicity; it does not (in general) give the optimal  $Q(\mathbf{x}|y)$  that minimise the objective function in Equation 2.1.

This simulation produces the PMD posterior probability shown in Figure 2, where the objects become clearly separated in both position and velocity, and the cluttered background has been suppressed because it does not have a characteristic velocity. Note the large gain in object visibility that is achieved by processing the raw data through the set of dynamical PMDs. In effect, the set of PMDs has computed something like the Hough transform of the data in Figure 1, because it has mapped straight lines in Figure 1 into points in Figure 2.

## VI. CONCLUSIONS

In this paper the theory of static PMDs has been generalised to dynamical PMDs, and an example of how this type of network may be applied to the tracking of objects in a cluttered background has been presented. Although a comparison of this technique with other techniques has not been presented, the strong theoretical underpinning of dynamical PMDs, and also their computational simplicity, are powerful arguments for the use of this type of approach to the problem of building factorial dynamical networks in general. In particular, note that because this is an autoencoder approach it is not necessary to construct a density model in input space, which avoids many of the problems that arise in the density modelling approach.

## VII. ACKNOWLEDGEMENTS

I thank Chris Webber for many useful discussions that we had during the course of this research.

- 
- [1] A P Dempster, N M Laird, and D B Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society B **39** (1977), no. 1, 1–38.
  - [2] Z Ghahramani and M I Jordan, *Factorial hidden Markov models*, Computational Cognitive Science Report 9502, MIT, Cambridge, 1995.
  - [3] G E Hinton and R S Zemel, *Advances in neural information processing systems*, vol. 6, ch. Autoencoders, minimum description length, and Helmholtz free energy, pp. 3–10, Morgan Kaufmann, San Mateo, 1994.
  - [4] M Isard and A Blake, *Contour tracking by stochastic propagation of conditional density*, Proceedings of European conference on computer vision (Cambridge), Springer, 1996, pp. 343–356.
  - [5] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
  - [6] S P Luttrell, *Self-supervised training of hierarchical vector quantisers*, Proceedings of IEE conference on artificial neural networks (Bournemouth), IEE, 1991, pp. 5–9.
  - [7] ———, *Self-supervised adaptive networks*, Proceedings of the IEE F **139** (1992), no. 6, 371–377.
  - [8] ———, *A Bayesian analysis of self-organising maps*,

- Neural Computation **6** (1994), no. 5, 767–794.
- [9] ———, *Partitioned mixture distribution: an adaptive Bayesian network for low-level image processing*, IEE Proceedings on Vision, Image, and Signal Processing **141** (1994), no. 4, 251–260.
- [10] ———, *Handbook of neural computation*, ch. Designing analysable networks, pp. B5.3:1–B5.3:8, IOP and OUP, Oxford, 1995.
- [11] ———, *Maximum entropy and Bayesian methods*, ch. The partitioned mixture distribution: multiple overlapping density models, pp. 279–286, Kluwer, Dordrecht, 1995.
- [12] ———, *The development of dominance stripes and orientation maps in a self-organising visual cortex network (VICON)*, (1996).
- [13] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
- [14] R S Zemel, *A minimum description length framework for unsupervised learning*, Ph.D. thesis, Toronto University, 1993.



# Partitioned Mixture Distributions: A First Order Perturbation Analysis \*

S P Luttrell<sup>†</sup>

*Defence Research Agency, St Andrews Road, Malvern, Worcs, WR14 3PS, United Kingdom*

A perturbation analysis is used to linearise the theory of partitioned mixture distribution (PMD) encoder/decoder networks, which allows many connections to be made with previous results. For instance, the difference of Gaussians (DOG) filtering operation commonly used in image processing emerges very naturally, and the dependence of the wavelength of dominance stripes in a visual cortex network (VICON) may then be derived.

## I. INTRODUCTION

An introduction to partitioned mixture distributions (PMD) has been given elsewhere in these proceedings [7]. The purpose of this paper is to show how certain familiar results arise as approximations to the theory of PMD encoder/decoders, which are themselves as special case of the folded Markov chains discussed in [3].

A PMD is defined by the form of its posterior probability  $\Pr(y|\mathbf{x})$  that input vector  $\mathbf{x}$  is encoded as code index  $y$

$$\begin{aligned}\Pr(y|\mathbf{x}) &= \frac{1}{M} \sum_{y' \in \mathcal{N}^{-1}(y)} \Pr(y|\mathbf{x}; y') \\ \Pr(y|\mathbf{x}; y') &= \frac{Q(\mathbf{x}|y) \delta_{y \in \mathcal{N}(y')}}{\sum_{y'' \in \mathcal{N}(y')} Q(\mathbf{x}|y'')} \quad (1.1)\end{aligned}$$

where  $Q(\mathbf{x}|y)$  is the likelihood function for code index  $y$ ,  $\mathcal{N}(y)$  is the set of code indices that are deemed to be

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 3, 2019.

This appeared in Proceedings of the 5th International Conference on Artificial Neural Networks, Cambridge, 280-284, 1997. © 1997

“in the neighbourhood of” index  $y$ ,  $\mathcal{N}^{-1}(y)$  is the corresponding “inverse neighbourhood” defined as  $\mathcal{N}^{-1}(y) = \{y'|y \in \mathcal{N}(y')\}$ , and  $M$  is the number of mixture distributions that are embedded in the PMD.  $\Pr(y|\mathbf{x})$  satisfies the expected normalisation condition  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$ . A minor extension of the above result for  $\Pr(y|\mathbf{x})$  is to allow “leakage” of probability thus

$$\Pr(y|\mathbf{x}) \longrightarrow \sum_{y'=1}^M \Pr(y|y') \Pr(y'|\mathbf{x}) \quad (1.2)$$

where the conditional probability  $\Pr(y|y')$  redistributes the posterior probability from code index  $y'$  onto code indices  $y$ . Typically, leakage leads to topographic ordering of the entries in the codebook, in order to minimise the damaging effect that leakage has on the encoded input [3].

An objective function  $D$  for optimising a PMD is bounded above as follows [6, 7]

BRITISH CROWN COPYRIGHT/DRA.

<sup>†</sup>Electronic address: luttrell@signal.dra.hmg.gb

---


$$D \leq \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 + \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 \quad (1.3)$$


---

where  $\mathbf{x}'(y)$  is any vector function of  $y$  (which will be called a reference vector which forms the basis of a reconstruction of  $\mathbf{x}$ ), and  $n$  is the number of code indices that are simultaneously used (out of the codebook of  $M$  entries) when  $\mathbf{x}$  is encoded. This type of objective function includes the Kohonen self-organising map [1] and the LBG vector quantiser [2] as special cases [4], and has been used successfully to optimise a self-organising PMD network which models the behaviour of the mammalian visual cortex [5].

## II. TAYLOR EXPANSION OF POSTERIOR PROBABILITY

The relationship of the objective function  $D$  in Equation 1.3 to other self-organising networks is not clear, so the purpose of this section is to derive a Taylor expansion of  $D$ . It is convenient to use a matrix/vector notation for this derivation, where  $\mathbf{q}$  is the vector whose components are the  $Q(\mathbf{x}|y)$ ,  $\mathbf{p}$  is the vector whose components are  $\Pr(y|\mathbf{x})$ ,  $\mathbf{A}$  is a matrix that implements the neighbourhood summation operation  $\sum_{y' \in \mathcal{N}(y)} (\dots)$ , and  $\mathbf{A}^T$  is a matrix that implements the inverse neighbourhood summation operation  $\sum_{y' \in \mathcal{N}^{-1}(y)} (\dots)$ ; the  $\mathbf{A}$ -notation makes it easy to implement arbitrary neigh-

bourhood structures where  $\sum_{y' \in \mathcal{N}(y)}(\dots)$  is replaced by  $\sum_{y'=1}^M A_{y,y'}(\dots)$  ( $A_{y,y'}$  is the neighbourhood profile function around code index  $y$ ). Thus, before probability leakage is introduced, the PMD posterior probability  $\Pr(y|\mathbf{x})$  may be written as [7]

$$\mathbf{p} = \frac{1}{M} \mathbf{q} \mathbf{A}^T \cdot \left( \frac{1}{\mathbf{A} \cdot \mathbf{q}} \right) \quad (2.1)$$

Now write  $\mathbf{q}$  as  $\mathbf{q} = q_0 \mathbf{1} + \Delta$ , where  $q_0 \mathbf{1}$  is a convenient uniform likelihood about which to expand. The term  $\frac{1}{\mathbf{A} \cdot \mathbf{q}}$  may then be expanded as follows

$$\begin{aligned} \frac{1}{\mathbf{A} \cdot \mathbf{q}} &= \frac{1}{\mathbf{A} \cdot (q_0 \mathbf{1} + \Delta)} \\ &= \frac{1}{q_0} \frac{1}{\mathbf{A} \cdot \mathbf{1}} \left( 1 - \frac{1}{q_0} \frac{\mathbf{A} \cdot \Delta}{\mathbf{A} \cdot \mathbf{1}} \right) + \mathcal{O}(\Delta^2) \end{aligned} \quad (2.2)$$

This Taylor series is convergent for those  $\Delta$  that satisfy  $\frac{1}{q_0} \left| \frac{(\mathbf{A} \cdot \Delta)_y}{(\mathbf{A} \cdot \mathbf{1})_y} \right| < 1$  for  $y = 1, 2, \dots, M$  (i.e. which lie inside the radius of convergence of the Taylor series). This condition can readily be enforced for a network whose likelihood functions  $Q(\mathbf{x}|y)$  saturate at lower and upper bounds, such as would occur with sigmoid-shaped likelihood functions, where  $Q(\mathbf{x}|y) = \frac{1}{1 + \exp(-\mathbf{w}(y) \cdot \mathbf{x} - b(y))}$  which satisfies  $0 < Q(\mathbf{x}|y) < 1$ , in which case  $q_0 = \frac{1}{2}$  would ensure that  $|\Delta_y| < \frac{1}{2}$ ,  $\left| \frac{(\mathbf{A} \cdot \Delta)_y}{(\mathbf{A} \cdot \mathbf{1})_y} \right| < \frac{1}{2}$  (for an arbitrary neighbourhood matrix  $\mathbf{A}$ ), so  $\frac{1}{q_0} \left| \frac{(\mathbf{A} \cdot \Delta)_y}{(\mathbf{A} \cdot \mathbf{1})_y} \right| < 1$  as required.

There is an important caveat to be noted when using this approximation to  $\mathbf{p}$ . The total contribution from the higher order terms (denoted by  $\mathcal{O}(\Delta^2)$ ) is convergent inside the radius of convergence, but this does not guarantee that the  $\mathcal{O}(\Delta^2)$  terms have a negligible contribution relative to the  $\mathcal{O}(1)$  and  $\mathcal{O}(\Delta)$  terms; close to the radius of convergence the contribution from the  $\mathcal{O}(\Delta^2)$  terms will make the approximation to  $\mathbf{p}$  in Equation 2.4 inaccurate. The radius of convergence is approached when  $\Delta$  is such that  $|(\mathbf{A} \cdot \Delta)_y|$  approaches its maximum possible value, which occurs when the likelihood functions

(in the neighbourhood of  $y$ ) all approach either the lower or the upper limit of their response range. For example, networks which use sigmoid likelihood functions to minimise the upper bound on  $D$  given in Equation 1.3 cause most of the input vectors to lie in the quasi-linear region of the sigmoids, which allows the reference vectors  $\mathbf{x}'(y)$  to build an accurate reconstruction of the input vector  $\mathbf{x}$ . This quasi-linear region is not close to the radius of convergence which lies in the tails of the sigmoids, so the  $\mathcal{O}(\Delta^2)$  terms in the Taylor expansion will be small.

If the neighbourhood structure is such that  $\mathbf{A} \cdot \mathbf{1} = \alpha \mathbf{1}$  and  $\mathbf{A}^T \cdot \mathbf{1} = \alpha \mathbf{1}$  (e.g.  $\mathbf{A}$  is a symmetric shift invariant matrix), then it is convenient to introduce a normalisation factor into the definition of  $\mathbf{A}$  to enforce  $\alpha = 1$ , so that  $\mathbf{A} \cdot \mathbf{1} = \mathbf{1}$  and  $\mathbf{A}^T \cdot \mathbf{1} = \mathbf{1}$ . This does not affect the expression for the PMD posterior probability because the normalisation factors in the  $\mathbf{A}^T$  (in the numerator) and the  $\mathbf{A}$  (in the denominator) cancel out.  $\frac{1}{\mathbf{A} \cdot \mathbf{q}}$  may then be simplified to yield

$$\frac{1}{\mathbf{A} \cdot \mathbf{q}} = \frac{1}{q_0} \left( \mathbf{1} - \frac{1}{q_0} \mathbf{A} \cdot \Delta \right) + \mathcal{O}(\Delta^2) \quad (2.3)$$

whence  $\mathbf{p}$  may be expanded thus

$$\mathbf{p} = \frac{1}{M} \left( \mathbf{1} + \frac{1}{q_0} \mathbf{L}^T \cdot \left( \mathbf{1} - \mathbf{A}^T \cdot \mathbf{A} \right) \cdot \Delta \right) + \mathcal{O}(\Delta^2) \quad (2.4)$$

where probability leakage has been introduced (as in Equation 1.2) using  $L_{y,y'} \equiv \Pr(y'|y)$ .

### III. INTERPRETATION AS A DOG FILTER

The result in Equation 2.4 may be interpreted as follows. In the continuum limit (i.e.  $a \gg 1$ ,  $l \gg 1$  and  $M \gg 1$ ) let both  $\mathbf{A}$  and  $\mathbf{L}$  be Toeplitz Gaussian matrices, where  $A_{y,y'} = \frac{1}{\sqrt{2\pi}a} \exp\left(-\frac{(y-y')^2}{2a^2}\right)$  and  $L_{y,y'} = \frac{1}{\sqrt{2\pi}l} \exp\left(-\frac{(y-y')^2}{2l^2}\right)$ , whence

---


$$\left( \mathbf{L}^T \cdot \left( \mathbf{1} - \mathbf{A}^T \cdot \mathbf{A} \right) \right)_{y,y'} = \frac{1}{\sqrt{2\pi}l} \exp\left(-\frac{(y-y')^2}{2l^2}\right) - \frac{1}{\sqrt{2\pi}(2a^2+l^2)} \exp\left(-\frac{(y-y')^2}{2(2a^2+l^2)}\right) \quad (3.1)$$


---

which is a difference of Gaussians (DOG) filter, as widely used in image processing to enhance edges. Note that this result also resembles the “Mexican hat” lateral inhibition function as discussed in [1]. The matrix  $\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A})$  induces a type of lateral inhibition between the different components of  $\Delta$ . Thus the PMD posterior probabil-

ity is obtained by passing the raw likelihoods through a DOG filter (or a generalisation thereof), which suppresses featureless regions and enhances regions in which there are fluctuations. The fluctuations in  $\mathbf{p}$  (as given by Equation 2.4) are proportional to the dimensionless ratio  $\frac{\Delta}{q_0}$ , so their amplitude is proportional to the relative am-

plitude of the likelihood fluctuations (not their absolute amplitude). There is also a constant background term provided by the  $\mathbf{1}$  in Equation 2.4, which together with the overall normalisation constant  $\frac{1}{M}$  ensures that  $\mathbf{p}$  is a posterior probability that satisfies  $\sum_{y=1}^M p_y = 1$  to order  $\mathcal{O}(\Delta^2)$ . This result may be obtained from Equation 2.4 by noting that  $\sum_{y=1}^M (\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \cdot \Delta)_y = 0$  because  $\sum_{y'=1}^M L_{y,y'} = 1$  and  $\sum_{y=1}^M A_{y,y'} = 1$ .

Because  $\Delta$  appears in the expression for  $\mathbf{p}$  only in the form  $\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \cdot \Delta$ , the  $\Delta$  which causes the maximum perturbation to the  $y^{\text{th}}$  component of  $\mathbf{p}$  is proportional to the  $y^{\text{th}}$  column of the matrix  $(\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \cdot \mathbf{L}$  (this is the shape of  $\Delta$  for which the lateral inhibition function is the matched filter). Thus the presence of the lateral inhibition matrix  $\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A})$  implies that the PMD encoder/decoder network behaves as a localised factorial network, in which each component of the PMD posterior probability  $\mathbf{p}$  responds to only a limited number of neighbouring components of the vector of likelihoods  $\mathbf{q}$ . In turn, the  $y^{\text{th}}$  component of  $\mathbf{q}$  responds to the input

vector  $\mathbf{x}$  in a way that is determined by the likelihood function  $Q(\mathbf{x}|y)$ , which implies that the relationship between  $\mathbf{p}$  and  $\mathbf{x}$  is also factorial. The detailed form of  $\mathbf{A}$ ,  $\mathbf{L}$  and  $Q(\mathbf{x}|y)$  determine the precise way in which the factorial nature of this network manifests itself. For instance, if  $\mathbf{A}$  and  $\mathbf{L}$  are Gaussian (as above) and  $Q(\mathbf{x}|y)$  depends only on a limited receptive field of components of  $\mathbf{x}$ , then the factorial network is localised, in the sense that each factor depends only on a localised window of components of  $\mathbf{x}$ .

#### IV. ONE CODE INDEX

When  $n = 1$ , only one code index is used out of the codebook of  $M$  entries, and only the first term of the upper bound on the objective function  $D$  in Equation 1.3 contributes; the  $\leq$  becomes  $=$  in this limit [6]. When the expansion for the PMD posterior probability in Equation 2.4 is substituted into  $D$  it yields

$$D \approx \frac{2}{M} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \left( 1 + \frac{1}{q_0} \left( \mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \cdot \Delta \right)_y \right) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \quad (4.1)$$

where terms of order  $\mathcal{O}(\Delta^2)$  have been omitted. In order to develop a gradient descent training algorithm for the  $\Delta$  and  $\mathbf{x}'(y)$  parameters, differentiate  $D$  with respect to its parameters to obtain

$$\begin{aligned} \frac{\partial D}{\partial \mathbf{x}'(y)} &\approx -\frac{4}{M} \int d\mathbf{x} \Pr(\mathbf{x}) \left( 1 + \frac{1}{q_0} \left( \mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \cdot \Delta \right)_y \right) (\mathbf{x} - \mathbf{x}'(y)) \\ \frac{\partial D}{\partial \Delta_y} &\approx \frac{2}{M q_0} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y'=1}^M \left( \mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \right)_{y',y} \|\mathbf{x} - \mathbf{x}'(y')\|^2 \end{aligned} \quad (4.2)$$

For each vector  $\mathbf{x}$  in the training set (which is described by  $\Pr(\mathbf{x})$ ), a gradient descent algorithm requires that the parameter update is in the opposite direction to the gradient of  $D$  with respect to that parameter.

The  $\mathbf{x}'(y)$  parameters (for  $y = 1, 2, \dots, M$ ) are updated towards  $\mathbf{x}$  by an amount that is proportional to their differences  $\mathbf{x} - \mathbf{x}'(y)$ , and proportional to the weighting factors  $1 + \frac{1}{q_0} (\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \cdot \Delta)_y$ . As discussed earlier,  $\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A})$  is a lateral inhibition matrix, which typically has the form of a DOG filter, so the above weighting factor causes the  $\mathbf{x}'(y)$  updates to be weighted by an amount that depends on how strong the likelihood fluctuations are in the neighbourhood of  $y$ . In the extreme case where one of the components of  $\Delta$  is much larger than the others (within the limitations imposed by the use of a Taylor expansion which ignores terms of order  $\mathcal{O}(\Delta^2)$ ), then the weighting factor is approximately  $1 + \frac{\text{DOG}}{q_0}$  with the DOG filter centred on the

large component of  $\Delta$ , otherwise the weighting factor is  $1 + \frac{\text{linear combination of DOG}}{q_0}$ . The limiting case resembles the update prescription that is used in a Kohonen network [1].

The  $\Delta_y$  parameters (for  $y = 1, 2, \dots, M$ ) are updated by an amount that is proportional to  $-\sum_{y'=1}^M (\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}))_{y',y} \|\mathbf{x} - \mathbf{x}'(y')\|^2$ . Now  $\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A})$  is a lateral inhibition matrix, which typically has the form of a DOG filter, so  $\Delta_y \leq 0$  if the vector of Euclidean reconstruction errors ( $\|\mathbf{x} - \mathbf{x}'(y=1)\|^2, \|\mathbf{x} - \mathbf{x}'(y=2)\|^2, \dots$ ) has a positive (or zero) projection onto the DOG filter (or its generalisation). Thus  $\Delta_y \leq 0$  in the vicinity of a localised peak of Euclidean reconstruction errors, and vice versa for a localised trough. This prescription for updating the likelihood functions thus relocates posterior probability onto code indices that have smaller than the local average error value, as is intuitively to be expected.

There is a Hebbian interpretation of the  $\mathbf{x}'(y)$  update prescription, because the update step size is proportional to the product of two quantities: (1) A vector of likelihoods  $\mathbf{q}$  (filtered through the lateral inhibition matrix  $\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A})$ ) which lives in output space, and (2) A reconstruction error vector  $\mathbf{x} - \mathbf{x}'(y)$  which lives in input space. This product of input space and output space quantities is characteristic of a Hebbian update prescription.

In order to demonstrate the usefulness of this  $n = 1$  perturbation expansion, an estimate of the wavenumber of dominance stripes (as reported in [5] may be obtained by using  $\mathbf{x} = (x, x, \dots, x)$  (i.e. all components of the input vector are constrained to be equal, as in the dominance stripe simulations in [5]), setting  $\frac{\Delta_y}{q_0} = \frac{x}{x_0} \sin(ky)$  in the expression for  $D$  in Equation 4.1 (which is an appropriate parametric form for  $\frac{\Delta_y}{q_0}$  when the training conditions lead to dominance stripes), using the continuum limit DOG form for  $\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A})$  given in Equation 3.1, minimising  $D$  first with respect to  $\mathbf{x}'(y)$ , and then with respect to  $k$ . The optimum wavenumber  $k_0$  is then given

by

$$k_0 = \frac{1}{a} \sqrt{\log\left(\frac{2a^2}{l^2} + 1\right)} \quad (4.3)$$

which shows that  $k_0 \propto \frac{1}{a}$  times a logarithmic correction factor, as would be expected if each lateral inhibition length (which is proportional to  $a$ ) is to contain a complete set of computing machinery for processing the input vector [5]. This result breaks down as  $l \rightarrow 0$  because the continuum limit assumption is then violated. This optimum wavenumber result is in agreement with the results of dominance stripe simulations reported in [5].

## V. MANY CODE INDICES

When  $n \rightarrow \infty$  only the second term of the objective function  $D$  in Equation 1.3 contributes. When the expansion for the PMD posterior probability in Equation 2.4 is substituted into the upper bound for  $D$  it yields

---


$$D \leq 2 \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \frac{1}{M} \sum_{y=1}^M \left( 1 + \frac{1}{q_0} \left( \mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \cdot \boldsymbol{\Delta} \right)_y \right) \mathbf{x}'(y) \right\|^2 \quad (5.1)$$

where terms of order  $\mathcal{O}(\boldsymbol{\Delta}^2)$  have been omitted. Define the residual vector

$$\mathbf{d} \equiv \mathbf{x} - \frac{1}{M} \sum_{y'=1}^M \left( 1 + \frac{1}{q_0} \left( \mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \cdot \boldsymbol{\Delta} \right)_{y'} \right) \mathbf{x}'(y') \quad (5.2)$$

whence the derivatives  $D$  are

$$\begin{aligned} \frac{\partial D}{\partial \mathbf{x}'(y)} &\approx -\frac{4}{M} \int d\mathbf{x} \Pr(\mathbf{x}) \left( 1 + \frac{1}{q_0} \left( \mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \cdot \boldsymbol{\Delta} \right)_y \right) \mathbf{d} \\ \frac{\partial D}{\partial \Delta_y} &\approx -\frac{4}{M q_0} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y'=1}^M \left( \mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \right)_{y',y} (\mathbf{x}'(y') \cdot \mathbf{d}) \end{aligned} \quad (5.3)$$


---

where the symbol  $D$  now stands for the right hand side of Equation 5.1, and the expression for  $\frac{\partial D}{\partial \mathbf{x}'(y)}$  has some  $\mathcal{O}(\boldsymbol{\Delta}^2)$  terms left in it for convenience.

The  $\mathbf{x}'(y)$  parameters are updated in a similar way to the  $n = 1$  case, except that on the right hand side the full residual vector  $\mathbf{d}$  is used rather than the partial residual vector  $\mathbf{x} - \mathbf{x}'(y)$ .

The  $\Delta_y$  parameters are updated in a similar way to the  $n = 1$  case, except that on the right hand side  $-\mathbf{x}'(y') \cdot \mathbf{d}$  is used rather than the partial Euclidean error  $\|\mathbf{x} - \mathbf{x}'(y')\|^2$ . By analogy with the  $n = 1$  case,  $\Delta_y \geq 0$  in the vicinity of a localised peak of  $\mathbf{x}'(y') \cdot \mathbf{d}$ . This prescription for updating the likelihood functions thus relocates

posterior probability onto code indices that have a larger than local average value of  $\mathbf{x}'(y') \cdot \mathbf{d}$ , which are precisely the ones than can most reduce the residual vector, as is intuitively to be expected.

There is a radial basis function (RBF) network interpretation of the expression for (the upper bound on)  $D$  in Equation 5.1 when the likelihood functions have the radial form  $Q(\mathbf{x}|y) = Q(\|\mathbf{x} - \mathbf{x}_0(y)\|^2)$ . The vector of outputs from the “radial basis functions” is given by  $\frac{1}{M} \left( \mathbf{1} + \frac{1}{q_0} (\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A}) \cdot \boldsymbol{\Delta}) \right)$ , where the non-linearities are contained in the likelihood functions that produce  $\boldsymbol{\Delta}$  from the input vector  $\mathbf{x}$ , and  $\boldsymbol{\Delta}$  is further processed linearly through the lateral inhibition matrix  $\mathbf{L}^T \cdot (\mathbf{1} - \mathbf{A}^T \cdot \mathbf{A})$

to produce (modulo multiplicative and additive constants) the hidden layer activity vector. Next, the output layer activity vector is produced from the hidden layer activity vector by a linear operation, where the weight vector fanning out from the  $y^{\text{th}}$  hidden layer unit is  $\mathbf{x}'(y)$ . Finally, an objective function is formed that measures (twice) the average Euclidean error between this output and a target output (which is the original input, in this case).

## VI. CONCLUSIONS

In this paper an objective function for optimising a partitioned mixture distribution (PMD) encoder/decoder has been approximated in order to simplify the compli-

cated expression for the PMD posterior probability. It has been shown how the outputs of the PMD laterally inhibit each other, which may be modelled as arising from a difference of Gaussians (DOG) operation on the input, and which closely resembles the ‘Mexican hat function’ used by Kohonen. These results (and others derived in the paper) show how some familiar operations may be viewed as special cases of the computations that are performed in a PMD.

## VII. ACKNOWLEDGEMENTS

I thank Chris Webber for many useful discussions that we had during the course of this research.

- 
- [1] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
  - [2] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
  - [3] S P Luttrell, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
  - [4] ———, *Handbook of neural computation*, ch. Designing analysable networks, pp. B5.3:1–B5.3:8, IOP and OUP, Oxford, 1995.
  - [5] ———, *The development of dominance stripes and orientation maps in a self-organising visual cortex network (VICON)*, (1996).
  - [6] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
  - [7] ———, *Partitioned mixture distributions: the dynamical case*, Proceedings of IEE international conference on artificial neural networks (Cambridge), IEE, 1997, pp. 59–63.



# Optimal Response Functions in a Network of Discretely Firing Neurons \*

S P Luttrell<sup>†</sup>

Defence Research Agency, St Andrews Road, Malvern, Worcs, WR14 3PS, United Kingdom

An objective function is defined which models an unsupervised neural network of discretely firing neurons. The objective function can be explicitly optimised with respect to the choice of probabilistic model that describes the neuron properties; this optimisation is done explicitly (i.e. algebraically) in some special cases.

## I. INTRODUCTION

In an earlier paper (Luttrell [3]) a Bayesian analysis of unsupervised neural networks was presented. The approach adopted was to model the network as an encoder/decoder pair, where the input vector was encoded probabilistically by the firing pattern of the output neurons, and then Bayes' theorem was used to deduce the inverse operation (i.e. the decoding process) which allowed a reconstruction of the input vector to be constructed. This type of probabilistic encoding process describes the behaviour of a set of discretely firing output neurons, where the encoded version of the input is the location in the output layer of the neuron that fires first in response to presentation of the input vector. A natural generalisation of this is to consider an encoded version of the input which specifies the set of locations of the first  $n$  (out of a total of  $M$ ) neurons that fire (Luttrell [4, 5]).

In this approach, the behaviour of the network is described completely by the joint probabilistic behaviour of the discretely firing output neurons, so optimisation of the network corresponds to choosing an appropriate form for this joint probability. Vector quantisers (Linde *et al* [2]) and Kohonen maps (Kohonen [1]) emerge as optimal special cases when  $n = 1$  (Luttrell [3]). The purpose of this paper is to explore other special cases.

In Section II a brief résumé of the underlying theory

\*Typeset in LATEX on May 22, 2019.

This appeared in Proceedings of the 5th International Conference on Artificial Neural Networks, Cambridge, 216-220, 1997. © 1997

(Luttrell [4, 5]) is presented, and in Section III a derivation of the vector quantiser result for  $n = 1$  and arbitrary  $M$  (Luttrell [3]) is presented. Section IV contains new results which are concerned with the special case of optimising a network of 2 neurons under various conditions.

## II. GENERAL THEORY

A network objective function  $D$  is defined for a 2-layer neural network as (Luttrell [3][4, 5])

$$D \equiv \sum_{y_1, y_2, \dots, y_n=1}^M \int d\mathbf{x} d\mathbf{x}' \Pr(\mathbf{x}) \Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x}'|\mathbf{y}) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (2.1)$$

where  $\mathbf{x}$  is the vector of input activities,  $\mathbf{y} = (y_1, y_2, \dots, y_n)$  is the vector of locations ( $1 \leq y_i \leq M$ ) of  $n$  firing events in the output layer,  $\mathbf{x}'$  is a reconstruction of the input activity from the firing events  $\mathbf{y}$ ,  $\Pr(\mathbf{x})$  is the PDF of input activity vectors,  $\Pr(\mathbf{y}|\mathbf{x})$  is the probability that  $\mathbf{y}$  occurs given that  $\mathbf{x}$  is known, and  $\Pr(\mathbf{x}'|\mathbf{y})$  is the corresponding Bayes' inverse PDF  $\Pr(\mathbf{x}'|\mathbf{y}) = \frac{\Pr(\mathbf{y}|\mathbf{x}') \Pr(\mathbf{x}')}{\int d\mathbf{x}'' \Pr(\mathbf{y}|\mathbf{x}'') \Pr(\mathbf{x}'')}$ . Make the independence assumption  $\Pr(\mathbf{y}|\mathbf{x}) = \Pr(y_1|\mathbf{x}) \Pr(y_2|\mathbf{x}) \dots \Pr(y_n|\mathbf{x})$  and express  $\Pr(\mathbf{x}'|\mathbf{y})$  using Bayes' theorem to obtain  $D$  in the form

BRITISH CROWN COPYRIGHT/DRA.

†Electronic address: luttrell@signal.dra.hmg.gb

---


$$D = \sum_{y_1, y_2, \dots, y_n=1}^M \int d\mathbf{x} d\mathbf{x}' \Pr(\mathbf{x}) \Pr(\mathbf{x}') \frac{[\Pr(y_1|\mathbf{x}) \dots \Pr(y_n|\mathbf{x})] [\text{ditto } \mathbf{x} \rightarrow \mathbf{x}']}{\int d\mathbf{x}'' [\text{ditto } \mathbf{x} \rightarrow \mathbf{x}''] \Pr(\mathbf{x}'')} \|\mathbf{x} - \mathbf{x}'\|^2 \quad (2.2)$$

which must be minimised with respect to the choice of  $\Pr(y_i|\mathbf{x})$ , whilst ensuring that  $\sum_{y_i} \Pr(y_i|\mathbf{x}) = 1$  (for  $i = 1, 2, \dots, n$ ) by using a Lagrange function  $L$  defined as  $L \equiv \sum_{i=1}^n \int d\mathbf{x} \lambda_i(\mathbf{x}) \sum_{y_i} \Pr(y_i|\mathbf{x})$ . Logarithmically differentiate  $D - L$  with respect to  $\Pr(y_i|\mathbf{x})$  to obtain

$$\frac{\delta(D - L)}{\delta \log \Pr(y_i|\mathbf{z})} = \Pr(y_i|\mathbf{z}) \sum_{\mathbf{y}/y_i} \frac{\Pr(\mathbf{y}|\mathbf{z})}{\Pr(y_i|\mathbf{z})} \left( 2 \Pr(\mathbf{z}) \left\| \mathbf{z} - \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x} \right\|^2 - \lambda_i(\mathbf{z}) \right) \quad (2.3)$$

whence the stationarity condition  $\frac{\delta(D - L)}{\delta \log \Pr(y_i|\mathbf{z})} = 0$  yields (for  $i = 1, 2, \dots, n$ ) either  $\Pr(y_i|\mathbf{z}) = 0$  or

$$\lambda_i(\mathbf{z}) = 2 \Pr(\mathbf{z}) \sum_{y_1, y_2, \dots, y_n/y_i} \frac{\Pr(y_1|\mathbf{z}) \dots \Pr(y_n|\mathbf{z})}{\Pr(y_i|\mathbf{z})} \left\| \mathbf{z} - \frac{\int d\mathbf{x} \Pr(y_1|\mathbf{x}) \dots \Pr(y_n|\mathbf{x}) \Pr(\mathbf{x}) \mathbf{x}}{\int d\mathbf{x} \Pr(y_1|\mathbf{x}) \dots \Pr(y_n|\mathbf{x}) \Pr(\mathbf{x})} \right\|^2 \quad (2.4)$$

where  $\sum_{y_1, y_2, \dots, y_n / y_i} (\dots)$  denotes summation over all components of  $\mathbf{y}$  *except*  $y_i$ . The rest of this paper will focus on solving this stationarity condition for a variety of values of  $M$  (number of neurons) and  $n$  (number of firing events).

### III. $M = \text{ANYTHING}$

Set  $n = 1$  in the stationarity condition in Equation 2.4 to obtain (Luttrell [3]) either  $\Pr(y|\mathbf{z})$  or

$$\lambda(\mathbf{z}) = 2\Pr(\mathbf{z}) \left( \left\| \mathbf{z} - \frac{\int d\mathbf{x} \Pr(y|\mathbf{x}) \Pr(\mathbf{x}) \mathbf{x}}{\int d\mathbf{x} \Pr(y|\mathbf{x}) \Pr(\mathbf{x})} \right\|^2 \right) \quad (3.1)$$

The solution is  $\Pr(y|\mathbf{x}) = \delta_{y,y(\mathbf{x})}$  (i.e. a Kronecker delta function), because if  $y \neq y(\mathbf{z})$  then  $\Pr(y|\mathbf{z}) = 0$  which satisfies the first stationarity condition, and if  $y = y(\mathbf{z})$  then  $\lambda(\mathbf{z}) = 2\Pr(\mathbf{z}) \left( \left\| \mathbf{z} - \frac{\int_{\mathbf{x}:y=y(\mathbf{x})} d\mathbf{x} \Pr(\mathbf{x}) \mathbf{x}}{\int_{\mathbf{x}:y=y(\mathbf{x})} d\mathbf{x} \Pr(\mathbf{x})} \right\|^2 \right)$  which satisfies the second stationarity condition. For this to be a consistent solution it is important that different values of  $y$  correspond to non-overlapping regions of  $\mathbf{z}$ -space, which is indeed the case for  $\Pr(y|\mathbf{z}) = \delta_{y,y(\mathbf{z})}$ . It remains to optimise the choice of mapping function  $y(\mathbf{z})$  which reduces to a standard LBG vector quantiser optimisation process (Linde *et al*, 1980).

### IV. $M = 2$

In this section a network of 2 neurons is optimised: Section IV A deals with the case of 2 firing events, Section IV B deals with the case of 3 firing events.

Choose  $M=2$  so that  $y$  can take only 2 possible values which may be chosen to be  $y = 0$  and  $y = 1$ , so  $\Pr(y|\mathbf{x})$  can be written as  $\Pr(y|\mathbf{x}) = y p(\mathbf{x}) + (1 - y)(1 - p(\mathbf{x}))$ , where  $p(\mathbf{x})$  is the posterior probability associated with the  $y = 1$  state, and  $1 - p(\mathbf{x})$  is the posterior probability associated with the  $y = 0$  state. Because  $y$  is now a binary variable, the summation  $\sum_{y_1, y_2, \dots, y_n / y_i}$  in Equation 2.4 sums over all states of an  $n - 1$  bit binary vector, and each term in the sum can be simplified because  $\frac{\Pr(y_1|\mathbf{z}) \Pr(y_2|\mathbf{z}) \dots \Pr(y_n|\mathbf{z})}{\Pr(y_i|\mathbf{z})}$  reduces to  $p(\mathbf{z})^m (1 - p(\mathbf{z}))^{n-m-1}$  when  $m$  of the bits are set to 1 (and thus  $n - m - 1$  bits are set to zero). If  $\mathbf{z}(i, j)$  is defined as  $\mathbf{z}(i, j) \equiv \frac{\int d\mathbf{x} p(\mathbf{x})^i (1-p(\mathbf{x}))^j \Pr(\mathbf{x}) \mathbf{x}}{\int d\mathbf{x} p(\mathbf{x})^i (1-p(\mathbf{x}))^j \Pr(\mathbf{x})}$  then the stationarity condition in Equation 2.4 may be simplified to yield either  $\begin{pmatrix} p(\mathbf{z}) \\ 1 - p(\mathbf{z}) \end{pmatrix} = 0$  or

---


$$\lambda_i(\mathbf{z}) = 2\Pr(\mathbf{z}) \sum_{m=0}^{n-1} \frac{(n-1)!}{(n-m-1)!m!} p(\mathbf{z})^m (1 - p(\mathbf{z}))^{n-m-1} \left\| \mathbf{z} - \begin{pmatrix} \mathbf{z}(m+1, n-m-1) \\ \mathbf{z}(m, n-m) \end{pmatrix} \right\|^2 \quad (4.1)$$

where the upper (lower) component of the column vectors corresponds to the  $y_i = 1$  ( $y_i = 0$ ) case. This result holds for any number  $n$  of firing events when there are 2 neurons (i.e.  $M = 2$ ). The following two subsections will focus on explicit results for  $n = 2$  and  $n = 3$ .

#### A. $M = 2, n = 2$

When  $n = 2$  there are only 2 terms that contribute to the stationarity condition in Equation 4.1, which thus reduces to either  $\begin{pmatrix} p(\mathbf{z}) \\ 1 - p(\mathbf{z}) \end{pmatrix} = 0$  or

$$\lambda_i(\mathbf{z}) = 2\Pr(\mathbf{z}) (1 - p(\mathbf{z})) \left\| \mathbf{z} - \begin{pmatrix} \mathbf{z}(1, 1) \\ \mathbf{z}(0, 2) \end{pmatrix} \right\|^2 + 2\Pr(\mathbf{z}) p(\mathbf{z}) \left\| \mathbf{z} - \begin{pmatrix} \mathbf{z}(2, 0) \\ \mathbf{z}(1, 1) \end{pmatrix} \right\|^2 \quad (4.2)$$


---

Assume that  $\Pr(\mathbf{x})$  factorises as  $\Pr(\mathbf{x}) = \Pr(x_1) \Pr(x_2) \dots \Pr(x_{\dim \mathbf{x}})$  where the various  $\Pr(x_i)$  are in general different PDFs (this readily generalises to any PDF which may be factorised by rotating the basis vectors by an appropriate amount), and try a solution

$p(\mathbf{x})$  of the form

$$p(\mathbf{x}) = \begin{cases} 0 & -\infty < x_1 \leq -s \\ \frac{1}{2} + \frac{x_1}{2s} & -s \leq x_1 \leq s \\ 1 & s \leq x_1 < \infty \end{cases} \quad (4.3)$$

whose form is justified by hindsight (i.e. it produces a consistent stationary solution). Note that there are

actually  $\dim \mathbf{x}$  alternative possible stationary solutions, each of which is aligned with one of the  $\dim \mathbf{x}$  input axes, and the correct solution should be selected as the one that produces the smallest value of  $D$ . Define  $I_k(a, b) \equiv \int_a^b dx_1 x_1^k \Pr(x_1)$  which may be eval-

uated once  $\Pr(x_1)$  is specified explicitly, and assume that  $\Pr(x_i) = \Pr(-x_i)$  for each  $i$ , so that  $I_k(-a, a) = 0$  for  $k$  odd, and  $I_k(-b, -a) = (-1)^k I_k(a, b)$ . Thus

---


$$\begin{aligned} \int d\mathbf{x} (1 - p(\mathbf{x})) p(\mathbf{x}) \Pr(\mathbf{x}) &= \frac{1}{4} I_0(-s, s) - \frac{1}{4s^2} I_2(-s, s) \\ \int d\mathbf{x} (1 - p(\mathbf{x})) p(\mathbf{x}) \Pr(\mathbf{x}) x_i &= 0 \\ \int d\mathbf{x} (1 - p(\mathbf{x}))^2 \Pr(\mathbf{x}) &= I_0(s, \infty) + \frac{1}{4} I_0(-s, s) + \frac{1}{4s} I_2(-s, s) \\ \int d\mathbf{x} (1 - p(\mathbf{x}))^2 \Pr(\mathbf{x}) x_i &= \begin{cases} -I_1(s, \infty) - \frac{1}{2s} I_2(-s, s) & i = 1 \\ 0 & i = 2, 3, \dots, \dim \mathbf{x} \end{cases} \\ \int d\mathbf{x} p(\mathbf{x})^2 \Pr(\mathbf{x}) &= I_0(s, \infty) + \frac{1}{4} I_0(-s, s) + \frac{1}{4s^2} I_2(-s, s) \\ \int d\mathbf{x} p(\mathbf{x})^2 \Pr(\mathbf{x}) x_i &= \begin{cases} I_1(s, \infty) + \frac{1}{2s} I_2(-s, s) & i = 1 \\ 0 & i = 2, 3, \dots, \dim \mathbf{x} \end{cases} \end{aligned} \quad (4.4)$$

and define the shorthand notation

$$a(s) \equiv \frac{I_1(s, \infty) + \frac{1}{2s} I_2(-s, s)}{I_0(s, \infty) + \frac{1}{4} I_0(-s, s) + \frac{1}{4s^2} I_2(-s, s)} \quad (4.5)$$

for a commonly occurring combination of these integrals. Now use these results in the stationarity condition in Equation 4.2 to obtain 2 alternative stationarity conditions (for  $y_i = 1$  and  $y_i = 0$ , respectively) in the interval  $-s \leq z_1 \leq s$  (assuming that the appropriate component of  $\begin{pmatrix} p(\mathbf{z}) \\ 1 - p(\mathbf{z}) \end{pmatrix}$  is non-zero, and that  $\Pr(\mathbf{z}) \neq 0$ )

$$\frac{\lambda_i(\mathbf{z})}{2\Pr(\mathbf{z})} = \begin{cases} \left(\frac{1}{2} - \frac{z_1}{2s}\right) \left(z_1^2 + \sum_{i=2}^{\dim \mathbf{x}} z_i^2\right) + \left(\frac{1}{2} + \frac{z_1}{2s}\right) \left((z_1 - a(s))^2 + \sum_{i=2}^{\dim \mathbf{x}} z_i^2\right) & y_i = 1 \\ \left(\frac{1}{2} + \frac{z_1}{2s}\right) \sum_{i=2}^{\dim \mathbf{x}} z_i^2 + \left(\frac{1}{2} - \frac{z_1}{2s}\right) \left((z_1 + a(s))^2 + \sum_{i=2}^{\dim \mathbf{x}} z_i^2\right) & y_i = 0 \end{cases} \quad (4.6)$$


---

These 2 conditions (for  $y_i = 1$  and  $y_i = 0$ , respectively) must produce the same  $\lambda_i(\mathbf{z})$ , which imposes the following consistency condition which the stationary solution must obey

$$z_1 a(s) \left( \frac{1}{s} a(s) - 2 \right) = 0 \quad (4.7)$$

This consistency equation will now be solved for the cases of uniform and Gaussian input PDFs.

If the input is scalar and its PDF is uniform, then  $\Pr(z_1) = \frac{1}{2}$  in  $-1 \leq z_1 \leq 1$ ,  $I_0(-s, s) = s$ ,  $I_0(s, \infty) = \frac{1}{2} - \frac{1}{2}s$ ,  $I_1(s, \infty) = \frac{1}{4} - \frac{1}{4}s^2$ ,  $I_2(-s, s) = \frac{1}{3}s^3$ , whence the consistency condition in Equation 4.7 becomes  $z_1(s^2 - 4s + 1) \frac{s^2 - 3}{s(s-3)^2} = 0$ . The only valid (i.e.  $0 \leq s \leq 1$ ) solution is  $s = 2 - \sqrt{3} \approx 0.26795$ . This result models a soft scalar quantiser in which the 2 quantisation cells

overlap each other (see Equation 4.3) with a pair of linear ramp functions in the region  $-s \leq z \leq s$ , but the regions  $-1 \leq z \leq -s$  and  $s \leq z \leq 1$  are assigned uniquely to one or other quantisation cell.

If the input is scalar and its PDF is Gaussian, then  $\Pr(z_1) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{z_1^2}{2\sigma^2}\right)$ , and using that  $\text{erf}(x) \equiv \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$  leads to  $I_0(-s, s) = \text{erf}\left(\frac{s}{\sqrt{2}\sigma}\right)$ ,  $I_0(s, \infty) = \frac{1}{2} \left(1 - \text{erf}\left(\frac{s}{\sqrt{2}\sigma}\right)\right)$ ,  $I_1(s, \infty) = \frac{\sigma}{\sqrt{2\pi}} e^{-\frac{1}{2}\frac{s^2}{\sigma^2}}$ ,  $I_2(-s, s) = \sigma^2 \text{erf}\left(\frac{s}{\sqrt{2}\sigma}\right) - 2s \frac{\sigma}{\sqrt{2\pi}} e^{-\frac{1}{2}\frac{s^2}{\sigma^2}}$ , whence the consistency condition in Equation 4.7 becomes (where only the relevant factor is retained)  $\frac{1}{\sqrt{2\pi}} \frac{\sigma}{s} e^{-\frac{1}{2}\frac{s^2}{\sigma^2}} + \frac{1}{2} \text{erf}\left(\frac{1}{\sqrt{2}} \frac{s}{\sigma}\right) = 1$ , which can be solved numerically to yield  $\frac{s}{\sigma} \approx 0.43633$ , which is a soft vector quantiser with linear ramps that overlap by a different amount than in the uniform input

PDF case above.

In the general vector input case, where  $\Pr(\mathbf{x}) = \Pr(x_1) \Pr(x_2) \cdots \Pr(x_{\dim \mathbf{x}})$ , there are  $\dim \mathbf{x}$  stationary solutions each of which is aligned with one of the  $\dim \mathbf{x}$  input axes, and the optimal solution is the one that minimises  $D$ . In both the uniform and Gaussian PDF cases the solution is the same as in the scalar case above, but aligned with the input axis that has the greatest input variance. Thus the 2 neurons act as a soft vector quantiser that encodes the component of the input that has the greatest variance, whilst ignoring the other components of the input.

### B. $M = 2, n = 3^1$

When  $n = 3$  there are 3 terms that contribute to the stationarity condition in Equation 4.1, but otherwise the solution is found in the same way as for the  $n = 2$  case above, i.e. the first step is to impose the condition that the stationarity conditions for the  $y_i = 1$  and  $y_i = 0$  cases are consistent where they overlap. Thus consider

<sup>1</sup> This section has been rewritten to correct an error in the derivation of the optimal  $p(x)$ ; the functional form of the result remains unchanged.

the scalar input case with uniform input PDF and try a solution of the form (where  $f(x) = -f(-x)$ )

$$p(x) = \begin{cases} 0 & -\infty < x \leq -s \\ \frac{1}{2} + f(x) & -s \leq x \leq s \\ 1 & s \leq x < \infty \end{cases} \quad (4.8)$$

whose form is justified in hindsight as in the  $n = 2$  case above. Define the following notation:  $a \equiv \int_{-s}^{+s} dx f(x) x$ ,  $b \equiv \int_{-s}^{+s} dx f(x)^2$ ,  $c \equiv \int_{-s}^{+s} dx f(x)^3 x$ ,  $d \equiv \frac{a-4c}{2b-s}$ ,  $e \equiv \frac{2+3a+4c-2s^2}{4+6b-3s}$ , whence the consistency condition in the interval  $-s \leq x \leq s$  is the integral equation

$$2(6d+2e)x f(x)^2 + 2(d^2-e^2)f(x) + (e-d)x = 0 \quad (4.9)$$

The value of  $s$  may be determined by demanding that  $p(x)$  is continuous at  $x = s$ , which requires that  $f(s) = \frac{1}{2}$ ; inserting these values into the above equation gives  $s = \frac{e-d}{2}$ . The consistency condition is quadratic in  $f(x)$  so it may be solved to yield

unchanged.

$$p(x) = \begin{cases} 0 & -\infty < x \leq -s \\ \frac{1}{2} + \frac{1}{4x} \frac{(d+e)s}{(s-(d+e))} \left( \sqrt{1 + 4 \frac{x^2(s-(d+e))}{(d+e)^2 s}} - 1 \right) & -s \leq x \leq s \\ 1 & s \leq x < \infty \end{cases} \quad (4.10)$$

where there is 1 remaining parameter  $d + e$  that remains to be determined to satisfy the consistency condition. The numerical solution is  $d = -0.058416$ ,  $e = 0.58366$ , which gives  $s = 0.32104$ .

It is instructive to plot  $p(x)$  in Figure 1. As in the  $n = 2$  case, near each boundary there is a region where  $p(x)$  is constant ( $p(x) = 0$  near  $x = -1$  and  $p(x) = 1$  near  $x = 1$ ). However, unlike the  $n = 2$  case, the overlap region  $-s \leq x \leq s$  has a pair of opposing *non-linear* functions (i.e. not the *linear* ramps that arose in the  $n=2$  case). The shape of these non-linear functions is such that for 3 firing events they lead to the minimum average Euclidean error in the reconstruction of the input.

In the general case where there are  $n$  firing events the consistency condition for the  $y_i = 1$  and  $y_i = 0$  stationarity conditions is a polynomial of order  $n - 1$  in  $p(\mathbf{z})$ ; this can be solved in closed form only for  $n \leq 5$ .

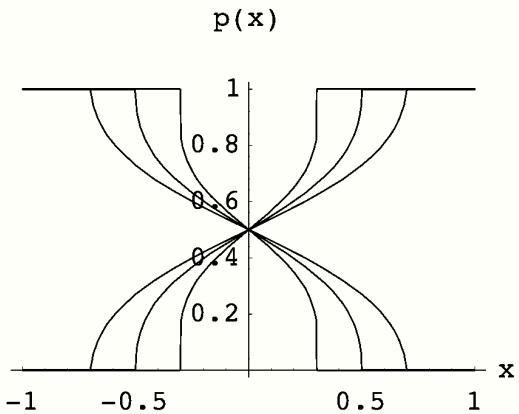


Figure 1: Plot of  $p(x)$  and  $p(-x)$ . Note that  $p(x) = 0$  for  $-1 \leq x \leq s$ , and  $p(x) = 1$  for  $s \leq x \leq 1$ . Taken together,  $p(x)$  and  $p(-x)$  constitute a pair of back-to-back neuron responses, where  $p(x)$  switches on for large positive input values,  $p(-x)$  switches on for large negative input values, and both respond partially for intermediate input values.

## V. CONCLUSIONS

Optimal neuron response functions have been derived for some special cases in a network of discretely firing neurons. The most natural way of expressing these results is in the form of the posterior probability (given the input vector) for a neuron to produce the next firing event that occurs in the network. In the special case where the input vector has statistically independent components and the network has just 2 neurons, the form of the optimal posterior probability for one of the neurons (the other has a mirror image result) is a linear ramp truncated to the range [0, 1] when 2 firing events are observed, and the optimal posterior probability is as shown in Figure 1 when 3 firing events are observed. Quite

generally, the 2 neurons work in a back-to-back fashion, with one of them switching fully on when the input is large and negative, and the other one fully on when the input is large and positive, with both of them partially switching on over an intermediate range of input values. If the properties of the input PDF are such that it may be rotated into a basis where the PDF factorises (as was assumed from Section IV A onwards), then the above results still hold true, but in the rotated basis. For instance, if the input is a set of correlated Gaussian variables, whose covariance matrix happens to have a principal eigenvector that resembles a centre-on/surround-off function, then one of the 2 back-to-back neurons will act as if it had a centre-on/surround-off response, and the other a centre-off/surround-on response.

- [1] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [2] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [3] S P Luttrell, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
- [4] ———, *Handbook of neural computation*, ch. Designing analysable networks, pp. B5.3:1–B5.3:8, IOP and OUP, Oxford, 1995.
- [5] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.



# Self-Organised Modular Neural Networks for Encoding Data \*

Stephen P. Luttrell

*Defence Evaluation and Research Agency, St Andrews Rd, Malvern, Worcs, WR14 3PS, U.K.*

It is shown how a neural network can be optimised so that multiple interlinked network modules emerge by self-organisation. The processing task chosen to illustrate this is encoding high-dimensional data, such as images, where multiple network modules implement a factorial encoder, in which the high-dimensional data space is broken up into a number of low-dimensional subspaces, each of which is separately encoded. This type of factorial encoder emerges through a process of self-organisation, provided that the input data lies on a curved manifold, as is indeed the case in image processing applications.

## I. INTRODUCTION

The problem of using multiple neural networks, and combining their results to achieve a goal, is difficult to analyse in general. The general problem is how best to make use of several network modules, with communication links between the modules to allow their mutual cooperation, to obtain a better performance than any single module could achieve alone. The work that is described in this chapter focuses on self-organised network behaviour, in which various interlinked network modules emerge spontaneously. The ultimate goal of this research is for all network behaviour to emerge by a process of self-organisation.

### A. An Image Processing Problem

In order to focus our attention it is necessary to identify a specific problem that is both easy to state, and which leads to the required self-organised formation of cooperating network modules. An ideal problem is the encoding of high-dimensional data, such as might be derived from an imaging sensor. Typically, an image contains a large number of constituents, which are spatially arranged in a highly structured way in order to construct a meaningful image. The problem is to characterise this structure in a succinct way. For instance, if the image consists of a single target placed at random, then the simplest code would merely record the coordinates of the target. This could be extended to an image of multiple targets placed at random, where the simplest code would be the coordinates of all of the targets.

However, the structure of real-world images is much more complicated than this. There may not be, and usually is not, a useful decomposition of the image into a superposition of images of separate targets. This type of encoding problem can in principle be solved by training a suitable adaptive network on many images, which

would then allow the network both to learn the image constituents, and simultaneously to learn an encoding algorithm based on these constituents. These two problems (i.e. learning the constituents, and learning an encoding algorithm) are mutually dependent, which makes this whole process rather subtle to analyse.

In this chapter it will be demonstrated how a self-organising neural network can learn non-trivial ways of encoding input vectors. The type of solution that emerges could be implemented either in software, or possibly in hardware on an imaging sensor itself. Note that the results are not restricted to image data alone, although this is their most natural area of application.

### B. Vector Quantisers

The existing literature on the simplest type of encoder (i.e. the vector quantiser (VQ)) includes the following examples:

1. A standard VQ, in which the input space is partitioned into a number of non-overlapping quantisation cells [4], which is also known as an LBG vector quantiser (after the initials of the authors of [4]). In operation, all of the input vectors lying closest in a Euclidean sense to a given code vector are assigned the same code index (which thus defines a quantisation cell), and the approximate reconstruction of these inputs is the centroid of the quantisation cell. This type of VQ can be viewed as a single-layer winner-take-all (WTA) neural network.

2. A topographic VQ, in which the code indices and quantisation cells are arranged so that code indices that differ by a small amount are assigned to quantisation cells that are close to each other (in the Euclidean sense). This topographic property automatically emerges if a VQ is optimised for encoding input vectors for transmission along a noisy communication channel [1, 3]. The Kohonen topographic mapping network [2] is an approximation to this type of encoder, as was explained in [7].

3. Simultaneously use more than one standard VQ, with each VQ encoding only a subspace of the input (see for example [8]); in effect, more than one code index is used to encode the input vector. By this means, a high-dimensional space can be split up into a number of lower dimensional pieces. This type of VQ is equivalent to mul-

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

This appeared as chapter 10 in Combining Artificial Neural Networks, Springer-Verlag, (ed. A. J. C. Sharkey), pp. 235-263, 1999.  
© British Crown Copyright 1998.

multiple single-layer WTA neural network modules operating independently on different subspaces of the input. This is an example of a factorial encoder, in which the input is split into a number of separate parts, or factors.

4. The simultaneous use of multiple VQs can be extended to a tree-like network of VQs [6]. This type of VQ is equivalent to multiple single layer WTA neural network modules connected together in a tree-like network of modules.

In this chapter a more general type of VQ will be discussed, in which more than one code index is used to encode the input vector; in this respect the network is a factorial encoder. For simplicity, only the case of a 2-layer network (i.e. an input and an output layer) will be considered, but otherwise the network will be obliged to learn how to make use of all of its neurons. The simplest encoder which has all of the required behaviour, and which includes the above 2-layer examples as special cases, is one in which the neurons fire discretely in response to the input, and, after a finite number of firing events has occurred, the input is then reconstructed as accurately as possible (in the Euclidean sense). In the special case where only a single firing event is observed, this reduces to a standard LBG vector quantiser that was discussed in case 1 above. In the more general case, where a finite number of firing events is observed, this can lead to factorial encoder networks of the type that was discussed in case 3 above.

This type of VQ, in which multiple code indices (or, from the neural viewpoint, multiple firing events) are used, may be informally viewed as an application of the minimum description length (MDL) principle [10], in which an optimal encoder uses the minimum number of bits (on average) to encode and reconstruct each input vector. In the MDL approach there are three contributions to the number of bits used: code cost, reconstruction cost, and model cost. The model cost is the number of bits used to specify the encoding scheme, which is a one-time cost that is ignored in this chapter. The code cost is the number of bits used to specify the code index, which is determined by the entropy of the probability distribution over code indices, and not merely by the total number of code indices. Finally, the reconstruction cost is the number of bits used to reconstruct the input given that the code index is known. Overall, the network that is considered in this chapter may be viewed as minimising the reconstruction cost alone, when using a Gaussian model for the distribution of possible reconstructions from a given code index, subject to the constraint that the total number of code indices is held constant; this is not a pure MDL approach.

### C. Curved Manifolds

The purpose of this chapter is to derive optimal ways of encoding data using neural networks in which multiple firing events are observed, and to show that factorial

encoder networks are optimal when the input data lies on a curved manifold, provided that the network does not have too many neurons, and that a sufficiently large number of firing events is observed.

In order to get a feel for how curved manifolds arise in image data, consider the examples shown in Figure 1 and Figure 2, which show the manifold generated by a single target (Figure 1) and by a pair of targets (Figure 2), when projected onto three neighbouring pixels (i.e. the locus of the 3-vector formed from these pixel values is plotted as the target(s) move around). Clearly, these image manifolds are curved, and the curvature gets greater the narrower the Gaussian bumps used to generate the target images become.

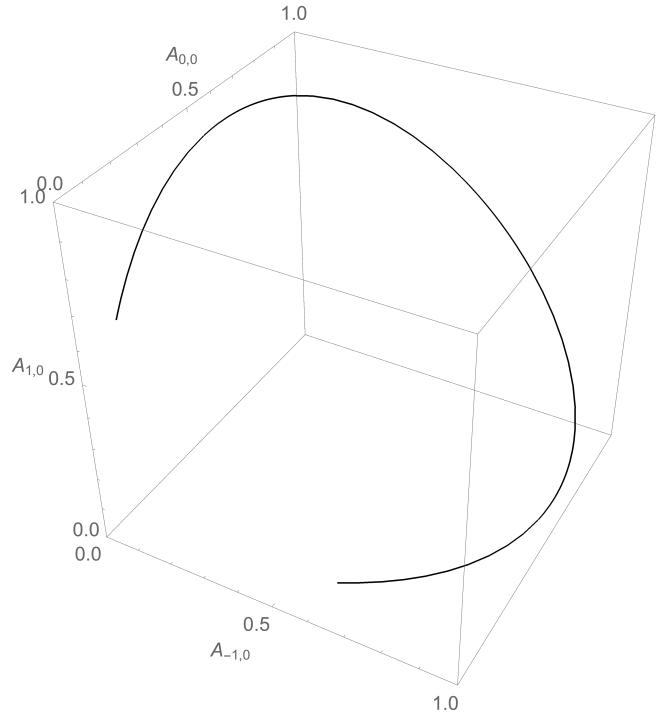


Figure 1: Manifold formed when the 1-dimensional image of a target (a Gaussian bump with a half-width of one pixel) is moved around. Only the projection  $A_{i,j}$  onto the pixels at  $(i, j) = (-1, 0), (0, 0)$  and  $(1, 0)$  is shown.

It is not at all obvious how best to encode vectors that lie on such manifolds. For instance, one might try to tile the manifold with a large number of small quantisation cells obtained from some variant of a VQ, or one might try to project the manifold onto a basis obtained from some variant of principal components analysis (PCA). In fact, these two examples are both special cases of the approach that is advocated in this chapter; a VQ corresponds to a single firing event, whereas PCA corresponds to an infinite number of firing events.

The problem of optimally encoding data that is derived from a general curved manifold requires a numerical solution. However, in order to develop our understanding, it is best to start with an analytically tractable example

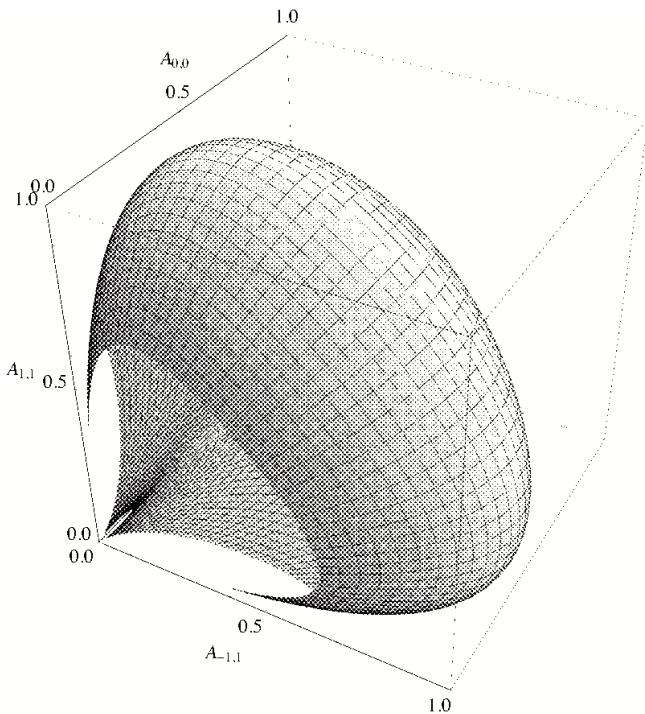


Figure 2: Manifold formed when the 2-dimensional image of a target (a Gaussian bump with half-widths of one pixel in each direction) is moved around. Only the projection  $A_{i,j}$  onto the pixels at  $(i,j) = (-1,1), (0,0)$  and  $(1,1)$  is shown.

based on a simple curved manifold, which is carefully selected to preserve the essential features of more general curved manifolds. With this in mind, the most important feature to preserve in the analytic example is curvature. The simplest curved manifold is a circle, which has the same curvature everywhere. For instance, a circular manifold could be used as a local approximation to the manifold shown in figure Figure 1. However, the example shown in Figure 2 requires two coordinates to specify a point in the manifold, so it cannot locally be approximated by a circle. In this case, there are two obvious alternative possibilities that could be used for modelling curvature simultaneously in two directions: a 2-sphere, or a 2-torus. Note that if the two curvatures are not the same, then the 2-sphere and 2-torus must be distorted appropriately. It turns out that a 2-torus is much easier to handle analytically than a 2-sphere, so it will be used in this chapter.

In Figure 3 a comparison is made between a toroidal manifold and a typical input manifold, which demonstrates that locally the approximation is very good, and that the approximation breaks down at displacements that are of order of 2 to 3 target widths. For instance, the behaviour of a network whose input is images with a limited size (i.e. 2 to 3 target widths) may be estimated by analysing the behaviour of the same network when its input is derived from a toroidal manifold. All of the results reported here should be interpreted in this light.

#### D. Structure of this Chapter

In Section II the basic theoretical framework is introduced (see [7]), from which some expressions are derived for optimising a network which is trained on data from a toroidal input manifold. In Section III the detailed results for encoding a circular input manifold are given (which are trivially related to the corresponding results for the case of joint encoding of a 2-torus), and in Section IV these results are extended to the case of factorial encoding of a 2-torus. These results are a generalisation of those in [8]; the network is now fully optimised, rather than being constrained to be a WTA network, as it was in [8]. There is a considerable amount of algebra involved in deriving these results, which was mostly done using algebraic manipulator software [13], and which is not reported in detail here. In Section V some useful asymptotic results are given, and in Section VI it is shown how the optimal network behaviour may be approximated [12].

## II. BASIC THEORETICAL FRAMEWORK

The encoder model that is assumed throughout this chapter is a 2-layer network of neurons. The input layer contains an activity pattern that represents the components of the input vector  $\mathbf{x}$ . The output layer consists of  $M$  discretely firing neurons, and the output  $\mathbf{y}$  is an encoded version of the input  $\mathbf{x}$ , where  $\mathbf{y}$  consists of observations of the locations  $(y_1, y_2, \dots, y_n)$  of the first  $n$  firing events that occur in response to input  $\mathbf{x}$ .

The information content of  $\mathbf{y}$  may be used to draw inferences about  $\mathbf{x}$ . This can be formalised by using Bayes' theorem in the form

$$\Pr(\mathbf{x}|\mathbf{y}) = \frac{\Pr(\mathbf{y}|\mathbf{x})\Pr(\mathbf{x})}{\int d\mathbf{x}' \Pr(\mathbf{y}|\mathbf{x}')\Pr(\mathbf{x}')} \quad (1)$$

where the probability density function (PDF)  $\Pr(\mathbf{x}|\mathbf{y})$  of the input  $\mathbf{x}$  given that the output  $\mathbf{y}$  is known (i.e. the inference about  $\mathbf{x}$ ) is completely determined by the likelihood  $\Pr(\mathbf{y}|\mathbf{x})$  that output  $\mathbf{y}$  occurs when input  $\mathbf{x}$  is present, and the prior PDF  $\Pr(\mathbf{x})$  that input  $\mathbf{x}$  could occur irrespective of whether  $\mathbf{y}$  is being observed.

The 2-layer network of neurons will be encouraged to adjust the way that it responds to its input, so that its output contains the maximum amount of information about the input. Information could be rigorously quantified by using the mutual information  $I[\mathbf{x}; \mathbf{y}]$  (where  $I[\mathbf{x}; \mathbf{y}] \equiv \sum_y \int d\mathbf{x} \Pr(\mathbf{x}, \mathbf{y}) \log \frac{\Pr(\mathbf{x}, \mathbf{y})}{\Pr(\mathbf{x})\Pr(\mathbf{y})}$ ) between input and output [11], and then maximising  $I[\mathbf{x}; \mathbf{y}]$  with respect to all possible ways in which the neurons could respond to the input. However,  $I[\mathbf{x}; \mathbf{y}]$  is not an easy quantity to handle in numerical simulations, essentially because  $\Pr(\mathbf{x}, \mathbf{y})$  occurs inside the logarithmic factor.

Rather than use mutual information to characterise the network performance, a closely related information-like quantity may be defined that is much easier to work

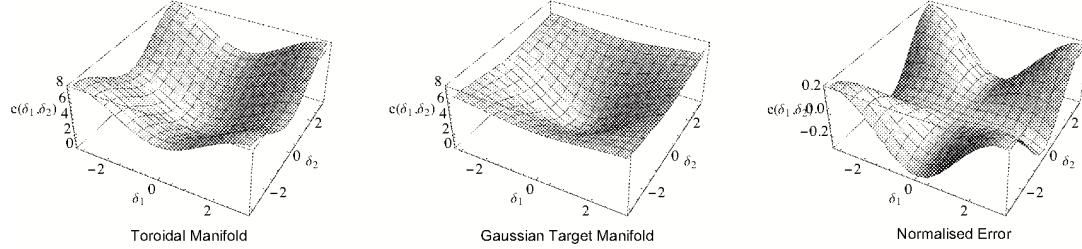


Figure 3: The purpose of these diagrams is to demonstrate how good a local approximation a 2-torus is to a typical 2-dimensional input manifold. Each manifold has 2 internal coordinates  $(\delta_1, \delta_2)$ , and Euclidean distance  $e(\delta_1, \delta_2)$  from some fixed reference point is measured as a function of these coordinates. The left hand diagram is generated from a 2-torus formed as the Cartesian product of a pair of unit circles, and whose 2 angles are  $(\delta_1, \delta_2)$ . The middle diagram is generated by a target whose image has a Gaussian profile, and whose centre is located at position  $\alpha(\delta_1, \delta_2)$ , and whose standard deviation is  $\beta$ . For  $\alpha = 0.9$  the value of  $\beta$  that gives the least squares fit of the left and middle diagrams is  $\beta \approx 1.038$ , and the right hand diagram then shows the difference between the values recorded in the left and middle diagrams (divided by the values in the middle diagram). The error is small except for a region near  $|\delta_1| = \pi$  and/or  $|\delta_2| = \pi$ , as expected.

with, and which ensures backward compatibility with other neural network approaches. Thus a network objective function is introduced in Section II A [7], and the stationarity conditions which must be satisfied for an optimal network behaviour are quoted in Section II B. Joint encoding on a 2-torus is discussed in Section II C, and factorial encoding on a 2-torus is discussed in Section II D.

### A. Objective Function

One way of measuring the information content that is available in the output layer of a 2-layer network is to attempt to deduce the state of the input layer from knowledge of the state of the output layer. An objective function may be defined as follows [7]

$$D \equiv \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \int d\mathbf{x}' \Pr(\mathbf{x}'|\mathbf{y}) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (2)$$

where  $\int d\mathbf{x} \Pr(\mathbf{x})(\dots)$  integrates over all the possible states of the input layer,  $\sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x})(\dots)$  sums over all the possible states of the output layer given that the state of the input layer is known,  $\int d\mathbf{x}' \Pr(\mathbf{x}'|\mathbf{y})(\dots)$  integrates over all the possible reconstructed states of the input layer given that the state of the output layer is known, and  $\|\mathbf{x} - \mathbf{x}'\|^2$  is the reconstruction error which is the Euclidean distance between the original input and its eventual reconstruction. Bayes' theorem (see Equation 1) relates  $\Pr(\mathbf{x}|\mathbf{y})$  to  $\Pr(\mathbf{y}|\mathbf{x})$ .

If the observed state of the output layer is the locations of  $n$  firing events on  $M$  neurons, then this expression for  $D$  can be manipulated into the following form [7]:

$$D \equiv 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1=1}^M \sum_{y_2=1}^M \cdots \sum_{y_n=1}^M \Pr(y_1, y_2, \dots, y_n | \mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y_1, y_2, \dots, y_n)\|^2 \quad (3)$$

where  $\Pr(\mathbf{y}|\mathbf{x})$  has now been replaced by the more explicit notation  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$ , and  $\mathbf{x}'(y_1, y_2, \dots, y_n)$  is a vector that is given by

$$\mathbf{x}'(y_1, y_2, \dots, y_n) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|y_1, y_2, \dots, y_n) \mathbf{x} \quad (4)$$

where  $\Pr(\mathbf{x}|y_1, y_2, \dots, y_n)$  may be expressed in terms

of  $\Pr(\mathbf{x})$  and  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  by using Bayes' theorem in Equation 1. The factor of 2 that has appeared in the expression for  $D$  in Equation 3 arises because the original expression for  $D$  (see Equation 2) is symmetrical under interchange of  $\mathbf{x}$  and  $\mathbf{x}'$ , which gives two identical contributions to the overall value of  $D$ . The goal now is to minimise the expression for  $D$  in Equation 3 with

respect to the function  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$ . The correct value for  $\mathbf{x}'(y_1, y_2, \dots, y_n)$  may be determined by treating  $\mathbf{x}'(y_1, y_2, \dots, y_n)$  as unknown parameter that has to be adjusted to minimise  $D$ .

$\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  may be interpreted as a recognition model which transforms the state of the input layer into (a probabilistic description of) the state of the output layer, and  $\mathbf{x}'(y_1, y_2, \dots, y_n)$  may be regarded as the corresponding generative model that transforms the state of the output layer into (an approximate reconstruction of) the state of the input layer.

There is so much flexibility in the choice of  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  (and the corresponding  $\mathbf{x}'(y_1, y_2, \dots, y_n)$ ) that even if it minimises  $D$ , it does not necessarily yield an encoded version of the input that is easily interpretable. One way in which a code can be encouraged to have a simple interpretation is to force  $\mathbf{x}'(y_1, y_2, \dots, y_n)$  (i.e. the generative model) to be parameterised thus [7]

$$\mathbf{x}'(y_1, y_2, \dots, y_n) = \frac{1}{n} (\mathbf{x}'(y_1) + \mathbf{x}'(y_2) + \dots + \mathbf{x}'(y_n)) \quad (5)$$

which is a (symmetric) superposition of reference vectors  $\mathbf{x}'(y)$  from each neuron  $y$  that has been observed to fire. In this case each neuron has a clearly identifiable contribution to the reconstruction of the input, which makes it much easier to interpret what each neuron is doing. In this case the  $\|\dots\|^2$  term in  $D$  is symmetric under interchange of the  $(y_1, y_2, \dots, y_n)$ , so only the symmetric part  $S[\Pr(y_1, y_2, \dots, y_n | \mathbf{x})]$  of  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  under interchange of the  $(y_1, y_2, \dots, y_n)$  contributes to  $D$ , because the summation  $\sum_{y_1=1}^M \sum_{y_2=1}^M \dots \sum_{y_n=1}^M (\dots)$  removes all non-symmetric contributions to  $D$ .

Define the marginal probabilities  $\Pr(y_1 | \mathbf{x})$  and  $\Pr(y_1, y_2 | \mathbf{x})$  of the symmetric part  $S[\Pr(y_1, y_2, \dots, y_n | \mathbf{x})]$  of  $\Pr(y_1, y_2, \dots, y_n | \mathbf{x})$  under interchange of the  $(y_1, y_2, \dots, y_n)$  as

$$\begin{aligned} \Pr(y_1 | \mathbf{x}) &\equiv \sum_{y_2, y_3, y_4, \dots, y_n=1}^M S[\Pr(y_1, y_2, \dots, y_n | \mathbf{x})] \\ \Pr(y_1, y_2 | \mathbf{x}) &\equiv \sum_{y_3, y_4, \dots, y_n=1}^M S[\Pr(y_1, y_2, \dots, y_n | \mathbf{x})] \end{aligned} \quad (6)$$

These marginal probabilities are for the case where  $n$  firing events have potentially been observed, but only the locations of 1 (or 2) firing event(s) chosen randomly from the total number  $n$  have actually been observed, with the locations of the other  $n-1$  (or  $n-2$ ) firing events having been averaged over.

If it is assumed that  $\Pr(y_1 | \mathbf{x})$  and  $\Pr(y_1, y_2 | \mathbf{x})$  are related by

$$\Pr(y_1, y_2 | \mathbf{x}) = \Pr(y_1 | \mathbf{x}) \Pr(y_2 | \mathbf{x}) \quad (7)$$

then the objective function  $D$  has an upper bound  $D_1 +$

$D_2$  given by [7]

$$\begin{aligned} D &\leq D_1 + D_2 \\ D_1 &\equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y | \mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \\ D_2 &\equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y | \mathbf{x}) \mathbf{x}'(y) \right\|^2 \end{aligned} \quad (8)$$

Each of the two marginal probabilities in Equation 6 contributes to a different term in  $D_1 + D_2$ ;  $\Pr(y_1 | \mathbf{x})$  contributes to  $D_1$ , whereas  $\Pr(y_1, y_2 | \mathbf{x})$  contributes to  $D_2$ . Informally speaking,  $D_1$  measures the information that a single firing event (out of  $n$  such events) contributes to the reconstruction of the input, whereas  $D_2$  measures the information that pairs of firing events (out of  $n$  such events) contribute to the reconstruction of the input.  $D_1$  is weighted by a factor  $\frac{1}{n}$  which suppresses the single firing event contribution as  $n \rightarrow \infty$ , whereas  $D_2$  is weighted by a factor  $\frac{n-1}{n}$  which suppresses the double firing event contribution as  $n \rightarrow 1$ , as expected. If only the  $D_1$  part of the objective function is used (i.e.  $n = 1$ ), then a standard LBG vector quantiser [4] emerges which approximates the input by a single reference vector  $\mathbf{x}'(y)$ , whereas if only the  $D_2$  part of the objective function is used (i.e.  $n \rightarrow \infty$ ), then the network behaves essentially as a principal component analyser (PCA) which approximates the input by a sum of reference vectors  $\sum_{y=1}^M \Pr(y | \mathbf{x}) \mathbf{x}'(y)$ , where the  $\Pr(y | \mathbf{x})$  are expansion coefficients which sum to unity, and the  $\mathbf{x}'(y)$  are basis vectors.

The upper bound  $D_1 + D_2$  on  $D$  contains LBG encoding and PCA encoding as two limiting cases, and gives a principled way of interpolating between these extremes. This useful property has been bought at the cost of replacing  $D$  by an upper bound bound  $D_1 + D_2$ , which will yield only a suboptimal (from the point of view of  $D$ ) encoder. However, this upper bound can be expected to be tight in cases where the input manifold can be modelled accurately using the parameteric form  $\mathbf{x}'(y_1) + \mathbf{x}'(y_2) + \dots + \mathbf{x}'(y_n)$ . These conditions are well approximated in images which consist of a discrete number of constituents, each of which may be represented by an  $\mathbf{x}'(y)$  for some choice of  $y$ . This model fails in situations where two or more constituents are placed so that they overlap, in which case the image will typically contain occluded objects, whereas the model assumes that the objects linearly superpose. Occlusion is not an easy situation to model, so it will be assumed that the image constituents are sufficiently sparse that they rarely occlude each other.

## B. Stationarity Conditions

The upper bound  $D_1 + D_2$  (see Equation 8) on the objective function  $D$  (see Equation 3) needs to be minimised with respect to two types of parameter: posterior

probabilities  $\Pr(y|\mathbf{x})$  and reference vectors  $\mathbf{x}'(y)$ . This could be done numerically for an arbitrary input PDF  $\Pr(\mathbf{x})$  by using a gradient descent algorithm [7], but here  $D_1 + D_2$  will be analytically minimised for some carefully

chosen special cases of  $\Pr(\mathbf{x})$ .

The stationarity condition  $\frac{\partial(D_1+D_2)}{\partial \mathbf{x}'(y)} = 0$  gives [8]

$$n \int d\mathbf{x} \Pr(\mathbf{x}|y) \mathbf{x} = \mathbf{x}'(y) + (n-1) \int d\mathbf{x} \Pr(\mathbf{x}|y) \sum_{y'=1}^M \Pr(y'|\mathbf{x}) \mathbf{x}'(y') \quad (9)$$

where Bayes' theorem in the form  $\Pr(\mathbf{x})\Pr(y|\mathbf{x}) = \Pr(\mathbf{x}|y)\Pr(y)$  has been used, and  $\Pr(y) > 0$  has been assumed. The  $\frac{\partial(D_1+D_2)}{\partial \mathbf{x}'(y)} = 0$  stationarity condition also has the solution  $\Pr(y) = 0$ , but this solution may be discarded because  $\Pr(y) > 0$  is always the case in practice, because  $\Pr(y) = 0$  would cause one or more neurons to be unused. The right hand side of the stationarity condition in Equation 9 has two contributions: a  $D_1$ -like contribution which is a single reference vector  $\mathbf{x}'(y)$ , plus a  $D_2$ -like contribution which is  $n-1$  times a sum of reference vectors  $\sum_{y'=1}^M (\int d\mathbf{x} \Pr(y'|\mathbf{x}) \Pr(\mathbf{x}|y)) \mathbf{x}'(y')$ , where the coefficient  $\int d\mathbf{x} \Pr(y'|\mathbf{x}) \Pr(\mathbf{x}|y)$  accounts for the ef-

fect (at neuron  $y$ ) of observing all pairs of firing events  $(y, y')$  for  $y' = 1, 2, \dots, M$ . The sum of these two terms is  $n$  times the total reference vector that is effectively associated with neuron  $y$ , which, by analogy with Equation 4, must be equal to  $n$  times  $\int d\mathbf{x} \Pr(\mathbf{x}|y) \mathbf{x}$ , as is indeed the case in Equation 9. Thus the stationarity condition in Equation 9 may be viewed as the form that Equation 4 takes when the reference vector is as given in Equation 5 and the posterior probability is as given in Equation 7.

The stationarity condition  $\frac{\partial(D_1+D_2)}{\partial \Pr(y|\mathbf{x})} = 0$  gives [9, <http://arxiv.org/abs/1505.00444> ]

$$\sum_{y'=1}^M (\Pr(y'|\mathbf{x}) - \delta_{y,y'}) \mathbf{x}'(y') \cdot \left( \frac{1}{2} \mathbf{x}'(y') - n \mathbf{x} + (n-1) \sum_{y''=1}^M \Pr(y''|\mathbf{x}) \mathbf{x}'(y'') \right) = 0 \quad (10)$$

where the constraint  $\sum_{y'=1}^M \Pr(y'|\mathbf{x}) = 1$  has been imposed, and  $\Pr(\mathbf{x}) > 0$  and  $\Pr(y|\mathbf{x}) > 0$  have been assumed. The  $\frac{\partial(D_1+D_2)}{\partial \Pr(y|\mathbf{x})} = 0$  stationarity condition also has two other solutions: either  $\Pr(\mathbf{x}) = 0$ , or  $\Pr(\mathbf{x}) > 0$  and  $\Pr(y|\mathbf{x}) = 0$ . Using the normalisation constraint  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$ , the last of these solutions ensures that  $\Pr(y'|\mathbf{x}) \leq 1$  for  $y' \neq y$ , and when all values of  $y$  are considered the net effect is to constrain  $\Pr(y|\mathbf{x})$  to the interval  $0 \leq \Pr(y|\mathbf{x}) \leq 1$ , as expected.

The solutions of the stationarity condition for  $\Pr(y|\mathbf{x})$  in Equation 10 are piecewise linear functions of  $\mathbf{x}$ , as was shown in [9]. This result can be seen to be intuitively reasonable because  $D_1 + D_2$  is of the form  $\int d\mathbf{x} \Pr(\mathbf{x}) f(\mathbf{x})$ , where  $f(\mathbf{x})$  is a linear combination of terms of the form  $\mathbf{x}^i \Pr(y|\mathbf{x})^j$  (for  $i = 0, 1, 2$  and  $j = 0, 1, 2$ ), which is a quadratic form in  $\mathbf{x}$  (ignoring the  $\mathbf{x}$ -dependence of  $\Pr(y|\mathbf{x})$ ). However, the terms that appear in this linear combination are such that a  $\Pr(y|\mathbf{x})$  that is a piecewise linear function of  $\mathbf{x}$  guarantees that  $f(\mathbf{x})$  is a piecewise linear combination of terms of the form  $\mathbf{x}^i$  (for  $i = 0, 1, 2$ ),

which is a quadratic form in  $\mathbf{x}$  (the normalisation constraint  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$  is used to remove a contribution that is potentially quartic in  $\mathbf{x}$ ). Thus a piecewise linear dependence of  $\Pr(y|\mathbf{x})$  on  $\mathbf{x}$  does not lead to any dependencies on  $\mathbf{x}$  that are not already explicitly present in  $D_1 + D_2$ . The stationarity condition on  $\Pr(y|\mathbf{x})$  (see Equation 10) then imposes conditions on the allowed piecewise linearities that  $\Pr(y|\mathbf{x})$  can have.

This piecewise linear property of  $\Pr(y|\mathbf{x})$  is an enormous simplification, because it means that rather than searching the infinite dimensional space of functions  $\Pr(y|\mathbf{x})$  for the optimal ones that minimise  $D_1 + D_2$ , one needs only search a finite dimensional space of piecewise linear functions  $\Pr(y|\mathbf{x})$  (subject to the constraints  $0 \leq \Pr(y|\mathbf{x}) \leq 1$  and  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$ ).

### C. Joint Encoding

Joint encoding is characterised by a  $\Pr(y|\mathbf{x})$  in which the neurons labelled by  $y$  form a discretised version of the

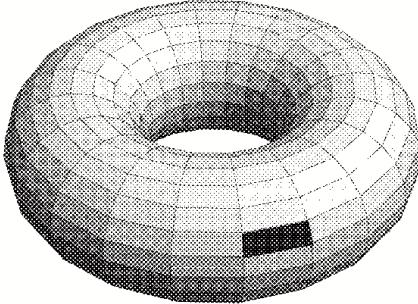


Figure 4: Joint encoding of a vector that lives on a 2-torus. The 2-torus is tiled with quantisation cells which overlap when  $n > 1$ . In the diagram, the torus is overlaid with a  $20 \times 20$  toroidal lattice, and a typical joint encoding cell is highlighted (this would use a total of  $400 = 20 \times 20$  neurons).

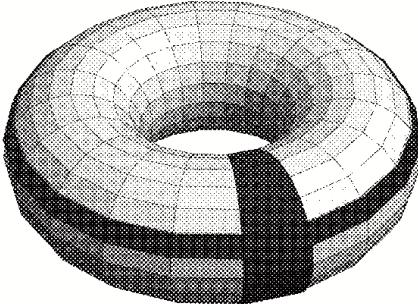


Figure 5: Factorial encoding of a vector that lives on a 2-torus. Each of the 2 circular subspaces within the 2-torus is tiled with quantisation cells which overlap when  $n > 1$ . An accurate encoding is obtained by a process that is akin to triangulation, in which the intersection between the 2 orthogonal encoding cells defines a region of the 2-torus that is equivalent to the corresponding joint encoding quantisation cell. In the diagram, the torus is overlaid with a  $20 \times 20$  toroidal lattice, and a typical pair of intersecting factorial encoding cells is highlighted (this would use a total of  $40 = 20 + 20$  neurons).

manifold that  $\mathbf{x}$  lives on. For instance, when  $\mathbf{x}$  lives on a 2-torus, so that  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$  where  $\mathbf{x}_1 = (\cos \theta_1, \sin \theta_1)$  and  $\mathbf{x}_2 = (\cos \theta_2, \sin \theta_2)$ , the  $\Pr(y|\mathbf{x})$  typically behave as shown in Figure 4. This diagram makes clear why such encoding is described as “joint”, because the response of each neuron depends on the values of both  $\mathbf{x}_1$  and  $\mathbf{x}_2$ . The neural network implementation of this type of joint encoder would have connections from each neuron to all of the input pixels.

For joint encoding of a 2-torus  $y$  must be replaced by the pair  $(y_1, y_2)$ , where the  $y_1$  index labels one direction around the toroidal lattice, and  $y_2$  labels the other direction (this notation must not be confused with the  $(y_1, y_2, \dots, y_n)$  notation that was used in Section II A). Thus  $\Pr(y|\mathbf{x}) = \Pr(y_1, y_2|\mathbf{x}_1, \mathbf{x}_2)$  with  $1 \leq y_1 \leq \sqrt{M}$  and  $1 \leq y_2 \leq \sqrt{M}$ . For simplicity, assume  $\Pr(\mathbf{x}_1, \mathbf{x}_2) = \Pr(\mathbf{x}_1)\Pr(\mathbf{x}_2)$ , where  $\Pr(\mathbf{x}_1)$  and  $\Pr(\mathbf{x}_2)$  are each constant. The following results for  $D_1$  and  $D_2$  may then easily be derived

$$\begin{aligned}
 D_1 &= \frac{4}{n} \int d\mathbf{x}_1 \Pr(\mathbf{x}_1) \sum_{y_1=1}^{\sqrt{M}} \Pr(y_1|\mathbf{x}_1) \|\mathbf{x}_1 - \mathbf{x}'_1(y_1)\|^2 \\
 D_2 &= \frac{4(n-1)}{n} \int d\mathbf{x}_1 \Pr(\mathbf{x}_1) \left\| \mathbf{x}_1 - \sum_{y_1=1}^{\sqrt{M}} \Pr(y_1|\mathbf{x}_1) \mathbf{x}'_1(y_1) \right\|^2
 \end{aligned} \tag{11}$$

These results for  $D_1$  and  $D_2$  show that, when  $\Pr(\mathbf{x}_1, \mathbf{x}_2) = \Pr(\mathbf{x}_1)\Pr(\mathbf{x}_2)$  (where  $\Pr(\mathbf{x}_1)$  and  $\Pr(\mathbf{x}_2)$  are each constant), the problem of optimising a joint encoder is equivalent to the problem of optimising an encoder for  $\mathbf{x}_1$  alone (with the replacement  $M \rightarrow \sqrt{M}$ ), and then multiplying the value of  $D_1 + D_2$  by a factor 2 to account for  $\mathbf{x}_2$  as well. This illustration of the behaviour of joint encoder posterior probabilities in the case of  $\Pr(y_1, y_2 | \mathbf{x}_1, \mathbf{x}_2)$  may readily be generalised to higher dimensions.

#### D. Factorial Encoding

Factorial encoding is characterised by a  $\Pr(y|\mathbf{x})$  in which the neurons labelled by  $y$  are partitioned into a number of subsets, each of which forms a discretised version of a subspace of the manifold that  $\mathbf{x}$  lives on. For instance, when  $\mathbf{x}$  lives on a 2-torus, and the neurons are

partitioned into two equal-sized subsets, the  $\Pr(y|\mathbf{x})$  typically behave as shown in Figure 5. This diagram makes clear why such encoding is described as “factorial”, because the response of each neuron depends on only one of  $x_1$  and  $x_2$ , or, in other words, on only one factor that parameterises the input space. The neural network implementation of this type of factorial encoder would have connections from each neuron to only half of the input pixels.

For factorial encoding of a 2-torus  $\Pr(y|\mathbf{x}) = \Pr(y|\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{2}\Pr(y|\mathbf{x}_1) + \frac{1}{2}\Pr(y|\mathbf{x}_2)$ , where  $\sum_{y=1}^{\frac{M}{2}} \Pr(y|\mathbf{x}_1) = 1$ ,  $\sum_{y=\frac{M}{2}+1}^M \Pr(y|\mathbf{x}_2) = 1$ ,  $\Pr(y|\mathbf{x}_1) = 0$  for  $\frac{M}{2} + 1 \leq y \leq M$ , and  $\Pr(y|\mathbf{x}_2) = 0$  for  $1 \leq y \leq \frac{M}{2}$ . For simplicity, assume  $\Pr(\mathbf{x}_1, \mathbf{x}_2) = \Pr(\mathbf{x}_1)\Pr(\mathbf{x}_2)$ , where  $\Pr(\mathbf{x}_1)$  and  $\Pr(\mathbf{x}_2)$  are each constant. The following results for  $D_1$  and  $D_2$  may then be derived

$$\begin{aligned} D_1 &= \frac{4}{n} \left( \int d\mathbf{x}_1 \Pr(\mathbf{x}_1) \sum_{y=1}^{\frac{M}{2}} \frac{1}{2} \Pr(y|\mathbf{x}_1) \|\mathbf{x}_1 - \mathbf{x}'_1(y)\|^2 + \int d\mathbf{x}_2 \Pr(\mathbf{x}_2) \frac{1}{2} \|\mathbf{x}_2\|^2 \right) \\ D_2 &= \frac{4(n-1)}{n} \int d\mathbf{x}_1 \Pr(\mathbf{x}_1) \left\| \mathbf{x}_1 - \sum_{y=1}^{\frac{M}{2}} \frac{1}{2} \Pr(y|\mathbf{x}_1) \mathbf{x}'_1(y) \right\|^2 \end{aligned} \quad (12)$$

These results for  $D_1$  and  $D_2$  show that, when  $\Pr(\mathbf{x}_1, \mathbf{x}_2) = \Pr(\mathbf{x}_1)\Pr(\mathbf{x}_2)$  (where  $\Pr(\mathbf{x}_1)$  and  $\Pr(\mathbf{x}_2)$  are each constant), the problem of optimising a factorial encoder is closely related to the problem of optimising two 1-dimensional encoders. This illustration of the behaviour of factorial encoder posterior probabilities in the case of  $\Pr(y|\mathbf{x}_1, \mathbf{x}_2)$  may readily be generalised to higher dimensions.

### III. CIRCULAR MANIFOLD

The analysis of how to encode data that lives on a curved manifold begins with the case of data that lives on a circle. In particular, assume that the input vector  $\mathbf{x}$  is uniformly distributed on the unit circle centred on the origin, so that  $\mathbf{x}$  can be parameterised by a single angular variable  $\theta$ , thus

$$\begin{aligned} \mathbf{x} &= (\cos \theta, \sin \theta) \\ \int d\mathbf{x} \Pr(\mathbf{x}) (\dots) &= \frac{1}{2\pi} \int_0^{2\pi} d\theta (\dots) \end{aligned} \quad (13)$$

The posterior probability  $\Pr(y|\mathbf{x})$  may thus be replaced by  $\Pr(y|\theta)$ , and for purely conventional reasons, the range of  $y$  is now chosen to be  $y = 0, 1, \dots, M-1$  rather than

$y = 1, 2, \dots, M$ . The set of  $M$  posterior probabilities for  $y = 0, 1, \dots, M-1$  can be parameterised as

$$\Pr(y|\theta) = p \left( \theta - \frac{2\pi y}{M} \right) \quad (14)$$

where  $p(\theta)$  is the  $\theta$ -dependence of the posterior probability associated with the  $y = 0$  neuron, which must have a piecewise sinusoidal dependence on  $\theta$  (i.e. made out of pieces that each have the functional form  $a + b \cos \theta + c \sin \theta$ ) in order to ensure that  $\Pr(y|\mathbf{x})$  is piecewise linear. Similarly, the  $M$  corresponding reference vectors can be parameterised as

$$\mathbf{x}'(y) = r \left( \cos \left( \frac{2\pi y}{M} \right), \sin \left( \frac{2\pi y}{M} \right) \right) \quad (15)$$

which all have length  $r$  and form a regular  $M$ -sided polygon.

It turns out that, for input vectors that live on a circular manifold, optimal joint encoding never causes more than 3 different neurons to fire in response to a given input (i.e. no more than 3 posterior probabilities overlap). This severely limits the number of different piecewise functions that have to be manipulated when solving the  $D_1 + D_2$  minimisation problem for input vectors

that live on a circle. A similar simplification also holds for joint and factorial encoding of a 2-torus. The case of 2 overlapping posterior probabilities can be optimised without too much difficulty, but the case of 3 overlapping posterior probabilities involves a prohibitively large amount of algebra, so it is convenient to use an algebraic manipulator [13] in this case. The calculations are highly structured, so the use of an algebraic manipulator could in principle be used in order to solve more complicated problems.

All of the results for encoding input data that lives on a circular manifold may be derived from the expression for  $D_1 + D_2$  in Equation 8 (and the corresponding stationarity conditions), with the replacement given in Equation 13 to ensure that the input manifold corresponds to a uniform distribution of data around a unit circle, and the functional forms given in Equation 14 and Equation 15 to ensure that the optimal solution is symmetric.

The corresponding results for joint encoding of data that lives on a 2-torus can be obtained directly from these results (see Section II C). The expression for the minimum value of  $D_1 + D_2$  for joint encoding a 2-torus using  $\sqrt{M} \times \sqrt{M}$  neurons is obtained by making the replacement  $M \rightarrow \sqrt{M}$  in the expression for the minimum value of  $D_1 + D_2$  for encoding a circle using  $M$  neurons, and then multiplying this result by 2 in order to account for both the circles that form the 2-torus (see Equation 11).

### A. 2 Overlapping Posterior Probabilities

Because the neurons have an angular separation of  $\frac{2\pi}{M}$ , the functional form of  $p(\theta)$  may be defined as

$$p(\theta) = \begin{cases} 1 & 0 \leq |\theta| \leq \frac{\pi}{M} - s \\ f(\theta) & \frac{\pi}{M} - s \leq |\theta| \leq \frac{\pi}{M} + s \\ 0 & |\theta| \geq \frac{\pi}{M} + s \end{cases} \quad (16)$$

where the  $s$  parameter is half the angular width of the overlap between the posterior probabilities of adjacent neurons on the unit circle, in which case  $0 \leq s \leq \frac{\pi}{M}$  ensures that no more than 2 neurons can respond to a given input. Anticipating the optimum solution, a typical example of this type of posterior probability is shown in Figure 6.

In order to guarantee that  $\Pr(y|\mathbf{x})$  has a piecewise linear dependence on  $\mathbf{x}$ , as required of solutions of Equa-

tion 10,  $f(\theta)$  must have the sinusoidal dependence  $f(\theta) = a + b \cos \theta + c \sin |\theta|$ , where the use of  $|\theta|$  arises because  $p(\theta) = p(-\theta)$ . Note that the  $\Pr(\mathbf{x}) = 0$  solution to the stationarity condition on  $\Pr(y|\mathbf{x})$  (see Equation 10) implies that  $\Pr(y|\mathbf{x})$  is undefined for any  $\mathbf{x}$  that does not lie on the unit circle. However, for those  $\mathbf{x}$  that do lie on the unit circle, the  $a$ ,  $b$  and  $c$  parameters can be determined by demanding continuity of  $p(\theta)$  at the ends of its piecewise intervals (i.e. at  $\theta = \frac{\pi}{M} - s$  and  $\theta = \frac{\pi}{M} + s$ ), and by demanding that the total probability of any neuron firing first is unity (i.e. the total posterior probability is normalised such that  $f(\theta) + f(\frac{2\pi}{M} - \theta) = 1$  in the interval  $\frac{\pi}{M} - s \leq \theta \leq \frac{\pi}{M} + s$ ), to obtain

$$f(\theta) = \frac{1}{2} + \frac{1}{2} \frac{\sin(\frac{\pi}{M} - \theta)}{\sin s} \quad (17)$$

This corresponds to a piecewise linear contribution to  $\Pr(y|\mathbf{x})$  whose gradient points in the  $(-\sin(\frac{\pi}{M}), \cos(\frac{\pi}{M}))$  direction. A typical example of this type of posterior probability is shown in Figure 6.

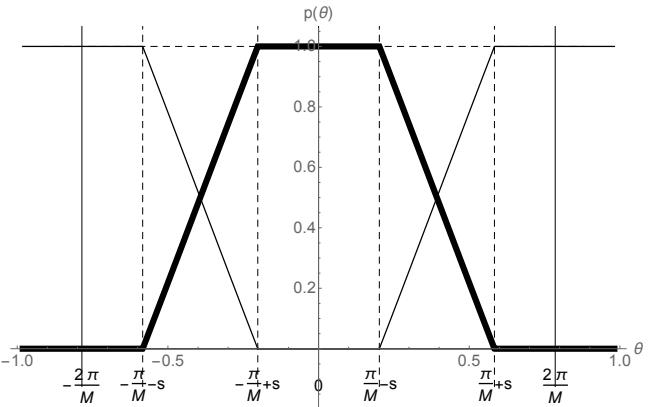


Figure 6: Plot of the optimal  $p(\theta)$  for  $M = 8$  and  $n = 2$ . The optimal value of  $s$  is  $s \approx 0.49 \frac{\pi}{M}$ . The departure of  $p(\theta)$  from linearity in the interval  $\frac{\pi}{M} - s \leq \theta \leq \frac{\pi}{M} + s$  is too small to be easily seen.

Without loss of generality (because the solution is symmetric under rotations of  $\theta$  which are multiples of  $\frac{2\pi}{M}$ ) set  $y = 0$  in Equation 10, to obtain in the interval  $\frac{\pi}{M} - s \leq \theta \leq \frac{\pi}{M} + s$

$$0 = r \csc^2 s \sin\left(\frac{\pi}{M}\right) \sin\left(\frac{\pi}{M} - \theta\right) \left(\sin s - \sin\left(\frac{\pi}{M} - \theta\right)\right) \left(n \sin s - (n-1) r \sin\left(\frac{\pi}{M}\right)\right) \quad (18)$$

which may be solved for the optimum length  $r$  of the reference vectors, to yield

$$r = \frac{n}{n-1} \frac{\sin s}{\sin\left(\frac{\pi}{M}\right)} \quad (19)$$

Set  $y = 0$  in Equation 9 to obtain a transcendental equation that must be satisfied by the optimum  $s$

$$\frac{\sin s}{\sin(\frac{\pi}{M})} - \frac{n-1}{n} \frac{M}{\pi} \sin\left(\frac{\pi}{M}\right) (\cos s + s \sin s) = 0 \quad (20)$$

The symmetry of the solution may be used to make the replacement  $\frac{1}{2\pi} \int_0^{2\pi} d\theta(\dots) \rightarrow \frac{M}{\pi} \int_0^{\frac{\pi}{M}} d\theta(\dots)$  in the expressions for  $D_1$  and  $D_2$ . The expression for  $D_1$  may then be written as

$$D_1 = \frac{2M}{n\pi} \left( \begin{array}{l} \int_0^{\frac{\pi}{M}-s} d\theta \left\| \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} - r \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\|^2 \\ + \int_{\frac{\pi}{M}-s}^{\frac{\pi}{M}} d\theta f(\theta) \left\| \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} - r \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\|^2 \\ + \int_{\frac{\pi}{M}}^{\frac{\pi}{M}+s} d\theta f(\frac{2\pi}{M} - \theta) \left\| \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} - r \begin{pmatrix} \cos \frac{2\pi}{M} \\ \sin \frac{2\pi}{M} \end{pmatrix} \right\|^2 \end{array} \right) \quad (21)$$

and the expression for  $D_2$  may be written out as

$$D_2 = \frac{2(n-1)M}{n\pi} \left( \begin{array}{l} \int_0^{\frac{\pi}{M}-s} d\theta \left\| \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} - r \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\|^2 \\ + \int_{\frac{\pi}{M}-s}^{\frac{\pi}{M}} d\theta \left\| \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} - rf(\theta) \begin{pmatrix} 1 \\ 0 \end{pmatrix} - rf(\frac{2\pi}{M} - \theta) \begin{pmatrix} \cos \frac{2\pi}{M} \\ \sin \frac{2\pi}{M} \end{pmatrix} \right\|^2 \end{array} \right) \quad (22)$$


---

which may be evaluated and simplified to yield the minimum  $D_1 + D_2$  as

$$D_1 + D_2 = 2 - \frac{n}{n-1} \frac{M}{2\pi} (2s + \sin(2s)) \quad (23)$$

The value of  $s$  in this expression for  $D_1 + D_2$  is the solution of Equation 20 for the chosen values of  $M$  and  $n$ .

Note that the expression for  $r$  in Equation 19 and the expression for  $D_1 + D_2$  in Equation 23 both have a finite limits as  $n \rightarrow 1$ , because the limiting behaviour of the solution  $s$  of Equation 20 is  $s \rightarrow (n-1) \frac{M}{\pi} \sin^2(\frac{\pi}{M})$  (see the asymptotic results in Section V), which contains a factor  $n-1$  to cancel the  $\frac{1}{n-1}$  factor that appears in both Equation 19 and Equation 23.

### B. 3 Overlapping Posterior Probabilities

Because the neurons have an angular separation of  $\frac{2\pi}{M}$ , the functional form of  $p(\theta)$  may be defined as

$$p(\theta) = \begin{cases} f_1(\theta) & 0 \leq |\theta| \leq -\frac{\pi}{M} + s \\ f_2(\theta) & -\frac{\pi}{M} + s \leq |\theta| \leq \frac{3\pi}{M} - s \\ f_3(\theta) & \frac{3\pi}{M} - s \leq |\theta| \leq \frac{\pi}{M} + s \\ 0 & |\theta| \geq \frac{\pi}{M} + s \end{cases} \quad (24)$$


---

where the  $s$  parameter is half the angular width of the overlap between the posterior probabilities of adjacent neurons on the unit circle, in which case  $\frac{\pi}{M} \leq s \leq \frac{2\pi}{M}$  ensures that no more than 3 neurons can respond to a given input. Anticipating the optimum solution, a typical example of this type of posterior probability is shown in Figure 7.

In order to guarantee that  $\Pr(y|\mathbf{x})$  has a piecewise linear dependence on  $\mathbf{x}$ , the  $f_i(\theta)$  must have the sinusoidal dependence  $f_i(\theta) = a_i + b_i \cos \theta + c_i \sin |\theta|$  for  $i = 1, 2, 3$ . For those  $\mathbf{x}$  that lie on the unit circle, the  $a_i$ ,  $b_i$  and  $c_i$  parameters can be determined by imposing continuity on  $p(\theta)$  at  $\theta = -\frac{\pi}{M} + s$ ,  $\theta = \frac{3\pi}{M} - s$  and  $\theta = \frac{\pi}{M} + s$ , and normalisation of the total posterior probability such that  $f_1(\theta) + f_3(\frac{2\pi}{M} + \theta) + f_3(\frac{2\pi}{M} - \theta) = 1$  in the interval  $0 \leq \theta \leq -\frac{\pi}{M} + s$ , and  $f_2(\theta) + f_2(\frac{2\pi}{M} - \theta) = 1$  in the interval  $-\frac{\pi}{M} + s \leq \theta \leq \frac{3\pi}{M} - s$ . Also, to satisfy the stationarity conditions, set  $y = 0$  in Equation 9, and also set  $y = 0$  in Equation 10 in each of the intervals  $0 \leq \theta \leq -\frac{\pi}{M} + s$ ,  $-\frac{\pi}{M} + s \leq \theta \leq \frac{3\pi}{M} - s$  and  $\frac{3\pi}{M} - s \leq \theta \leq \frac{\pi}{M} + s$ . These conditions are sufficient to solve for the optimum  $f_i(\theta)$  for  $i = 1, 2, 3$ , the optimum  $r$ , and the optimum  $s$ .

The optimum  $f_i(\theta)$  are

$$\begin{aligned} f_1(\theta) &= -\frac{1}{4} \left( \cos\left(\frac{4\pi}{M} - s\right) + \cos s - 2 \cos\left(\frac{\pi}{M}\right) \cos \theta \right) \csc^2\left(\frac{\pi}{M}\right) \sec\left(\frac{2\pi}{M} - s\right) \\ f_2(\theta) &= \frac{1}{2} \left( \cot\left(\frac{\pi}{M}\right) \sec\left(\frac{2\pi}{M} - s\right) \sin\left(\frac{\pi}{M} - \theta\right) + 1 \right) \\ f_3(\theta) &= -\frac{1}{4} \csc^2\left(\frac{\pi}{M}\right) \left( \cos\left(\frac{3\pi}{M} - \theta\right) \sec\left(\frac{2\pi}{M} - s\right) - 1 \right) \end{aligned} \quad (25)$$

which correspond to different piecewise linear contributions to  $\Pr(y|\boldsymbol{x})$ . The  $f_1(\theta)$  piece has a gradient that points in the  $(1, 0)$  direction, the  $f_2(\theta)$  piece has a gradient that points in the  $(-\sin(\frac{\pi}{M}), \cos(\frac{\pi}{M}))$  direction, and the  $f_3(\theta)$  piece has a gradient that points in the  $(-\sin(\frac{3\pi}{M}), \cos(\frac{3\pi}{M}))$  direction. The optimum  $r$  is

$$r = \frac{n}{n-1} \frac{\cos(\frac{2\pi}{M} - s)}{\cos(\frac{\pi}{M})} \quad (26)$$

and the transcendental equation that must be satisfied by the optimum  $s$  (for  $M = 4$  this reduces to Equation 20) is

$$\frac{1}{n} \frac{\cos(\frac{2\pi}{M} - s)}{\cos(\frac{\pi}{M})} - \frac{n-1}{n} \frac{M}{\pi} \cos\left(\frac{\pi}{M}\right) \left( \sin\left(\frac{2\pi}{M} - s\right) - \left(\frac{2\pi}{M} - s\right) \cos\left(\frac{2\pi}{M} - s\right) \right) = 0 \quad (27)$$

and the minimum  $D_1 + D_2$ , which is obtained from the factorial encoder analogue of Equation 21 and Equation 22, may be simplified to give the result

$$D_1 + D_2 = \frac{n((n-1)(2\pi\frac{n-2}{n} - Ms) - \pi \sec^2(\frac{\pi}{M}))}{2(n-1)^2\pi} - \frac{n((n-1)(2\pi - Ms) + \pi \sec^2(\frac{\pi}{M})) \cos(\frac{4\pi}{M} - 2s)}{2\pi(n-1)^2} \quad (28)$$

As in Section III A, the limit  $n \rightarrow 1$  is well behaved because the limiting behaviour of the solution  $s$  of Equation 27 contains a factor  $n-1$  (see the asymptotic results in Section V) to cancel the  $\frac{1}{n-1}$  factor that appears in both Equation 26 and Equation 28. A typical example of the type of result that occurs when 3 posterior probabilities overlap is shown in Figure 7.

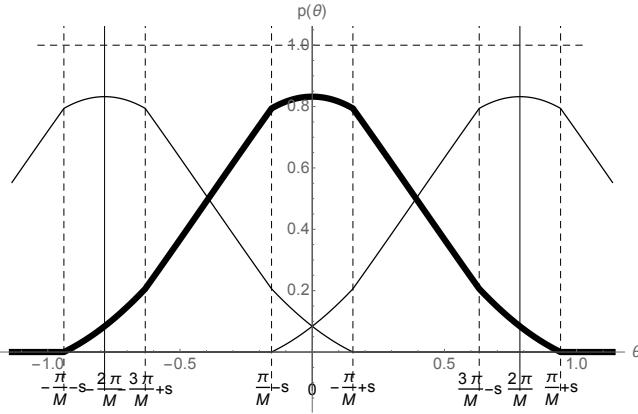


Figure 7: Plot of the optimal  $p(\theta)$  for  $M = 8$  and  $n = 100$ . The optimal value of  $s$  is  $s \approx 1.39\frac{\pi}{M}$ .

The results for the optimum value of  $s$  (i.e. Equation 20 and Equation 27) may be combined to yield the results shown in Figure 8. Asymptotically, as  $M \rightarrow \infty$  and  $n \rightarrow \infty$ , the contour  $s = \frac{\pi}{M}$  (the dashed line in Figure 8), which is the boundary between the regions where 2 and 3 posterior probabilities overlap, is given by  $n \approx 3\frac{M^2}{\pi^2}$  (see the asymptotic results in Section V).

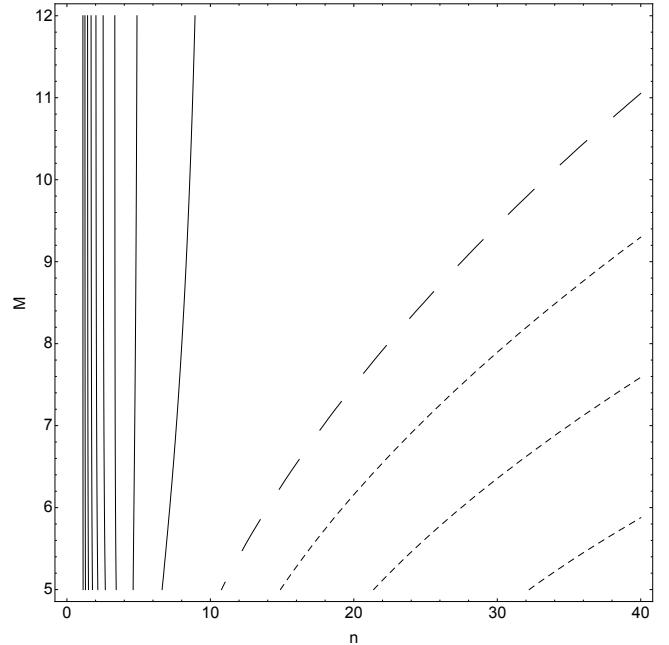


Figure 8: Contour plot of the optimum value of  $s$  versus  $(n, M)$  for encoding of a circular manifold. The solid contours are for the interval  $0 \leq s < \frac{\pi}{M}$ , the dotted contours are for  $\frac{\pi}{M} < s \leq \frac{2\pi}{M}$ , and the dashed contour is for  $s = \frac{\pi}{M}$ . The contours are all separated by intervals of  $\frac{\pi}{10M}$ .

#### IV. TOROIDAL MANIFOLD: FACTORIAL ENCODING

All of the results for factorial encoding of input data that lives on a toroidal manifold may be derived from the expression for  $D_1 + D_2$  in Equation 12 (and the corresponding stationarity conditions), with the appropriate replacements to ensure that the input manifold has data uniformly distributed on a 2-torus, and to ensure that

the solution is symmetric.

The posterior probability  $p(\theta)$  then has the same functional form as for a circular manifold, except that  $M$  is replaced by  $\frac{M}{2}$  because each of the two dimensions uses exactly half of the total of  $M$  neurons, so these results are not quoted explicitly here. The steps in the derivation of the optimum values of  $r$  and  $s$  and the minimum value of  $D_1 + D_2$  are analogous to the steps that appear in the derivation for a circular input manifold, and the results are sufficiently different from the ones that were obtained from a circular manifold that they are quoted explicitly here.

### A. 2 Overlapping Posterior Probabilities

The stationarity conditions yield the optimum  $r$  as

$$r = \frac{2n}{n-1} \frac{\sin s}{\sin(\frac{2\pi}{M})} \quad (29)$$

The transcendental equation that must be satisfied by the optimum  $s$  is

---


$$\frac{\sin s}{\sin(\frac{2\pi}{M})} - \frac{n-1}{n+1} \frac{M}{2\pi} \sin\left(\frac{2\pi}{M}\right) (\cos s + s \sin s) = 0 \quad (30)$$

The expression for the minimum  $D_1 + D_2$  is

$$D_1 + D_2 = 4 - \frac{n}{n-1} \frac{M}{2\pi} (2s + \sin(2s)) \quad (31)$$

### B. 3 Overlapping Posterior Probabilities

The stationarity conditions yield the optimum  $r$  as

$$r = \frac{2n}{n-1} \frac{\cos(\frac{4\pi}{M} - s)}{\cos(\frac{2\pi}{M})} \quad (32)$$

The transcendental equation that must be satisfied by the optimum  $s$  is

$$\frac{1}{n} \frac{\cos(\frac{4\pi}{M} - s)}{\cos(\frac{2\pi}{M})} - \frac{n-1}{2n} \frac{M}{2\pi} \cos\left(\frac{2\pi}{M}\right) \left( \sin\left(\frac{4\pi}{M} - s\right) - \left(\frac{4\pi}{M} - s\right) \cos\left(\frac{4\pi}{M} - s\right) \right) = 0 \quad (33)$$

The expression for the minimum  $D_1 + D_2$  is

$$D_1 + D_2 = \frac{n((n-1)(4\pi\frac{n-2}{n} - Ms) - 4\pi \sec^2(\frac{2\pi}{M}))}{2\pi(n-1)^2} - \frac{n((n-1)(4\pi - Ms) + 4\pi \sec^2(\frac{2\pi}{M})) \cos(\frac{8\pi}{M} - 2s)}{2\pi(n-1)^2} \quad (34)$$


---

The results for the optimum value of  $s$  (i.e. Equation 30 and Equation 33) may be combined to yield the results shown in Figure 9.

## V. ASYMPTOTIC RESULTS

Anticipating the results that are summarised in Figure 10, the asymptotic behaviour as  $M \rightarrow \infty$  lies in the region where 2 posterior probabilities overlap, and the asymptotic behaviour as  $n \rightarrow \infty$  lies in the region where 3 posterior probabilities overlap, so care must be taken to use the appropriate results when deriving the various asymptotic approximations below. The boundary between the regions where 2 or 3 posterior probabili-

ties overlap can be obtained for a circular input manifold by putting  $s = \frac{\pi}{M}$  in Equation 20 (or  $s = \frac{2\pi}{M}$  in Equation 30 in the case of a toroidal input manifold), and as  $M \rightarrow \infty$  this is given by

$$n \approx \begin{cases} 3 \frac{M^2}{\pi^2} & \text{circular manifold} \\ \frac{3}{2} \frac{M^2}{\pi^2} & \text{toroidal manifold (factorial encoding)} \end{cases} \quad (35)$$

As  $M \rightarrow \infty$  the asymptotic behaviour of  $D_1 + D_2$  for a circular input manifold may be obtained by asymptotically expanding the  $s$  dependence of Equation 20 (or Equation 30 in the case of a toroidal input manifold) in inverse powers of  $M$ , to yield

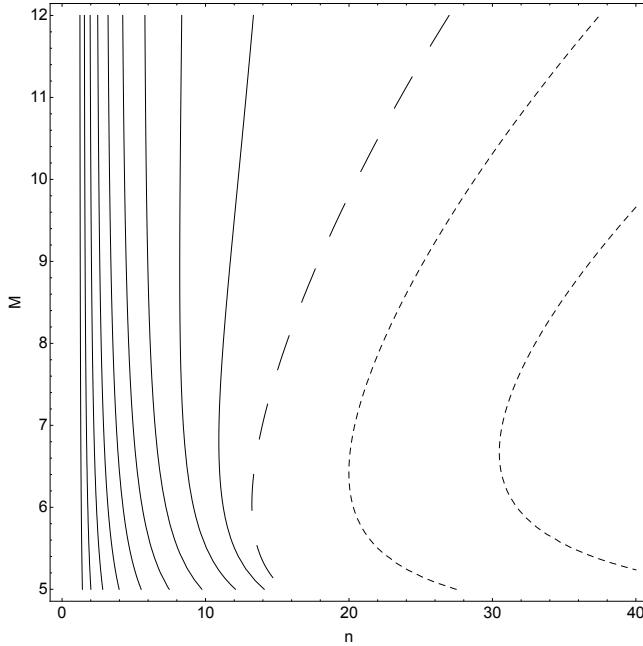


Figure 9: Contour plot of the optimum value of  $s$  versus  $(n, M)$  for factorial encoding of a toroidal manifold. The solid contours are for the interval  $0 \leq s < \frac{2\pi}{M}$ , the dotted contours are for  $\frac{2\pi}{M} < s \leq \frac{4\pi}{M}$ , and the dashed contour is for  $s = \frac{2\pi}{M}$ . The contours are all separated by intervals of  $\frac{\pi}{5M}$ .

$$s \approx \begin{cases} \frac{n-1}{n} \frac{\pi}{M} + \frac{(n-1)(n^2-4n+2)}{3n^3} \frac{\pi^3}{M^3} & \text{circular manifold} \\ \frac{n-1}{n+1} \frac{2\pi}{M} + \frac{(n-1)(n^2-6n+1)}{3(n+1)^3} \left(\frac{2\pi}{M}\right)^3 & \text{toroidal manifold (factorial encoding)} \end{cases} \quad (36)$$

and substituting this solution into the appropriate expression for  $r$  to obtain

$$r \approx \begin{cases} 1 + \frac{(2n^2-6n+3)}{6n^2} \frac{\pi^2}{M^2} & \text{circular manifold} \\ \frac{2n}{n+1} + \frac{8n(n^2-4n+1)}{3(n+1)^3} \frac{\pi^2}{M^2} & \text{toroidal manifold (factorial encoding)} \end{cases} \quad (37)$$

and substituting this solution into the appropriate expression for  $D_1 + D_2$  to obtain

$$D_1 + D_2 \approx \begin{cases} \frac{2(2n-1)}{3n^2} \frac{\pi^2}{M^2} & \text{circular manifold} \\ \frac{4}{n+1} + \frac{64n^2}{3(n+1)^3} \frac{\pi^2}{M^2} & \text{toroidal manifold (factorial encoding)} \end{cases} \quad (38)$$

The asymptotic result for a circular manifold may be used to determine the corresponding result for a linear manifold. Thus, if lengths are scaled so that the separation of the neurons (as measured around the circular manifold) becomes unity, which requires that all lengths are divided by  $\frac{2\pi}{M}$ , then asymptotically as  $M \rightarrow \infty$  the circular manifold solution becomes identical to the solution for a linear manifold with neurons separated by unit distance. Thus the optimum solution for a linear manifold with neurons separated by unit distance is  $s = \frac{n-1}{2n}$  and  $D_1 + D_2 = \frac{2n-1}{6n^2}$  (note that  $D_1 + D_2$  has the dimensions of  $(\text{length})^2$ ).

As  $n \rightarrow 1$  (i.e. the LBG vector quantiser limit) the asymptotic behaviour of  $D_1 + D_2$  for a circular input manifold may be obtained by expanding the  $s$  dependence of Equation 20 about the point  $s = 0$  (or Equation 30 about the point  $s = 0$  for a toroidal input manifold), to yield

$$s \approx \begin{cases} (n-1) \frac{M}{\pi} \sin^2\left(\frac{\pi}{M}\right) & \text{circular manifold} \\ \frac{(n-1)}{2} \frac{M}{2\pi} \sin^2\left(\frac{2\pi}{M}\right) & \text{toroidal manifold (factorial encoding)} \end{cases} \quad (39)$$

which gives  $s = 0$  when  $n = 1$ , so there is no overlap between the posterior probabilities for different neurons, as would be expected in a vector quantiser where only one neuron is allowed to fire. Substitute this solution into the

appropriate expression for  $r$  to obtain at  $n = 1$

$$r \approx \begin{cases} \frac{M}{\pi} \sin\left(\frac{\pi}{M}\right) & \text{circular manifold} \\ \frac{M}{2\pi} \sin\left(\frac{2\pi}{M}\right) & \text{toroidal manifold (factorial encoding)} \end{cases} \quad (40)$$

which is the distance of the centroid of an arc of the unit circle (with angular length  $\frac{2\pi}{M}$  for a circular manifold, or angular length  $\frac{4\pi}{M}$  for a toroidal manifold) from the origin, as expected for a network in which only one neuron can fire. So the best reconstruction is the centroid of the inputs that could have caused the single firing event. These results may be substituted into the appropriate expression for  $D_1 + D_2$  to obtain at  $n = 1$

$$D_1 + D_2 \approx \begin{cases} 2 - 2\left(\frac{M}{\pi}\right)^2 \sin^2\left(\frac{\pi}{M}\right) & \text{circular manifold} \\ 4 - 2\left(\frac{M}{2\pi}\right)^2 \sin^2\left(\frac{2\pi}{M}\right) & \text{toroidal manifold (factorial encoding)} \end{cases} \quad (41)$$

These results for  $D_1 + D_2$  have a simple geometrical interpretation. For a circular manifold  $D_1 + D_2$  is twice the average squared distance from an arc with angular length  $\frac{2\pi}{M}$  to its associated reference vector, which is exactly what would be expected. For a toroidal manifold  $D_1 + D_2$  is the same result with  $M \rightarrow \frac{M}{2}$ , plus an extra contribution of 2, because a factorial encoder with only 1 firing event acts as a conventional encoder using  $\frac{M}{2}$  neurons for the circular dimension that is fortunate enough to be associated with the firing event (hence the first contribution to  $D_1 + D_2$ ), and acts as no encoder at all for the other circular dimension which is associated with no firing events (hence the extra contribution of 2 to  $D_1 + D_2$ ).

As  $n \rightarrow \infty$  the asymptotic behaviour of  $D_1 + D_2$  for a circular input manifold may be obtained by expanding the  $s$  dependence of Equation 27 about the point  $s = \frac{2\pi}{M}$  (or Equation 33 about the point  $s = \frac{4\pi}{M}$  for a toroidal input manifold), to yield

$$s \approx \begin{cases} \frac{2\pi}{M} - \left(\frac{3\pi}{Mn \cos^2\left(\frac{\pi}{M}\right)}\right)^{\frac{1}{3}} & \text{circular manifold} \\ \frac{4\pi}{M} - \left(\frac{12\pi}{Mn \cos^2\left(\frac{2\pi}{M}\right)}\right)^{\frac{1}{3}} & \text{toroidal manifold (factorial encoding)} \end{cases} \quad (42)$$

where the limiting values of  $s$  as  $n \rightarrow \infty$  (i.e.  $s \rightarrow \frac{2\pi}{M}$  for a circular manifold, and  $s \rightarrow \frac{4\pi}{M}$  for a toroidal manifold) stops just short of allowing 4 or more posterior probabilities to overlap. In this limit  $D_1 = 0$ , so for a circular manifold the network acts as a PCA encoder (see the discussion after Equation 8) whose expansion coefficients sum to unity. In order to encode vectors on a unit circle without error 3 basis vectors are required (3 = 2 + 1 basis vectors are required in order that there are 2 independent expansion coefficients). This is the reason why it is sufficient to consider no more than 3 overlapping posterior probabilities for encoding data that lives in a 2-dimensional manifold (this argument generalises straightforwardly to higher dimensions). The same argument applies to the case of factorial encoding of a toroidal manifold. Substitute this solution into the appropriate expression for  $r$  to obtain

$$r \approx \begin{cases} \frac{1}{2} \sec\left(\frac{\pi}{M}\right) \left(2 - \left(\frac{3\pi}{Mn \cos^2\left(\frac{\pi}{M}\right)}\right)^{2/3}\right) & \text{circular manifold} \\ \sec\left(\frac{2\pi}{M}\right) \left(2 - \left(\frac{12\pi}{Mn \cos^2\left(\frac{2\pi}{M}\right)}\right)^{2/3}\right) & \text{toroidal manifold (factorial encoding)} \end{cases} \quad (43)$$

and substitute these results into the appropriate expression for  $D_1 + D_2$  to obtain

$$D_1 + D_2 \approx \begin{cases} \frac{2}{n} \tan^2\left(\frac{\pi}{M}\right) & \text{circular manifold} \\ \frac{4}{n} \left(2 \sec^2\left(\frac{2\pi}{M}\right) - 1\right) & \text{toroidal manifold (factorial encoding)} \end{cases} \quad (44)$$

Thus as  $n \rightarrow \infty$  it is possible to derive a value of  $M$  for which the asymptotic  $D_1 + D_2$  is the same for joint and factorial encoding of a toroidal manifold. This value of  $M$  must satisfy  $\frac{2}{n} \tan^2\left(\frac{\pi}{\sqrt{M}}\right) = \frac{4}{n} \left(2 \sec^2\left(\frac{2\pi}{M}\right) - 1\right)$ , which yields  $M \approx 11.74$ .

## VI. APPROXIMATE THE POSTERIOR PROBABILITY

A posterior probability may always be written in the form

$$\Pr(y|\mathbf{x}) = \frac{Q(\mathbf{x}|y)}{\sum_{y'=0}^{M-1} Q(\mathbf{x}|y')} \quad (45)$$

where  $Q(\mathbf{x}|y) \geq 0$  (with  $Q(\mathbf{x}|y) > 0$  for at least one value of  $y$  for each  $\mathbf{x}$ ). If the neurons behaved in such a way that they produced independent Poissonian firing events in response to a given input, then  $Q(\mathbf{x}|y)$  would be the firing rate (or activation function) of neuron  $y$  in response to input  $\mathbf{x}$ .

The optimum solution  $p(\theta)$  (as given in Equation 16 and Equation 17) may be approximated on the unit circle (i.e.  $\mathbf{x} = (\cos \theta, \sin \theta)$ ) by defining  $Q(\mathbf{x}|y)$  as

$$\begin{aligned} Q(\mathbf{x}|y) &= \begin{cases} \mathbf{w} \cdot \mathbf{x} - a & \mathbf{w} \cdot \mathbf{x} \geq a \\ 0 & \mathbf{w} \cdot \mathbf{x} \leq a \end{cases} \\ \mathbf{w} &= \left( \cos \frac{2\pi y}{M}, \sin \frac{2\pi y}{M} \right) \\ a &= \cos \frac{\pi}{M} - \sin \frac{\pi}{M} \sin s \end{aligned} \quad (46)$$

where  $a$  is a threshold parameter, and  $\mathbf{w}$  is a unit weight vector. This is the form of the neural activation function that is used in [12]. This leads to a good approximation to the optimum solution  $p(\theta)$  because

$$p(\theta) = \begin{cases} 0 & \theta \leq -\frac{\pi}{M} - s \\ \frac{Q(\mathbf{x}|y=0)}{Q(\mathbf{x}|y=0)+Q(\mathbf{x}|y=M-1)} + \mathcal{O}((\theta + \frac{\pi}{M})^3) & -\frac{\pi}{M} - s \leq \theta \leq -\frac{\pi}{M} + s \\ 1 & -\frac{\pi}{M} + s \leq \theta \leq \frac{\pi}{M} - s \\ \frac{Q(\mathbf{x}|y=0)}{Q(\mathbf{x}|y=0)+Q(\mathbf{x}|y=1)} + \mathcal{O}((\theta - \frac{\pi}{M})^3) & \frac{\pi}{M} - s \leq \theta \leq \frac{\pi}{M} + s \\ 0 & \theta \geq \frac{\pi}{M} + s \end{cases} \quad (47)$$

This approximation works well because curved input manifolds can be optimally encoded by using appropriate hyperplanes (as defined in Equation 46) to slice off pieces of the manifold.

This approximation breaks down as  $M \rightarrow \infty$ , as can be seen by inspecting the series expansion of  $p(\theta)$  near  $\theta = \frac{\pi}{M}$ .

$$p(\theta) = \begin{cases} \frac{1}{2} - \frac{1}{2 \sin s} (\theta - \frac{\pi}{M}) + \frac{1}{12 \sin s} (\theta - \frac{\pi}{M})^3 + \mathcal{O}((\theta - \frac{\pi}{M})^4) & \text{exact} \\ \frac{1}{2} - \frac{1}{2 \sin s} (\theta - \frac{\pi}{M}) + \frac{1}{12} \left( \frac{1}{\sin s} - \frac{3}{\tan \frac{\pi}{M} \sin^2 s} \right) (\theta - \frac{\pi}{M})^3 + \mathcal{O}((\theta - \frac{\pi}{M})^4) & \text{approx} \end{cases} \quad (48)$$

which differ in the  $\mathcal{O}((\theta - \frac{\pi}{M})^3)$  term. In the limit  $M \rightarrow \infty$  the half-width parameter  $s$  behaves like  $M^{-1}$ , so the  $\mathcal{O}((\theta - \frac{\pi}{M})^3)$  term behaves like  $M(\theta - \frac{\pi}{M})^3$  in the exact case, and  $M^3(\theta - \frac{\pi}{M})^3$  in the approximate case (because of the contribution from the  $\frac{3}{\tan \frac{\pi}{M} \sin^2 s}$  term). As  $M \rightarrow \infty$  each neuron responds to a progressively smaller angular range of inputs on the unit circle, so from the point of view of each neuron the curvature of the input manifold becomes negligible (i.e. the input manifold appears to more and more closely approximate a straight line), which ultimately makes it impossible to use hyperplanes to slice off pieces of the manifold. In the  $M \rightarrow \infty$  limit, a better approximation to the posterior probability would be to use ball-shaped regions (e.g. a radial basis function network) to cut up the input manifold into pieces.

## VII. JOINT VERSUS FACTORIAL ENCODING

The above results may be used to deduce when a factorial encoder is favoured with respect to a joint encoder (for input data that lives on a 2-torus). Firstly, Equation 20 (with the replacement  $M \rightarrow \sqrt{M}$ , and set-

ting  $s = \frac{\pi}{\sqrt{M}}$ ) may be used to deduce the region of the  $(n, M)$  plane where joint encoding of a 2-torus involves no more than 2 overlapping posterior probabilities, and Equation 30 (with  $s = \frac{2\pi}{M}$ ) may be used to deduce the corresponding result for factorial encoding of a 2-torus. Once these regions have been established, it is then possible to decide which of Equation 23 or Equation 28 (with  $M \rightarrow \sqrt{M}$  and then multiplied overall by 2) to use to calculate  $D_1 + D_2$  in the case of joint encoding a 2-torus, and which of Equation 31 or Equation 34 to use to calculate  $D_1 + D_2$  in the case of factorial encoding a 2-torus. These results are gathered together in Figure 10.

The need to derive results where up to 3 posterior probabilities overlap (which involves a large amount of algebra) is clear from the results shown in Figure 10, where it may be seen that most of the region where the factorial encoder is favoured with respect to the joint encoder has up to 3 overlapping posterior probabilities. The degree to which a factorial encoder is favoured with respect to a joint coder may be seen in Figure 11.

If the number of neurons  $M$  is restricted (i.e.  $M \lesssim 11.74$ ), then the joint encoding scheme in which the 2-torus is encoded using small response regions as shown in Figure 4, is usually not as good as the factorial encoding scheme in which the 2-torus is encoded using the inter-

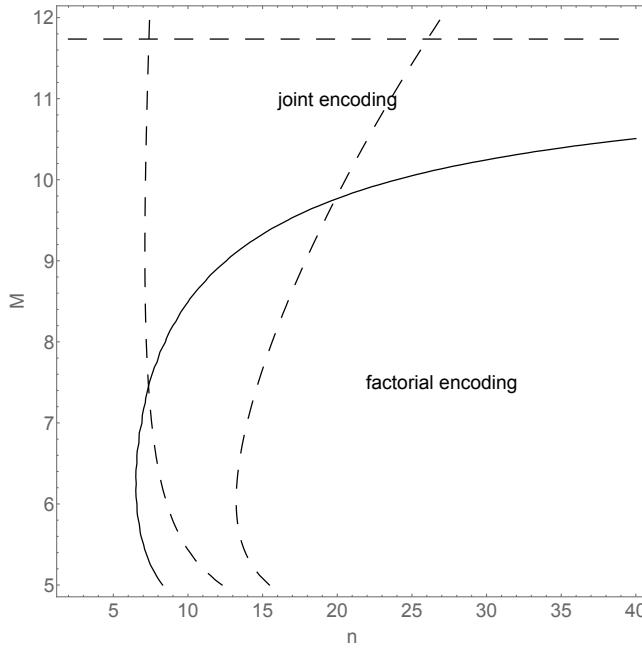


Figure 10: The diagram shows various results pertaining to joint and factorial encoding of a 2-torus. The solid line is the boundary between the regions of the  $(n, M)$  plane where joint or factorial encoding are favoured, and the horizontal dashed line is the asymptotic limit  $M \approx 11.74$  of this boundary as  $n \rightarrow \infty$ . The left hand dashed line is the boundary between the regions where 2 or 3 overlapping posterior probabilities occur in joint encoding, and the right hand dashed line is the corresponding boundary for factorial encoding.

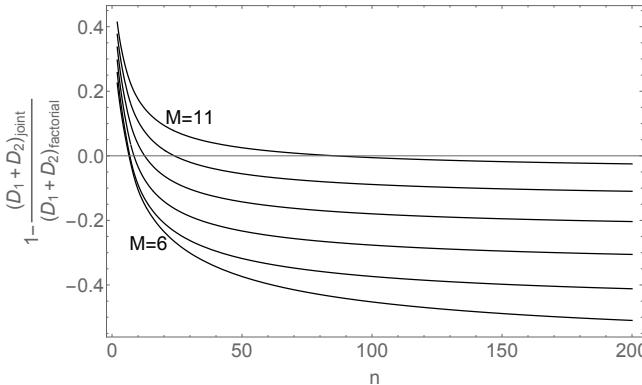


Figure 11: Plots for  $M = 6, 7, 8, 9, 10, 11$  of  $(D_1 + D_2)_{\text{factorial}} - (D_1 + D_2)_{\text{joint}}$  in units in which  $(D_1 + D_2)_{\text{factorial}} = 1$ . This makes it clear that the degree to which a factorial encoder is favoured with respect to a joint encoder is quite significant for large  $n$ .

section of pairs of elongated response regions as shown in Figure 5. The reason for this is that the factorial encoder can achieve the same resolution on the 2-torus using fewer neurons than the joint encoder (or, equivalently, a finer resolution using the same number of neurons). However, this does require that the number of firing events  $n$  is

sufficiently large that both subsets of  $\frac{M}{2}$  neurons in the factorial encoder are virtually guaranteed to each receive at least 1 firing event, so that they can approximate the input vector by the intersection of a pair of response regions.

If the number of neurons  $M$  is large (i.e.  $M \gtrsim 11.74$ ), then the joint encoding scheme is always favoured with respect to the factorial encoding scheme, because there are sufficient neurons to encode the 2-torus well using small response regions, as shown in Figure 4. This includes the limiting case  $M \rightarrow \infty$ , where the curvature of the input manifold is not visible to each neuron separately (because each neuron then responds to only an infinitesimally small angular interval), which implies that joint encoding is always favoured when the input manifold is planar.

Although not presented here, these results generalise readily to higher dimensional toruses, where factorial encoding is even more favoured, because (roughly speaking) the number of neurons required to do joint encoding with a given resolution increases exponentially with the dimensionality of the input, whereas the number of neurons required to do factorial encoding with a given resolution increases linearly with the dimensionality of the input.

## VIII. CONCLUSIONS

The goal of this chapter is to demonstrate that, by minimising an objective function that measures how well a network performs as an encoder, a network will find that brute-force encoding in which the network acts as a single encoder module which encodes the entire input space (joint encoding) is not necessarily the best solution. Rather, the network can self-organise into a number of encoder modules, each of which encodes only a subspace of the input (factorial encoding). The particular conditions under which factorial encoding is favoured with respect to joint encoding arise when the input data is derived from a curved input manifold, provided that the number of neurons is not too large, and provided that the number of observed neural firing events is large enough.

Factorial encoding does not emerge when the input manifold is insufficiently curved, or equivalently when there are too many neurons (more than approximately 12 in the case of a 2-torus), because then each neuron does not have a sufficiently large response region to be aware of the manifold's curvature. Under these conditions joint encoding is favoured with respect to factorial encoding. The need for the input manifold to be sufficiently curved is not really a restriction, because, in the case of image processing applications, only highly contrived data will lie on a manifold with low curvature. For instance, the set of images generated by placing a Gaussian blob at random in a 2-dimensional image is curved in precisely the way that is required for the analysis contained in this chapter to be applied.

Self-organised factorial encoding allows the input data to be encoded using a much smaller number of neurons than would be the case if joint encoding were used. Because only a small number of neurons is used, a factorial encoding scheme must be succinct, so it has to abstract the underlying degrees of freedom in the input manifold; this is a very useful side-effect of factorial encoding.

The main simplification that makes all these calculations possible is that, in an optimal neural network (i.e. one that minimises the assumed objective function) the form for the posterior probability is a piecewise linear function of the input vector. This leads to an enormous simplification in the mathematics, because only the space of piecewise linear functions needs to be searched for the optimal solution, rather than the whole space of functions (subject to normalisation and non-negativity constraints).

All of the results reported here are for a network whose input is well-approximated by inputs that are derived from a toroidal manifold (specifically, a 2-torus). For image processing applications, one interpretation is that

each input to the network is an image with a limited size (see the introductory section). This may be generalised to inputs derived from higher dimensional toruses, one interpretation of which is that each input is now an image containing multiple embedded target images, with each such image existing within a window of limited size. The factorial encoder solution becomes even more favourable with respect to the joint encoder solution as the number of targets increases, so the optimal network is a number of subnetworks each of which receives its input from a window of limited size within which there is typically one target.

A convenient approximation to this type of factorial encoder is the partitioned mixture distribution (PMD) network [5], in which the individual subnetworks in the factorial encoder network are constrained to share parameters, which thus leads to an upper bound on the minimum value of the objective function that would have ideally been obtained with the unconstrained factorial encoder network.

- [1] N Farvardin, *A study of vector quantisation for noisy channels*, IEEE Transactions on Information Theory **36** (1990), no. 4, 799–809.
- [2] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [3] H Kumazawa, M Kasahara, and T Namekawa, *A construction of vector quantisers for noisy channels*, Electronics and Engineering in Japan **67** (1984), no. 4, 39–47.
- [4] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [5] S P Luttrell, *Partitioned mixture distribution: an adaptive Bayesian network for low-level image processing*, IEE Proceedings on Vision, Image, and Signal Processing **141** (1994), no. 4, 251–260.
- [6] ———, *A discrete firing event analysis of the adaptive cluster expansion network*, Network: Computation in Neural Systems **7** (1996), no. 2, 285–290.
- [7] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
- [8] ———, *Self-organisation of multiple winner-take-all neural networks*, Connection Science **9** (1997), no. 1, 11–30.
- [9] ———, *Some theoretical properties of a network of discretely firing neurons*, (1998).
- [10] J Rissanen, *Modelling by shortest data description*, Automatica **14** (1978), no. 5, 465–471.
- [11] C E Shannon and W Weaver, *The mathematical theory of communication*, University of Illinois Press, Urbana, 1949.
- [12] C J S Webber, *Self-organisation of transformation-invariant detectors for constituents of perceptual patterns*, Network: Computation in Neural Systems **5** (1994), no. 4, 471–496.
- [13] S Wolfram, *The Mathematica book*, Wolfram Media and Cambridge University Press, 1996.



# An Adaptive Network for Encoding Data Using Piecewise Linear Functions \*

S P Luttrell<sup>†</sup>

DERA, St Andrews Rd, Malvern, Worcs, WR14 3PS, United Kingdom

An objective function that encourages an encoder to have the minimum overall Euclidean reconstruction error is shown to lead to encoders that can be implemented using functions that depend only in a piecewise linear fashion on the input vector. From the neural network viewpoint, the optimal form of the probability that each neuron is the next one to fire is a piecewise linear function of the input vector.

## I. INTRODUCTION

The general problem that is addressed in this paper is the encoding, transmission, and collation of data from multiple sources. The approach used in this paper to finding a solution to this problem is to define a basic encoder module, whose properties, as specified by its objective function, are such that it yields known types of encoder in simple cases, and also yields new types of encoder in more complicated cases.

This approach was introduced in [3]. It can be viewed as a 2-layer neural network in which the neurons in the output layer independently emit firing events in response to the vector presented in the input layer, and the locations of a finite number of these firing events constitutes the encoded version of the input. The standard vector quantiser [1] emerges as a special case where the code consists of only one firing event. In [5] it was shown how a factorial encoder (effectively, multiple parallel vector quantisers) could arise as an optimal solution when the approach in [3] was applied to input data that lives on a curved manifold.

This paper is concerned specifically with proving that the optimal encoder can be implemented using functions that depend only in a piecewise linear fashion on the input vector. Alternatively, from the 2-layer neural network viewpoint, the optimal form of the probability that each neuron is the next one to fire is a piecewise linear function of the input vector.

In the Section II the case of a single encoder is considered, and in Section III the case where two or more encoders are linked together acyclically is briefly considered.

## II. BASIC ENCODER

In this section the basic encoding model will be reviewed (see [3]), in which an input vector is encoded as

a vector of code indices, and then decoded as a reconstruction vector which approximates the input vector. In order to optimise this encoder an objective function will be presented, and a stationarity condition will be derived whose solution is the optimal choice of encoder. Finally, the key result will be derived, in which it is shown that the optimal encoder has a piecewise linear dependence on the input vector.

### A. Encoding/Decoding Model

The encoding/decoding model is an autoencoder in which a continuous-valued input vector  $\mathbf{x}$  with PDF  $\Pr(\mathbf{x})$  is encoded to produce a discrete-valued output vector  $\mathbf{y}$  with probability  $\Pr(\mathbf{y}|\mathbf{x})$ , which is finally decoded to produce a reconstruction vector  $\mathbf{x}'$  with PDF  $\Pr(\mathbf{x}'|\mathbf{y})$ . The average of the Euclidean distortion  $\|\mathbf{x} - \mathbf{x}'\|^2$  caused by the sequence of transformations  $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}'$  is given by [2]

$$D = \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \int d\mathbf{x}' \Pr(\mathbf{x}'|\mathbf{y}) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (2.1)$$

which may be simplified to [2]

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (2.2)$$

with the proviso that  $\mathbf{x}'(\mathbf{y})$  should be chosen so that it minimises  $D$ , giving  $\mathbf{x}'(\mathbf{y}) = \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ .

A simplification of this encoding model can be made by assuming that the  $\mathbf{y}$  vector has  $n$  components, each of which is independently generated from  $\mathbf{x}$ , and that the functional form of  $\mathbf{x}'(\mathbf{y})$  is a linear superposition of contributions from each of the components of  $\mathbf{y}$ . This may be summarised as

$$\begin{aligned} \mathbf{y} &= (y_1, y_2, \dots, y_n) \\ \Pr(\mathbf{y}|\mathbf{x}) &= \Pr(y_1|\mathbf{x}) \Pr(y_2|\mathbf{x}) \dots \Pr(y_n|\mathbf{x}) \\ \mathbf{x}'(\mathbf{y}) &= \frac{1}{n} (\mathbf{x}'(y_1) + \mathbf{x}'(y_2) + \dots + \mathbf{x}'(y_n)) \end{aligned} \quad (2.3)$$

where the  $y_i$  are integers satisfying  $1 \leq y_i \leq M$ . Unfortunately, the linear superposition assumption prevents  $\mathbf{x}'(\mathbf{y})$  from minimising  $D$ , so the value of  $D$  is now greater than given in its original definition in Equation 2.1. The

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

This appeared in Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN99), Edinburgh, 198-203, 1999. © BRITISH CROWN COPYRIGHT 1999/DERA.

<sup>†</sup>Electronic address: luttrell@signal.dera.gov.uk

linear superposition assumption therefore yields an upper bound on  $D$  in the form  $D \leq D_1 + D_2$  [3], where  $D_1$  and  $D_2$  are defined as

$$\begin{aligned} D_1 &= \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \quad (2.4) \\ D_2 &= \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 \end{aligned}$$

Minimising  $D_1 + D_2$  leads to a least upper bound on  $D$ . This type of solution will not be optimal from the point of view of  $D$  as originally defined in Equation 2.1, but it will be close to being optimal in cases where the linear superposition assumption in Equation 2.3 closely approximates the functional form of the  $\mathbf{x}'(\mathbf{y})$  that would actually have minimised  $D$  in Equation 2.2.

This encoding/decoding model may be viewed as a generalised form of the minimum Euclidean reconstruction error objective function used in [1], in which a stochastic encoder  $\Pr(y|\mathbf{x})$  is used to generate  $n$  code indices independently, and the reconstruction is a linear superposition of code vectors (the case  $n = 1$  reduces to the encoder discussed in [1]).

This encoding/decoding model has some similarities to the cooperative vector quantisation model introduced in [7]. In [7], *multiple* stochastic encoders are each used to

generate a distributed coded version of the input, and then a linear superposition is used to reconstruct the input from all of these distributed codes. However in this paper, a *single* stochastic encoder is used to generate a single distributed coded version of the input (i.e. the components  $(y_1, y_2, \dots, y_n)$ ), and then a linear superposition is used to reconstruct the input from this distributed code (i.e. as  $\frac{1}{n}(\mathbf{x}'(y_1) + \mathbf{x}'(y_2) + \dots + \mathbf{x}'(y_n))$ ). The motivation for using multiple encoders in [7] was to allow different encoders to encode different properties of the input. However, it has been shown in [5] how a single encoder together with the minimisation of an appropriate objective function (i.e.  $D_1 + D_2$ ) results in the single encoder automatically splitting into multiple encoders as required, each of which then encodes a different property of the input.

## B. Stationarity Condition

The stationarity condition for  $\Pr(y|\mathbf{x})$  will now be derived. Thus, functionally differentiate  $D_1 + D_2$  with respect to  $\log \Pr(y|\mathbf{x})$ , where logarithmic differentiation implicitly imposes the constraint  $\Pr(y|\mathbf{x}) \geq 0$ , and use a Lagrange multiplier term  $L \equiv \int d\mathbf{x} \lambda(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x})$  to impose the normalisation constraint  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$  for each  $\mathbf{x}$ , to obtain

---


$$\begin{aligned} \frac{\delta(D_1 + D_2 - L)}{\delta \log \Pr(y|\mathbf{x})} &= \frac{2}{n} \Pr(\mathbf{x}) \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \\ &\quad - \frac{4(n-1)}{n} \Pr(\mathbf{x}) \Pr(y|\mathbf{x}) \mathbf{x}'(y) \cdot \left( \mathbf{x} - \sum_{y'=1}^M \Pr(y'|\mathbf{x}) \mathbf{x}'(y') \right) \\ &\quad - \lambda(\mathbf{x}) \Pr(y|\mathbf{x}) \end{aligned} \quad (2.5)$$

where  $\frac{\delta U}{\delta V}$  denotes functional differentiation of  $U$  with respect to  $V$ . The stationarity condition implies that  $\sum_{y=1}^M \Pr(y|\mathbf{x}) \frac{\delta(D_1 + D_2 - L)}{\delta \log \Pr(y|\mathbf{x})} = 0$ , which may be used to determine the Lagrange multiplier function  $\lambda(\mathbf{x})$ . When  $\lambda(\mathbf{x})$  is substituted back into the stationarity condition itself, it yields

---


$$0 = \Pr(\mathbf{x}) \Pr(y|\mathbf{x}) \sum_{y'=1}^M (\Pr(y'|\mathbf{x}) - \delta_{y,y'}) \mathbf{x}'(y') \cdot \left( \left( \frac{\mathbf{x}'(y')}{2} - n\mathbf{x} \right) + (n-1) \sum_{y''=1}^M \Pr(y''|\mathbf{x}) \mathbf{x}'(y'') \right) \quad (2.6)$$


---

where  $\delta_{y,y'}$  is a Kronecker delta, which is defined as  $\delta_{y,y'} = 1$  if  $y = y'$ , and  $\delta_{y,y'} = 0$  if  $y \neq y'$ . There are several classes of solution to this stationarity condition, corresponding to one (or more) of the three factors in Equation 2.6 being zero.

Type 1:  $\Pr(\mathbf{x}) = 0$ . If the input PDF is zero at  $\mathbf{x}$ , then nothing can be deduced about  $\Pr(y|\mathbf{x})$ , because there are no input vectors to explore the network's response to this

$\mathbf{x}$ .

Type 2:  $\Pr(\mathbf{x}) > 0$  and  $\Pr(y|\mathbf{x}) = 0$ . The singularity in  $\log \Pr(y|\mathbf{x})$  when  $\Pr(y|\mathbf{x}) = 0$  is what causes this solution to emerge, and it blocks access to the inadmissible region  $\Pr(y|\mathbf{x}) < 0$ . Using the normalisation constraint  $\sum_{y'=1}^M \Pr(y'|\mathbf{x}) = 1$ , this solution ensures that  $0 \leq \Pr(y|\mathbf{x}) \leq 1$ , as expected.

Type 3:  $\Pr(\mathbf{x}) > 0$ ,  $\Pr(y|\mathbf{x}) > 0$  and  $\sum_{y'=1}^M (\Pr(y'|\mathbf{x}) - \delta_{y,y'}) \mathbf{x}'(y') \cdot \left( \left( \frac{\mathbf{x}'(y')}{2} - n\mathbf{x} \right) + (n-1) \sum_{y''=1}^M \Pr(y''|\mathbf{x}) \mathbf{x}'(y'') \right) = 0$

$\delta_{y,y'} \mathbf{x}'(y') \cdot (\dots) = 0$ . This is the most interesting solution, because it constrains the admissible  $\Pr(y'|\mathbf{x})$  and  $\mathbf{x}'(y')$  for  $y' = 1, 2, \dots, M$ .

### C. Piecewise Linear Optimal Solution

The  $\Pr(y|\mathbf{x})$  that minimise  $D_1 + D_2$  will now be shown to be piecewise linear functions of  $\mathbf{x}$ . The type 3 solution  $\sum_{y'=1}^M (\Pr(y'|\mathbf{x}) - \delta_{y,y'}) \mathbf{x}'(y') \cdot (\dots) = 0$  can be rearranged into the form

$$0 = (n-1) \|\mathbf{z}(\mathbf{x})\|^2 + \mathbf{a}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{x}) + \frac{z(\mathbf{x})}{2} + a(\mathbf{x}) \quad (2.7)$$

where the following definitions have been made

$$\begin{aligned} \mathbf{z}(\mathbf{x}) &\equiv \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \\ z(\mathbf{x}) &\equiv \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x}'(y)\|^2 \\ \mathbf{a}(\mathbf{x}) &\equiv -(n-1) \mathbf{x}'(y) - n \mathbf{x} \\ a(\mathbf{x}) &\equiv -\mathbf{x}'(y) \cdot \left( \frac{\mathbf{x}'(y)}{2} - n \mathbf{x} \right) \end{aligned} \quad (2.8)$$

This form is convenient because it separates the  $\mathbf{x}$ -dependence into parts which *do* depend on  $\Pr(y|\mathbf{x})$  (i.e.  $\|\mathbf{z}(\mathbf{x})\|^2$ ,  $\mathbf{z}(\mathbf{x})$  and  $z(\mathbf{x})$ ) and parts which do *not* depend on  $\Pr(y|\mathbf{x})$  (i.e.  $\mathbf{a}(\mathbf{x})$  and  $a(\mathbf{x})$ ). The  $\Pr(y|\mathbf{x})$  which satisfy Equation 2.7 must have an  $\mathbf{x}$ -dependence that exactly cancels the  $\mathbf{x}$ -dependence that arises from the  $\mathbf{a}(\mathbf{x})$  and  $a(\mathbf{x})$  coefficients.

The table below shows how the powers of  $\mathbf{x}$  that appear in  $\Pr(y|\mathbf{x})$  influence the powers of  $\mathbf{x}$  that appear in each of the terms of the righthand side of the type 3 stationarity condition in Equation 2.7. Thus the numerical values in this table are the powers of  $\mathbf{x}$  that arise when 3 specific choices for the  $\mathbf{x}$ -dependence of  $\Pr(y|\mathbf{x})$  are made. The column labelled “constant” refers to  $\Pr(y|\mathbf{x})$  having no  $\mathbf{x}$  dependence, so its first row contains 0. The column labelled “linear” refers to  $\Pr(y|\mathbf{x})$  having at most a linear  $\mathbf{x}$  dependence, so its first row contains 0, 1. The column labelled “quadratic” refers to  $\Pr(y|\mathbf{x})$  having at most a quadratic  $\mathbf{x}$  dependence, so its first row contains 0, 1, 2. Each of the other entries may be deduced by inserting the appropriate  $\mathbf{x}$  dependence for  $\Pr(y|\mathbf{x})$  into Equation 2.8, and then reading off the  $\mathbf{x}$  dependences of the various terms in Equation 2.7.

The functional form of the optimal  $\Pr(y|\mathbf{x})$  may then readily be deduced from the entries in this table, because the stationarity condition in Equation 2.7 can be satisfied only if the coefficients of each power of  $\mathbf{x}$  are separately zero. This assumes that  $\mathbf{x}$  ranges over a finite-sized domain, so that the different powers of  $\mathbf{x}$  can be separated from each other. The last 4 rows of the table contain the entries for the various terms that appear in Equation 2.7. For instance, if  $\Pr(y|\mathbf{x})$  is constant, then the entries in

Table I: The table below shows how the powers of  $\mathbf{x}$  that appear in  $\Pr(y|\mathbf{x})$  influence the powers of  $\mathbf{x}$  that appear in each of the terms of the righthand side of the type 3 stationarity condition in Equation 2.7. Thus the numerical values in this table are the powers of  $\mathbf{x}$  that arise when 3 specific choices for the  $\mathbf{x}$ -dependence of  $\Pr(y|\mathbf{x})$  are made. The column labelled “constant” refers to  $\Pr(y|\mathbf{x})$  having no  $\mathbf{x}$  dependence, so its first row contains 0. The column labelled “linear” refers to  $\Pr(y|\mathbf{x})$  having at most a linear  $\mathbf{x}$  dependence, so its first row contains 0, 1. The column labelled “quadratic” refers to  $\Pr(y|\mathbf{x})$  having at most a quadratic  $\mathbf{x}$  dependence, so its first row contains 0, 1, 2. Each of the other entries may be deduced by inserting the appropriate  $\mathbf{x}$  dependence for  $\Pr(y|\mathbf{x})$  into Equation 2.8, and then reading off the  $\mathbf{x}$  dependences of the various terms in Equation 2.7.

	constant	linear	quadratic
$\Pr(y \mathbf{x})$	0	0, 1	0, 1, 2
$\mathbf{z}(\mathbf{x})$ and $z(\mathbf{x})$	0	0, 1	0, 1, 2
$\ \mathbf{z}(\mathbf{x})\ ^2$	0	0, 1, 2	0, ..., 4
$\mathbf{a}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{x})$	0, 1	0, 1, 2	0, ..., 3
$a(\mathbf{x})$	0, 1	0, 1	0, 1

the first column of the table say that  $\mathbf{z}(\mathbf{x})$  and  $z(\mathbf{x}) = \text{constant}$ ,  $\|\mathbf{z}(\mathbf{x})\|^2 = \text{constant}$ ,  $\mathbf{a}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{x}) = \text{linear}$ , and  $a(\mathbf{x}) = \text{linear}$ , so the coefficients of 2 different powers of  $\mathbf{x}$  must be zero in Equation 2.7. Similarly, if  $\Pr(y|\mathbf{x})$  is linear, then the entries in the second column of the table say that  $\mathbf{z}(\mathbf{x})$  and  $z(\mathbf{x}) = \text{linear}$ ,  $\|\mathbf{z}(\mathbf{x})\|^2 = \text{quadratic}$ ,  $\mathbf{a}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{x}) = \text{quadratic}$ , and  $a(\mathbf{x}) = \text{linear}$ , so the coefficients of 3 different powers of  $\mathbf{x}$  must be zero in Equation 2.7. In both of these cases the  $\mathbf{x}'(y)$  dependence (see Equation 2.8) of each of the terms in Equation 2.7 may be used to achieve the desired cancellation, by appropriately adjusting  $\mathbf{x}'(y)$  (whilst ensuring that  $\mathbf{x}'(y)$  also satisfies its own stationarity condition). However, this trick will not work when  $\Pr(y|\mathbf{x})$  has a quadratic or higher dependence on  $\mathbf{x}$ . This problem is shown explicitly for the quadratic case in the last column of the table, where the quartic power of  $\mathbf{x}$  in the  $\|\mathbf{z}(\mathbf{x})\|^2$  term clearly has no partner to cancel it.

The conclusion from the above argument is that a type 3 solution for  $\Pr(y|\mathbf{x})$  can have at most a linear dependence on  $\mathbf{x}$ . Note that a type 2 solution is trivially linear in  $\mathbf{x}$ , because it is constant. Thus, if it is assumed that  $\Pr(\mathbf{x}) > 0$ , so that a type 1 solution does not occur, then all possible solutions (i.e. type 2 or type 3) for  $\Pr(y|\mathbf{x})$  can have at most a linear dependence on  $\mathbf{x}$ . For a given value of  $y$ , each region of  $\mathbf{x}$ -space where a type 3 solution occurs is bounded by hyperplanar surfaces where  $\Pr(y|\mathbf{x}) = 0$  or  $\Pr(y|\mathbf{x}) = 1$ ; these are the surfaces where the type 3 solution for  $\Pr(y|\mathbf{x})$  is about to leave its allowed range  $0 \leq \Pr(y|\mathbf{x}) \leq 1$ , so it must switch to a different solution. Furthermore, within each such region of  $\mathbf{x}$ -space there may be additional hyperplanar surfaces generated by the solutions  $\Pr(y'|\mathbf{x})$  for  $y' \neq y$ . Overall, the set of all of these hyperplanar surfaces (i.e. taking

all values of  $y$  into account) partitions  $\mathbf{x}$ -space into a set of polyhedra, within each of which  $\Pr(y|\mathbf{x})$  has at most a linear dependence on  $\mathbf{x}$ . This is the result we have been seeking, namely that  $\Pr(y|\mathbf{x})$  is a *piecewise linear* function of  $\mathbf{x}$ .

$\Pr(y|\mathbf{x})$  typically consists of a number of contiguous type 3 solutions joined together, surrounded by a region where  $\Pr(y|\mathbf{x}) = 0$  or  $\Pr(y|\mathbf{x}) = 1$ . In Figure 1 a typical solution for  $\Pr(y|\mathbf{x})$  is shown that demonstrates the properties discussed above. This type of solution can be obtained either analytically by assuming that  $\Pr(y|\mathbf{x})$  is piecewise linear and constructing an appropriately parameterised functional form of  $\Pr(y|\mathbf{x})$  then solving for the stationary point of  $D_1 + D_2$  with respect to these parameter values, or by numerically modelling

$\Pr(y|\mathbf{x})$  by exhaustively tabulating (a discretised version of)  $\Pr(y|\mathbf{x})$  then varying these table entries (subject to the constraints  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$  and  $0 \leq \Pr(y|\mathbf{x}) \leq 1$  for  $1 \leq y \leq M$ ) until  $D_1 + D_2$  is minimised. In both cases the  $\mathbf{x}'(y)$  must be simultaneously optimised analytically or numerically, respectively.

This piecewise linearity result can be seen to be intuitively reasonable by the following loose argument.  $D_1 + D_2$  can be written as (using that  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$ )

$$D_1 + D_2 = \int d\mathbf{x} \Pr(\mathbf{x}) D(\mathbf{x}, z(\mathbf{x}), \mathbf{z}(\mathbf{x})) \quad (2.9)$$

where

$$D(\mathbf{x}, z(\mathbf{x}), \mathbf{z}(\mathbf{x})) \equiv \frac{2(n-1)}{n} \|\mathbf{z}(\mathbf{x})\|^2 + \frac{2}{n} z(\mathbf{x}) - 4 \mathbf{x} \cdot \mathbf{z}(\mathbf{x}) + 2 \|\mathbf{x}\|^2 \quad (2.10)$$

where the  $\mathbf{x}$ -dependence has been separated into parts which *do* depend on  $\Pr(y|\mathbf{x})$  (i.e.  $\|\mathbf{z}(\mathbf{x})\|^2$ ,  $\mathbf{z}(\mathbf{x})$  and  $z(\mathbf{x})$ ) and parts which *do not* depend on  $\Pr(y|\mathbf{x})$  (i.e.  $\mathbf{x}$  and  $\|\mathbf{x}\|^2$ ). As with the stationarity condition in Equation 2.7, a  $\Pr(y|\mathbf{x})$  that contains quadratic, or higher, powers of  $\mathbf{x}$  in some finite-sized region would lead to terms in  $D_1 + D_2$  that had a power law dependence on  $\mathbf{x}$  different from that of all of the other terms, which would make it impossible for  $D_1 + D_2$  to be stationary with respect to variations of  $\Pr(y|\mathbf{x})$  in this region, which forces the  $\mathbf{x}$ -dependence of  $\Pr(y|\mathbf{x})$  to be linear in this region. The conclusion is that  $\Pr(y|\mathbf{x})$  has a piecewise linear  $\mathbf{x}$ -dependence.

### III. LINKED ENCODERS

The results of the Section II will now be generalised to the case where two or more encoders are linked together acyclically, such as might be used in the encoding, transmission, and collation of data from multiple sources. The components of the input vector to each encoder will be restricted to being a *linear* combination of posterior probabilities computed by all of the other encoders, plus a *linear* combination of the components of the original input vector, and also must respect the acyclic constraint. The objective function will be restricted to being a *linear* combination of the Euclidean distortions of the type given in Equation 2.1 for each of the encoders. This acyclic network of encoders (plus objective function) includes as a special case the single encoder/decoder that was discussed in the previous section.

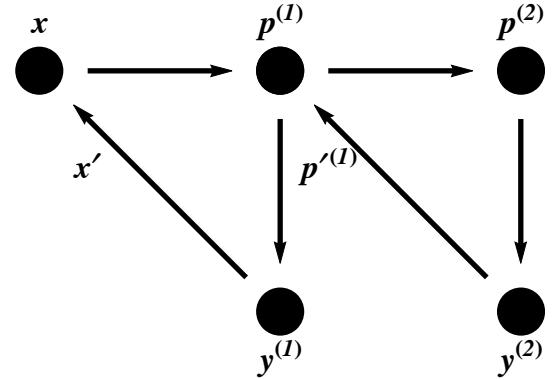


Figure 2: A chain of two linked encoders. The first encoder inputs  $\mathbf{x}$ , computes a vector  $\mathbf{p}^{(1)}$  of posterior probabilities (where  $p_i^{(1)} \equiv \Pr(y_i|\mathbf{x})$ ), draws an  $n^{(1)}$ -dimensional vector  $\mathbf{y}^{(1)}$  of independent samples from this probability distribution, and attempts to reconstruct the input vector from  $\mathbf{y}^{(1)}$ . The second encoder inputs  $\mathbf{p}^{(1)}$ , but otherwise operates analogously to the first encoder.

The simplest non-trivial example of two encoders is shown in Figure 2 and the objective function is a weighted sum of a pair of Euclidean distortions  $w^{(1)}D^{(1)} + w^{(2)}D^{(2)}$ , whose upper bound is  $w^{(1)}(D_1^{(1)} + D_2^{(1)}) + w^{(2)}(D_1^{(2)} + D_2^{(2)})$ . This will be shown to lead to optimal posterior probabilities that are each a piecewise linear function of the corresponding encoder's input. This result may readily be generalised to acyclically linked net-

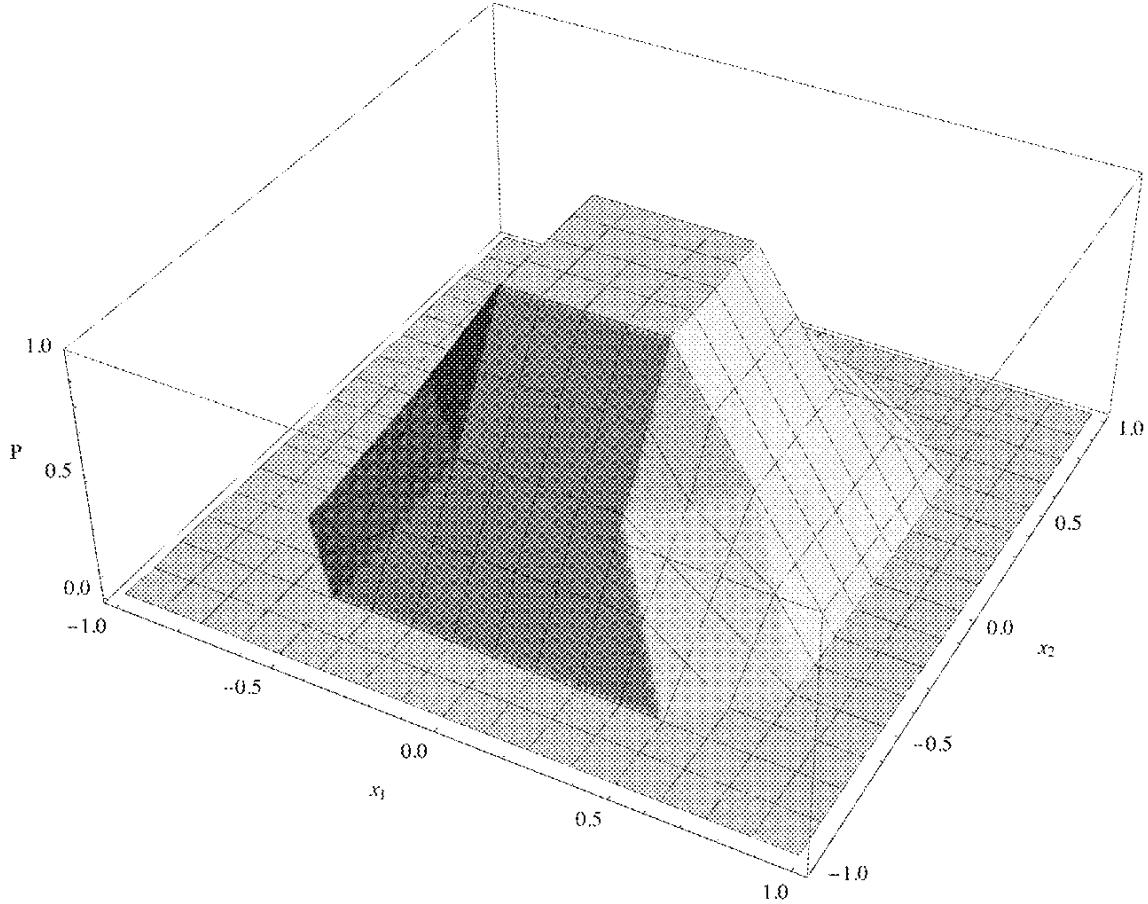


Figure 1: A typical example of a piecewise linear solution  $\Pr(y|\mathbf{x})$  to the stationarity condition. The input vector  $\mathbf{x}$  is 2-dimensional and  $\Pr(\mathbf{x})$  is constant over the whole input plane. The stationary solution is a uniform grid of overlapping  $\Pr(y|\mathbf{x})$ , such that for each  $\mathbf{x}$  the normalisation condition  $\sum_y \Pr(y|\mathbf{x})$  holds. For a given  $y$  the diagram shows how the various linear pieces of  $\Pr(y|\mathbf{x})$  link together.

works of more than two encoders.

Thus, using the techniques introduced in the previous section, the second encoder, which implements the sequence of transformations  $\mathbf{p}^{(1)} \rightarrow \mathbf{p}^{(2)} \rightarrow \mathbf{y}^{(2)} \rightarrow \mathbf{p}'^{(1)}$ , may be optimised by minimising  $D_1^{(2)} + D_2^{(2)}$  to produce a posterior probability vector  $\mathbf{p}^{(2)}$  whose elements are each a piecewise linear function of its input vector  $\mathbf{p}^{(1)}$ . Regions of  $\mathbf{p}^{(1)}$ -space for which  $\Pr(\mathbf{p}^{(1)}) = 0$ , which leads to a type 1 solution, may be ignored. If the second encoder were absent, then the first encoder, which implements the sequence of transformations  $\mathbf{x} \rightarrow \mathbf{p}^{(1)} \rightarrow \mathbf{y}^{(1)} \rightarrow \mathbf{x}'$ , could then be optimised by minimising  $D_1^{(1)} + D_2^{(1)}$  to produce a posterior probability vector  $\mathbf{p}^{(1)}$  whose elements were each a piecewise linear function of its input vector  $\mathbf{x}$ .

However, the presence of the second encoder introduces an additional complication, because the  $\mathbf{p}^{(1)}$  that we are trying to optimise appears not only in the first encoder's objective function, but also in the second encoder's objective function. Fortunately, this additional  $\mathbf{p}^{(1)}$ -dependence turns out to be benign. Thus,  $\mathbf{p}^{(2)}$  has

already been shown to be a piecewise linear function of  $\mathbf{p}^{(1)}$ , so the second encoder's objective function is a piecewise quadratic function of  $\mathbf{p}^{(1)}$  (see the discussion after Equation 2.9, but with  $\mathbf{x}$  replaced by  $\mathbf{p}^{(1)}$ ), which is the same form as the  $\mathbf{p}^{(1)}$ -dependence of the first encoder's objective function. Finally, if we now assume that  $\mathbf{p}^{(1)}$  is itself a piecewise linear function of  $\mathbf{x}$ , then both encoder's objective functions are piecewise quadratic functions of  $\mathbf{x}$ , so the  $\mathbf{x}$ -dependence of the total objective function  $w^{(1)}(D_1^{(1)} + D_2^{(1)}) + w^{(2)}(D_1^{(2)} + D_2^{(2)})$  is piecewise quadratic, which allows a stationary solution for  $\mathbf{p}^{(1)}$  to be found, using arguments that are analogous to those used after Equation 2.9.

#### IV. CONCLUSIONS AND DISCUSSION

In this paper it has been shown that, for arbitrary input PDFs, and using a generalised form of the minimum Euclidean reconstruction error objective function, the optimal form of the encoder depends in a piecewise linear fashion on the input vector. This piecewise linear prop-

erty holds also for acyclically linked encoders, subject to certain linearity constraints on the input vector to each encoder and on the overall objective function.

This piecewise linearity property makes it possible to analytically derive optimum encoders in cases where the input PDF has a simple structure. For instance, in [5] it is shown that if an encoder has input vectors that live on a toroidal manifold, then the optimum solution can be a factorial encoder, in which each underlying circular degree of freedom in the torus is encoded separately. Thus, unlike in [7], it is not necessary to assume the existence of multiple parallel encoders in cases where a factorial encoder would be appropriate, because the parallel encoders emerge by self-organisation during the optimisation process. Also, it was shown in [5] that, if the input data lies on a circular manifold, then the semilinear neural activation function proposed in [6] gives a good approximation to the functional form of the optimal encoder.

The context within which the results contained in this paper may be used is the problem of encoding, transmission, and collation of data from multiple sources. Thus the multiple sources comprise the external input to a chain of encoders, and then optimisation of the overall objective function leads to an optimal acyclic network, in which each link in the chain self-organises into whatever form is optimal (e.g. a factorial encoder).

Finally, it is possible to show that the optimisation of an acyclic network (e.g. a chain) of encoders is equivalent to the optimisation of a Gauss-Markov density model of the joint PDF of the input vectors to all of the encoders [4]. From the neural network viewpoint, this is equivalent to optimising a Gauss-Markov density model of the joint PDF of the states of all of the layers of a multilayer network. Unlike other approaches to density modelling, this approach treats all of the network layers on an equal footing.

- [1] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [2] S P Luttrell, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
- [3] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
- [4] ———, *A unified theory of density models and auto-encoders*, Preprint NI97039-NNM, Isaac Newton Institute for Mathematical Sciences, Cambridge, 1997.
- [5] ———, *Combining artificial neural nets: Ensemble and modular multi-net systems*, Perspectives in Neural Computing, ch. Self-organised modular neural networks for encoding data, pp. 235–263, Springer-Verlag, London, 1999.
- [6] C J S Webber, *Self-organisation of transformation-invariant detectors for constituents of perceptual patterns*, Network: Computation in Neural Systems **5** (1994), no. 4, 471–496.
- [7] R S Zemel, *A minimum description length framework for unsupervised learning*, Ph.D. thesis, Toronto University, 1993.

# The Emergence of Dominance Stripes and Orientation Maps in a Network of Firing Neurons \*

STEPHEN P. LUTTRELL

## I. INTRODUCTION

This chapter addresses the problem of training a self-organising neural network on images derived from multiple sources; this type of network potentially may be used to model the behaviour of the mammalian visual cortex (for a review of neural network models of the visual cortex see Swindale [9]). The network that will be considered is a soft encoder which transforms its input vector into a posterior probability over various possible classes (i.e. alternative possible interpretations of the input vector). This encoder will be optimised so that its posterior probability is able to retain as much information as possible about its input vector, as measured in the minimum mean square reconstruction error (i.e.  $L_2$  error) sense (Luttrell [4, 6]).

In the special case where the optimisation is performed over the space of all possible soft encoders, the optimum solution is a hard encoder (i.e. it is a “winner-take-all” network, in which only one of the output neurons is active) which is an optimal vector quantiser (VQ) of the type described in Linde et al. [3], for encoding the input vector with minimum  $L_2$  error. A more general case is where the output of the soft encoder is deliberately damaged by the effects of a noise process. This type of noisy encoder leads to an optimal self-organising map (SOM) for encoding the input vector with minimum  $L_2$  error, which is closely related to the well-known Kohonen map (Kohonen [2]).

The soft encoder network that is discussed in this chapter turns out to have many of the emergent properties that are observed in the mammalian visual cortex, such as dominance stripes and orientation maps. It will therefore be referred to as a Visual COortex Network (VICON). It differs from other visual cortex models (see the review in Swindale [9]) because it uses Bayesian methods to analyse the information contained in sets of neural firing events, where the neuron inputs are high-dimensional.

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.  
This paper appeared in Proceedings of the 2nd Workshop on Information Theory and the Brain, Newquay, 101-121, 1996. Information Theory and the Brain, edited by Roland Beddoe, Peter

images rather than vectors in a low-dimensional abstract space. Also the network structure is derived from first principles, rather than reverse-engineered from observations of the structure of the visual cortex.

The layout of this chapter is as follows. In Section II the network objective function is presented (in the Appendix its derivatives are presented and interpreted). In Section III the concepts of dominance stripes and orientation maps are explained. In Section IV the results of computer simulations are presented, where the effects of varying several parameters are explored. Both one- and two-dimensional retinae are considered, and in each of these cases both single retinae and pairs of retinae are considered.

## II. THEORY

This section summarises the theory of a two-layer VICON. The network objective function is introduced, and its first-order perturbation expansion is interpreted.

The derivatives of the objective function with respect to its parameters are presented and interpreted in Appendix VI.

### A. Objective Function

For a two-layer network with  $M$  output neurons, the network objective function is given by (Luttrell [6])

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \quad (1)$$

where the probability  $\Pr(y|\mathbf{x})$  that neuron  $y$  fires first is given by

Hancock, and Peter Földiák. Copyright © 1999 Cambridge University Press. Printed in the United States of America. All rights reserved.

$$\Pr(y|\mathbf{x}) = \frac{1}{M} \sum_{y'=1}^M \Pr(y|y') Q(\mathbf{x}|y') \sum_{y'' \in \mathcal{N}^{-1}(y')} \frac{1}{\sum_{y''' \in \mathcal{N}(y'')} Q(\mathbf{x}|y''')} \quad (2)$$

where  $\mathbf{x}$  is the network input vector,  $Q(\mathbf{x}|y)$  is the raw

firing rate of neuron  $y$  in response to input  $\mathbf{x}$  (which is

proportional to the likelihood that input  $\mathbf{x}$  is produced by neuron  $y$  when it is run “in reverse” as a generative model, rather than as a recognition model),  $\mathcal{N}(y)$  is the *local* neighbourhood of neuron  $y$  (which is assumed to contain at least neuron  $y$ ) and  $\mathcal{N}^{-1}(y)$  is the inverse neighbourhood of neuron  $y$  defined as  $\mathcal{N}^{-1}(y) \equiv \{y' | y \in \mathcal{N}(y')\}$ . The term  $\frac{1}{\sum_{y''' \in \mathcal{N}(y'')} Q(\mathbf{x}|y''')}$  is the lateral inhibition factor that derives from the neighbourhood of neuron  $y''$ , which gives rise to a contribution to the lateral inhibition factor for all neurons  $y'$  in the neighbourhood of  $y''$  in the average  $\frac{1}{M} \sum_{y'' \in \mathcal{N}^{-1}(y')} (\dots)$ . Thus the overall lateral inhibition factor acting on neuron  $y'$  is derived *locally* from those neurons  $y'''$  that lie in the set  $\mathcal{N}(\mathcal{N}^{-1}(y'))$ , which is the union of the neighbourhoods  $\mathcal{N}(y)$ , where  $y \in \mathcal{N}^{-1}(y')$ .

The information that is available at the output of the network (as encoded in  $\Pr(y|\mathbf{x})$ ) has been selectively damaged, so that neuron  $y$  fires (rather than the originally intended neuron  $y'$ ) with probability  $\Pr(y|y')$ ; this is known as “probability leakage”. In this paper it will

be assumed that  $\Pr(y|y')$  is chosen to allow only *local* probability leakage (i.e. in a network where the  $y$  index is one-dimensional,  $\Pr(y|y')$  would be a rapidly decaying function of  $|y - y'|$ ), which leads to topographic ordering of neuron properties (Kohonen [2]). Finally, the network attempts to reconstruct its input vector by making use of the set of reference vectors  $\mathbf{x}'(y)$ ,  $y = 1, 2, \dots, M$ .

## B. Interpretation of the Objective Function

In order to interpret the objective function  $D$  in Equation 1, a first order perturbation expansion of the expression for  $\Pr(y|\mathbf{x})$  in Equation 2 will be derived. For simplicity, assume that all of the  $\mathcal{N}(y)$  and  $\mathcal{N}^{-1}(y)$  are the same size  $N$  (e.g. a translation invariant neighbourhood structure with periodic boundary conditions). Expand  $Q(\mathbf{x}|y)$  about  $q_0$ , define  $\Delta(\mathbf{x}|y) = Q(\mathbf{x}|y) - q_0$ , drop terms of order  $\Delta(\mathbf{x}|y)^2$  and higher (see Luttrell [7] for the details), to obtain

$$\Pr(y|\mathbf{x}) \approx \frac{1}{M} + \frac{1}{Mq_0} \sum_{y'=1}^M \Pr(y|y') \left( \Delta(\mathbf{x}|y') - \frac{1}{N^2} \sum_{\substack{y'' \in \mathcal{N}^{-1}(y') \\ y''' \in \mathcal{N}(y'')}} \Delta(\mathbf{x}|y'') \right) \quad (3)$$

which satisfies  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$ , so that  $\Pr(y|\mathbf{x})$  has the correct global normalisation.

The quadratic and higher order terms have been dropped from Equation 3, so it gives a linear (in  $\Delta(\mathbf{x}|y)$ ) approximation to  $\Pr(y|\mathbf{x})$ . This linearity means that, to this level of approximation, the properties of  $\Pr(y|\mathbf{x})$  can be completely analysed by investigating the effect of an isolated signal  $\Delta(\mathbf{x}|y) = a \delta_{y,y_0}$ . (i.e. an isolated peak in  $Q(\mathbf{x}|y)$ , which is located at  $y = y_0$ ). This is essentially a Green’s function analysis, in which the impulse response of a linear system is first derived, and then used subsequently to build up the overall response to an arbitrary input. In this case  $\Pr(y|\mathbf{x})$  is a sum of three pieces: (1) a constant background term  $\frac{1}{M}$ ; (2) an isolated leakage function peak  $\frac{a}{Mq_0} \Pr(y|y_0)$  centred at  $y = y_0$ ; (3) minus the average (for  $y \in \mathcal{N}(\mathcal{N}^{-1}(y_0))$ ) over a set of isolated leakage function peaks  $\frac{1}{Mq_0} \frac{1}{N^2} \sum_{y' \in \mathcal{N}^{-1}(y_0)} \sum_{y'' \in \mathcal{N}(y')} \Pr(y|y'')$ . Term (1) is the constant *free* response of the network, and terms (2) and (3) combine to give the local excitatory plus longer-range inhibitory *forced* response of the network. Because it is assumed that  $\Pr(y|y')$  describes local probability leakage, and that  $\mathcal{N}(y)$  is a local neighbourhood set, then the excitatory term (2) and the inhibitory term (3) are non-zero only over a local region. Since these two terms cancel out in the exact *global* normalisation property

$\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$ , and since both terms are non-zero only near  $y = y_0$ , this implies that there is an exact *local* normalisation property which holds over a local region that is sufficiently large to include all of the non-zero contributions from these two terms. The net effect is that the (positive) area of the excitatory peak cancels out the (negative) area of the surrounding inhibitory trough.

If the input is not an isolated signal  $\Delta(\mathbf{x}|y) \neq a \delta_{y,y_0}$ , then this local normalisation property is no longer exact, because of the contributions to  $\Pr(y|\mathbf{x})$  that arise from outside any local region that one might consider as a candidate for having a local normalisation property. However, this is an edge effect, which reduces as the size of the local region is increased; this ensures that there is an approximate local normalisation property. In summary, the approximate local normalisation property derives from the exact global normalisation property  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$  together with the assumed local nature of  $\Pr(y|y')$  and  $\mathcal{N}(y)$ .

Competition between local excitatory connections and longer range inhibitory connections is common to many models of the visual cortex (Swindale [9]). However, in this chapter the network objective function  $D$  introduces this effect indirectly, by first postulating a model in which the neurons fire independently (i.e.  $Q(\mathbf{x}|y)$ ), and then deriving an expression for the probability that a neuron

fires first (i.e.  $\Pr(y|\mathbf{x})$ ), where the neurons can no longer be treated in isolation of each other (although they still fire independently). Thus the excitatory and inhibitory connections are a consequence of building a probabilistic description of the neural firing events, rather than having been directly modelled at the level of the neural firing events themselves. In summary, knowledge about the firing behaviour of a network (i.e.  $\Pr(y|\mathbf{x})$ ) can exhibit lateral inhibition (i.e. if  $\Pr(y_1|\mathbf{x})$  is large then  $\sum_{y \neq y_1} \Pr(y|\mathbf{x})$  is small), even if the neurons fire independently.

### III. DOMINANCE STRIPES AND ORIENTATION MAPS

The purpose of this section is to discuss the nature of dominance stripes and orientation maps, and to present a simple picture that makes it clear what types of behaviour should be expected from a neural network that minimises the objective function in Equation 1.

#### A. Very-Low-Resolution Input Images

The simplest situation is when there are two retinae, each of which senses independently a featureless scene, i.e. all the pixels in a retina sense the same brightness value, but the two brightnesses that the left and right retinae sense are independent of each other. This situation approximates what happens when the images projected onto the two retinae are derived from different areas of a single image, whose resolution is low enough that each retina approximately senses a featureless scene. This limits the input data to lying in a two-dimensional space  $R^2$ . If these two featureless input images (i.e. left and right retinae) are then normalised so that the sum of left and right retina brightness is constrained to be constant, then the input data is projected down onto a one-dimensional space  $R^1$ , which effectively becomes the ocularity dimension (i.e. the difference between the right and left eye responses).

The optimal network (i.e. the one that minimises  $D$ ) would then be one in which each of the  $M$  neurons had an infinitely wide receptive field to “see” the whole of the featureless input image. The optimal weight vectors and biases must then be chosen to give the best encoding (i.e. it minimises the objective function  $D$ ) of the  $R^1$  space that is visible within these infinitely wide receptive fields. However, because of the limited receptive field size and output layer neighbourhood size, the neurons can at best cooperate a few at a time (this also depends on the size of the leakage neighbourhood). If the network properties are translation invariant, then minimising  $D$  leads in general to an optimal solution which fluctuates periodically with position across the network (Luttrell [5]), where each period typically contains a complete repertoire of the computing machinery that is needed to soft

encode  $R^1$ ; this effect is called “completeness”, and it is a characteristic emergent property of this type of neural network. This is the origin of dominance stripes.

A dubious step in this argument is the use of a normalisation procedure on the input. However, if the input to this network is assumed to be the posterior probability computed by the output layer of another network of the same type, then an approximate local normalisation property would automatically hold; this follows from the form of the posterior probability in Equation 2, which is discussed in Section II. This assumption would be correct for all layers of a multilayer network (except the input layer), provided that each layer integrates its input over time, so that the firing events that it responds to have a chance to build up into an extended pattern of activity, which is proportional to the posterior probability. This property may be used to normalise the input from the two retinae as follows. First, a pair (left and right) of retinal images is mapped to a single image in which the input pixels are interleaved (e.g. in a chessboard fashion) in order to bring corresponding left and right retina pixels into the same local region. Second, these pixel values are identified as the raw firing rates of a set of neurons. Finally, the posterior probability for which neuron fires first is derived, which automatically has an approximate local normalisation property. This posterior probability will be used as the input to our soft encoder network. The preprocessed retinal images are anti-correlated, even though the raw retinal images are statistically independent. This anti-correlation arises because the approximate local normalisation property implies that if the posterior probability for a left-eye neuron to fire first is large, then the posterior probability for nearby right-eye neurons to fire must be small, and vice versa. These results are summarised in Figure 1.

#### B. Low-Resolution Input Images

A natural generalisation of the above is to the case of not-quite-featureless input images. This could be brought about by gradually increasing the resolution of the input images until it is sufficient to reveal spatial detail on a size scale equal to the receptive field size. Instead of seeing a featureless input, each neuron would then see a brightness gradient within its receptive field. This could be interpreted by considering the low-order terms of a Taylor expansion of the input image about a point at the centre of the neuron’s receptive field: the zeroth term is local average brightness (which lives on a one-dimensional line  $R^1$ ), and the two first-order terms are the local brightness gradient (which lives in a two-dimensional space  $R^2$ ). When normalisation is applied this reduces the space in which the two images live to  $R^1 \times R^2 \times R^2$  ( $R^1$  from the zeroth-order Taylor term with normalisation taken into account,  $R^2$  from the first-order Taylor terms, counted twice to deal with both retinae). Note that the normalisation removes only one degree of

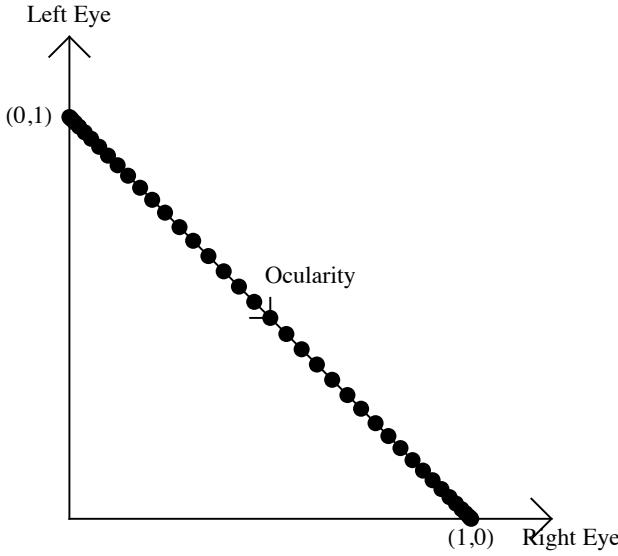


Figure 1: Typical neural reference vectors for very-low-resolution input images. The ocularity dimension runs from  $(0, 1)$  to  $(1, 0)$ . As a function of position across the network, the neuron properties fluctuate periodically back and forth along the ocularity dimension.

freedom; it does not apply separately to each term in the Taylor expansion.

The  $R^1$  from the zeroth-order Taylor term gives rise to ocular dominance stripes (as discussed above) which thus causes the left and right retinae to map to different stripe-shaped regions of the output layer. The remaining  $R^2 \times R^2$  then naturally splits into two contributions (left retina and right retina), each of which maps to the appropriate stripe. If the stripes did not separate the left and right retinae in the network output layer, then the  $R^2 \times R^2$  could not be split apart in this simple manner. Finally, since each ocular dominance stripe occupies a two-dimensional region of the output layer of the network, a direct mapping of the corresponding  $R^2$  (which carries local brightness gradient information) to output space can be made. As in the case of dominance stripes alone, the limited receptive field size and output neuron neighbourhood size causes the neurons to cooperate only a few at a time, so that each local patch of neurons contains a complete mapping from  $R^2$  to the two-dimensional output layer (see Luttrell [5] for a discussion of this point). These results are summarised in Figure 2.

If the amount of probability leakage is reduced then the oscillation back and forth along the dominance axis tends to be more like a square wave than a sine wave, in which case Figure 2 becomes as shown in Figure 3. This change occurs because the effect of probability leakage is to encourage topographic ordering of the neuron properties, so the less leakage there is, the less the neuron properties feel obliged to vary smoothly as a function of

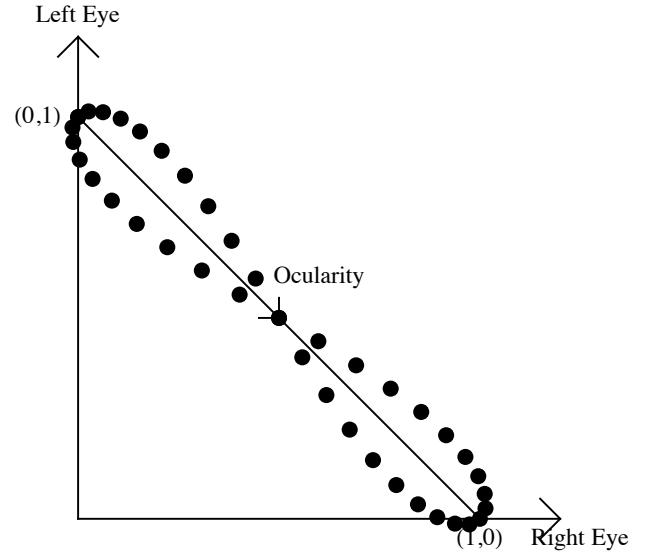


Figure 2: Typical neural reference vectors for low-resolution input images. The pure oscillation back and forth along the ocularity dimension that occurred in Figure 1 develops to reveal some additional degrees of freedom, only one of which is represented here (it is perpendicular to the ocularity axis).

position across the network.

The problem of choosing appropriate receptive field and output layer neighbourhood sizes is non-trivial. In this paper these parameters have been determined by hand. More generally, these parameters must be determined adaptively from a training set, but this much more difficult problem will not be addressed here.

The above arguments can be generalised to the case of input images with fine spatial structure (i.e. lots of high-order terms in the Taylor expansion of the image brightness are required). However, more and more neurons (per receptive field) are required in order to build a faithful mapping from input space to a two-dimensional representation in output space. For a given number of neurons (per receptive field) a saturation point will quickly be reached, where the least important detail (from the point of view of the objective function) is discarded, keeping only those properties of the input images that best preserve the ability of the neural network to reconstruct its own input with minimum Euclidean error (on average).

### C. Theoretical Prediction of Dominance Stripe Wavelength

In Luttrell [7] an estimate of the wavenumber  $k_0$  of dominance stripes (in the limit of zero resolution, as discussed above) was obtained using the first-order perturbation theory of Section II, and using neighbourhoods  $\mathcal{N}(y)$  having a soft Gaussian profile with half-width  $a$  (which approximately corresponds to the  $\frac{N}{2}$  that appears

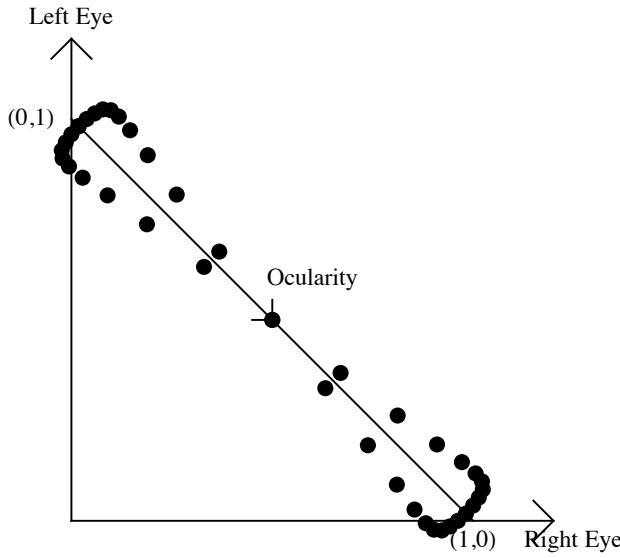


Figure 3: Typical neural reference vectors for low-resolution input images, where reduced leakage causes the ocularity to switch abruptly back and forth. The neural reference vectors are bunched near to the points  $(0, 1)$  and  $(1, 0)$ , and explore the additional degree(s) of freedom at each end of the ocularity axis. In the extreme case, where the ocularity switches back and forth as a square wave, the neurons separate into two clusters, one of which responds only to the left retina's image and the other to the right retina's image. Furthermore, within each of these clusters, the neurons explore the additional degree(s) of freedom that occur within the corresponding retina's image. Note only one such degree of freedom is represented here; it is perpendicular to the ocularity axis.

in Section II), and leakage probability functions  $\Pr(y'|y)$  having a Gaussian profile with half-width  $l$ , to yield

$$k_0 = \frac{1}{a} \sqrt{\log \left( \frac{2a^2}{l^2} + 1 \right)} \quad (4)$$

This result is not dependent on the training data, because the training data used had a very low resolution (as described above), and the receptive field and output layer neighbourhood sizes were fixed by hand. The wavenumber  $k_0$  is proportional to  $\frac{1}{a}$  times a logarithmic correction factor, so the inhibitory connection range (which is  $\mathcal{O}(a)$ ) determines the wavelength of the dominance stripes (which is  $\frac{2\pi}{k_0}$ ). As the leakage range  $l$  increases, the wavenumber  $k_0$  decreases slowly as expected, because the leakage forces neighbouring neuron properties to be more correlated than they would have been without leakage. However, the prediction in Equation 4 breaks down as  $l \rightarrow 0$ , because in order to derive  $k_0$  the Gaussian profiles of  $\mathcal{N}(y)$  and  $\Pr(y'|y)$  were assumed to be continuous, rather than sampled (Luttrell [7]).

## IV. SIMULATIONS

Two types of training data will be used: synthetic and natural. Synthetic data is used in order to demonstrate simple properties of VICON, without introducing extraneous detail to complicate the interpretation of the results. Natural data is used to remove any doubt that the neural network is capable of producing interesting and useful results when it encounters data that is more representative of what it might encounter in the real world.

In this section dominance stripes are produced from a one-dimensional retina, and these results are generalised to a two-dimensional retina. In both cases both synthetic and natural image results are shown.

### A. Dominance Stripes: the One-Dimensional Case

The purpose of the simulations that are presented in this section is to demonstrate the emergence of ocular dominance stripes in the simplest possible realistic case. The results will correspond to the situation that was outlined in Figure 1.

#### 1. Featureless Training Data

In this simulation the network is presented with very-low-resolution input images. In fact, the resolution is so low that each image is entirely featureless, so that all the neurons in a retina have the same input brightness, but the two retinæ have independent input brightnesses. These input images are preprocessed by interleaving their pixels in a chessboard fashion, then normalised by processing them so that they look like the posterior probability computed by the output layer of another such network; the neighbourhood size used for this normalisation process was chosen to be the same as the network's own output layer neighbourhood size. The emergence of dominance stripes depends critically on this hard-wired preprocessing, which brings together signals from corresponding parts of the two retinæ. More generally, this pattern of wiring should be allowed to emerge automatically when the network is optimised, but this problem will not be studied here.

In the first simulation the parameters used were: network size = 30, receptive field size = 9, output layer neighbourhood size = 5 (centred on the source neuron), leakage neighbourhood size = 5 (centred on the source neuron), number of training updates = 2000, update step size = 0.01. For each neuron the leakage probability had a Gaussian profile centred on the neuron, and the standard deviation was chosen as 1, to make the profile fall from 1 on the source neuron to  $\exp(-\frac{1}{2})$  on each of its two closest neighbours. The precise values of these parameters is not critical, although the period of the dominance stripes is largely determined by the output layer neighbourhood size (see Equation 4).

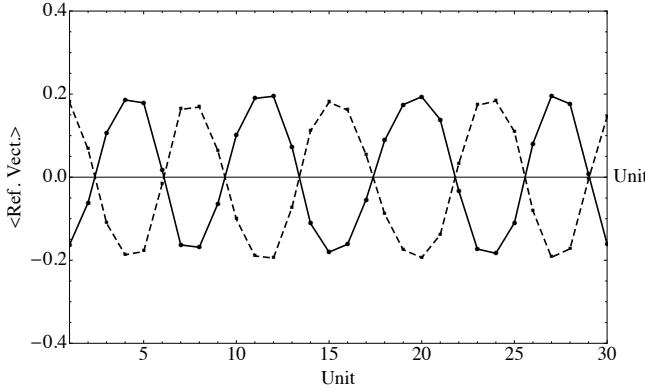


Figure 4: One-dimensional dominance stripes after training on synthetic data.

At the edges of the network the output layer neighbourhoods are symmetrically truncated by the edge of the network; thus the neighbourhood of an edge neuron contains only itself. The leakage neighbourhood is asymmetrically truncated on the side which abuts the edge; thus the leakage neighbourhood of an edge neuron contains itself and the two neighbouring neurons away from the edge (assuming a leakage neighbourhood size of 5). The details of this neighbourhood prescription do not have a marked effect on the appearance of dominance stripes in one-dimensional simulations, because their period is determined mainly by the width of the output layer neighbourhoods. However, in the two-dimensional simulations to be described below, the anisotropic shape of the output layer neighbourhoods strongly influences the orientation of the two-dimensional dominance stripes. For consistency, the same neighbourhood prescription is used for both one-dimensional and two-dimensional simulations.

The update scheme used was a crude gradient-following algorithm parameterised by three numbers which controlled the rate at which the weight vectors, biases and reference vectors were updated. These three numbers were continuously adjusted to ensure that the maximum rate of change (as measured over all the neurons in the network) of the length of each weight vector (divided by the receptive field size), and also the maximum rate of change of the absolute value of each bias, was always equal to the requested update step size; this prescription will adjust the parameter values until they move around in the neighbourhood of their optimum values. The reference vectors were controlled in a similar way to the weight vectors, except that they used three times the update step size, which made them more agile than the weights and biases they were trying to follow. A more sophisticated algorithm would allow the update step size to be reduced as the optimum parameter values were approached, but this has not been implemented in this study.

The ocular dominance stripes that emerge from this simulation are shown in Figure 4. The ocularity for a

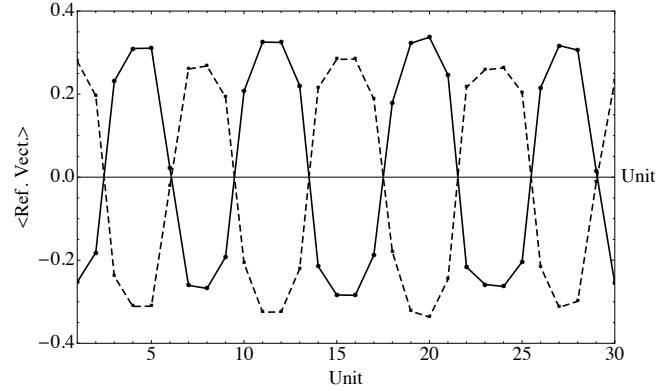


Figure 5: One-dimensional square-wave dominance stripes after further training with reduced probability leakage on synthetic data.

given neuron was estimated by computing the average of the absolute deviations (as measured with respect to the overall mean reference vector component value, which is zero for the zero-mean training data used here) of its reference vector components, both for the left retina and the right retina. This allows two plots to be drawn: average value of absolute deviations from the mean in left retina's receptive field as a function of position across the network, and similarly the right retina's receptive field.

As can be seen in Figure 4, these two curves are approximately periodic, and are in antiphase with each other; this corresponds to the situation shown in Figure 1. The amplitude of the ocularity curves is less than the 0.5 that would be required for the end points of the ocularity dimension to be reached, because one of the effects of leakage is to introduce a type of elastic tension between the reference vectors that causes them to contract towards zero ocularity.

Assuming that the effective neighbourhood Gaussian width is  $a = 2$  (i.e. the half-width of the output layer neighbourhood size) and the effective leakage Gaussian width is  $l = 1$  (i.e. the half-width of the leakage neighbourhood size), the dominance stripe wavenumber predicted by Equation 4 is  $k_0 = \frac{1}{2} \sqrt{\log(\frac{8}{1} + 1)} \approx 0.74$ , so the dominance stripe wavelength is predicted to be approximately  $\frac{2\pi}{0.74} = 8.5$ , which is greater than the estimated value of 7 that is obtained from the results shown in Figure 4. This discrepancy could be due to overestimating the size of either  $a$  or  $l$ , which is likely, since in the simulation both the neighbourhood and the leakage were implemented using only finite-sized windows.

If the above simulation is continued for a further 2000 updates with a reduced leakage, by reducing the standard deviation of the Gaussian leakage profile from 1 to 0.5, then the ocular dominance curves become more like square waves than sine waves, as shown in Figure 5; this is similar to the type of situation that was shown in Figure 3, except that the input images are featureless in this case.

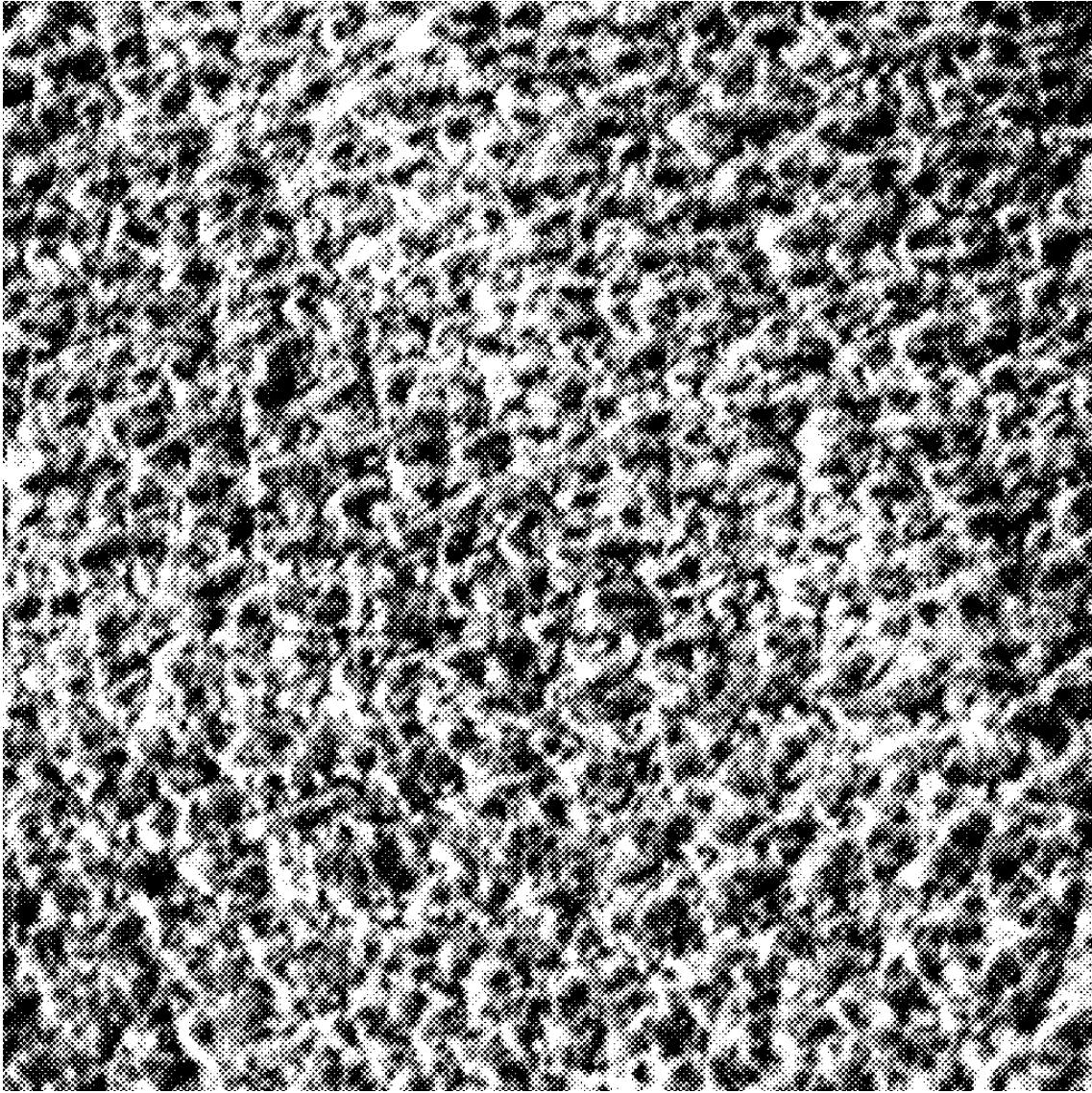


Figure 6: Brodaz texture image used as a natural training image. The correlation length of the texture structure is in the range 5-10 pixels.

## 2. Natural Training Data

Figure 6 shows the Brodaz texture image (Brodaz [1]) that was used to generate a more realistic training set than was used in the synthetic simulations described above.

The correlation length of this texture is comparable to the receptive field size (9 pixels) and the output layer neighbourhood size (5 pixels), so a simulation using one-dimensional training vectors extracted from this two-dimensional Brodaz image will effectively see very-low-resolution training data (i.e. each training vector occupies no more than one correlation length of the Brodaz image), and should thus respond approximately as described in Figure 1. This Brodaz image was chosen because it has a simple structure, which is similar to spa-

tially correlated noise. More complicated Brodaz images would make the results obtained in this paper more difficult to interpret.

The results corresponding to Figure 4 and Figure 5 for this image are shown in Figure 7 and Figure 8, respectively.

The featureless image and Brodaz image results are similar to each other, except that the depth of the ocularity fluctuations is somewhat less in the Brodaz image case, because in the Brodaz case the training data is not actually featureless within each receptive field. Intuitively, it can be seen that for a given network, any increase in the image structure will increase the effective dimensionality of the input space, which will cause the neuron parameters to spread out to fill that space; the reduced depth of the ocularity fluctuations is a side-effect

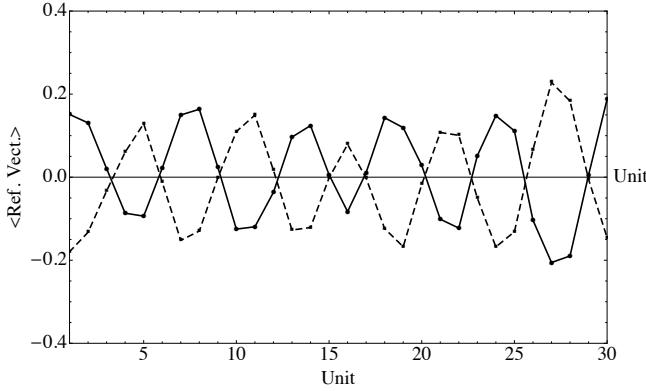


Figure 7: One-dimensional dominance stripes after training on natural data.

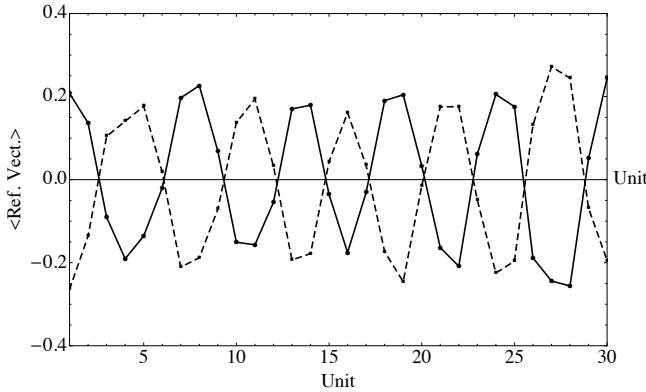


Figure 8: One-dimensional square wave dominance stripes after further training with reduced probability leakage on natural data.

of this phenomenon.

## B. Dominance Stripes: the Two-Dimensional Case

The results of the previous simulations are extended to the case of two-dimensional networks. The training schedule(s) used in the simulations have not been optimised. Usually the update rate is chosen conservatively (i.e. smaller than it needs to be) to avoid possible numerical instabilities, and the number of training updates is chosen to be larger than it needs to be to ensure that convergence has occurred. It is highly likely that much more efficient training schedules could be found.

### 1. Featureless Training Data

The results that were presented in Figure 4 may readily be extended to the case of a two-dimensional network. The parameters used were: network size =  $100 \times 100$ , receptive field size =  $3 \times 3$  (which is artificially small to allow the simulation to run faster), output layer

neighbourhood size =  $5 \times 5$  (centred on the source neuron), leakage neighbourhood size =  $3 \times 3$  (centred on the source neuron), number of training updates = 24,000 (dominance stripes develop quickly, so far fewer than 24,000 training updates could be used), update step size = 0.001. For each neuron the leakage probability had a Gaussian profile centred on the neuron, and the standard deviations were chosen as  $1 \times 1$ , to make the profile fall from 1 on the source neuron to  $\exp(-\frac{1}{2})$  on each of its four closest neighbours. Apart from the different parameter values, the simulation was conducted in precisely the same way as in the one-dimensional case, and the results for ocular dominance are shown in Figure 9, where ocularity has been quantised as a binary-valued quantity solely for display purposes.

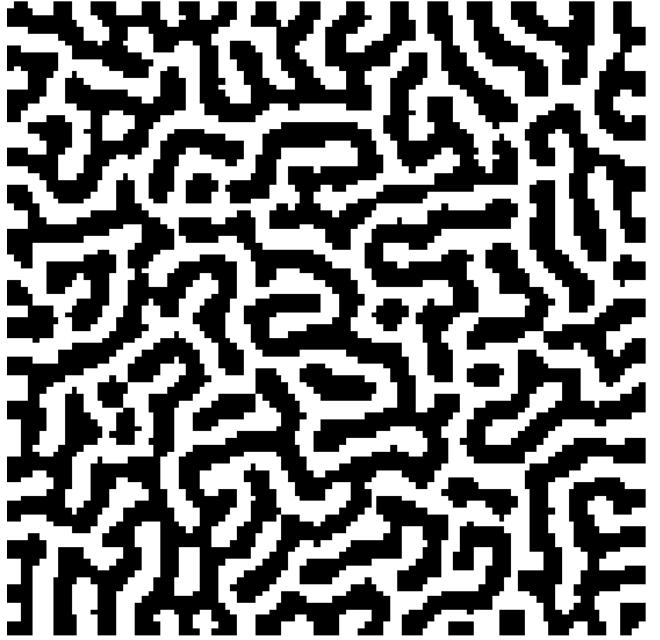


Figure 9: Two-dimensional dominance stripes after training on synthetic data. These results show the characteristic striped structure that is familiar from experiments on the mammalian visual cortex, and the dominance stripes run perpendicular to the boundary, as expected.

As in the one-dimensional case studied above, the output layer neighbourhoods are truncated symmetrically, and the leakage neighbourhoods are truncated asymmetrically, at the edges of the network. Whilst this had little effect on the results that were obtained in the one-dimensional case, in the two-dimensional case the shape of the output layer neighbourhoods determines the direction in which the dominance stripes run. The symmetric truncation of the output layer neighbourhoods is used to force these neighbourhoods to become highly anisotropic near the edges of the network, where they become one-dimensional neighbourhoods running parallel to the edge of the network. This type of neighbourhood leads to dominance stripes that run perpendicular to the edge of the network, as observed in Figure 9 (and in the mammalian

visual cortex). If a weaker anisotropy were used (such as would be the case if the output layer neighbourhood truncation scheme were asymmetric), then ideally the dominance stripes should still run perpendicular to the edge of the network, but simulations show that the network parameters are then more likely to become trapped in a suboptimal configuration where the dominance stripes do not run everywhere perpendicular to the edge of the network.

By counting the stripes as they pass through the boundary in Figure 9, the dominance stripe wavelength is estimated as 7. The theoretically predicted wavelength prediction is 8.5 (as in the one-dimensional case, where the same parameter values were used). Possible sources of this discrepancy are discussed below.

## 2. Natural Training Data

The simulation, whose results were shown in Figure 9, may be repeated using the Brodatz image training set shown in Figure 6, to yield the results shown in Figure 10.

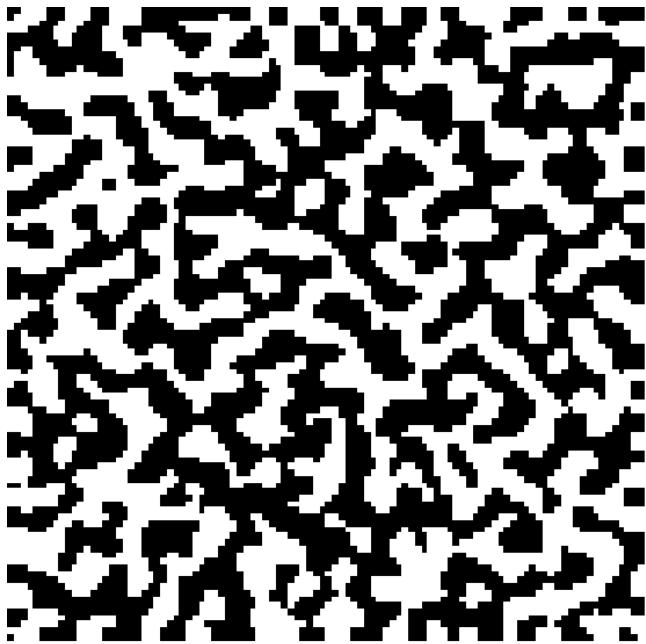


Figure 10: Two-dimensional dominance stripes after training on natural data. These results are not quite as stripe-like as the results in Figure 9, because in the Brodatz case the training data is not actually featureless within each receptive field.

The correlation length of the Brodatz image (approximately 10 pixels) implies an upper limit on the receptive field and output layer neighbourhood sizes, because if these are too large then each neuron will erroneously attempt to process signals that derive from independent correlation areas of the input image. These sizes have been chosen by hand, so that each neuron responds

mainly to signals that derive from a single correlation area. The period of the dominance stripes is thus approximately the correlation length of the Brodatz image.

## C. Orientation Maps

The purpose of the simulations presented in this section is to demonstrate the emergence of orientation maps in the simplest possible realistic case. In the case of two retinae, the results would correspond to the situation outlined in Figure 2 (or, at least, a higher-dimensional version of Figure 2).

### 1. Orientation Map (One Retina)

In this simulation the parameters used were: network size =  $30 \times 30$ , receptive field size =  $17 \times 17$ , output layer neighbourhood size =  $9 \times 9$  (centred on the source neuron), leakage neighbourhood size =  $3 \times 3$  (centred on the source neuron), number of training updates = 24,000, update step size = 0.01. For each neuron the leakage probability had a Gaussian profile centred on the neuron, and the standard deviations were chosen as  $1 \times 1$ , to make the profile fall from 1 on the source neuron to  $\exp(-\frac{1}{2})$  on each of its four closest neighbours. The output layer neighbourhoods were truncated symmetrically, and the leakage neighbourhoods were truncated asymmetrically, at the edges of the network.

Note that both the receptive field size and the output layer neighbourhood size are substantially larger than in the two-dimensional dominance stripe simulations, because many more neurons are required in order to allow orientation maps to develop than to allow dominance stripes to develop; in fact it would be preferable to use even larger sizes than were used here. To limit the computer run time this meant that the overall size of the neural network had to be reduced from  $100 \times 100$  to  $30 \times 30$ . The training set was the Brodatz texture image in Figure 6. The results are shown in Figure 11.

### 2. Using the Orientation Map

In Figure 12 the orientation map network shown in Figure 11 is used to encode and decode a typical input image. On the left of Figure 12 the input image (i.e.  $\mathbf{x}$ ) is shown, in the centre of Figure 12 the corresponding output (i.e. its posterior probability  $\Pr(y|\mathbf{x})$ ) produced by the orientation map in Figure 11 is shown, and on the right of Figure 12 the corresponding reconstruction (i.e.  $\sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y)$ ) is shown.

The form of the output in the centre of Figure 12 is familiar as a type of “sparse coding” of the input, where

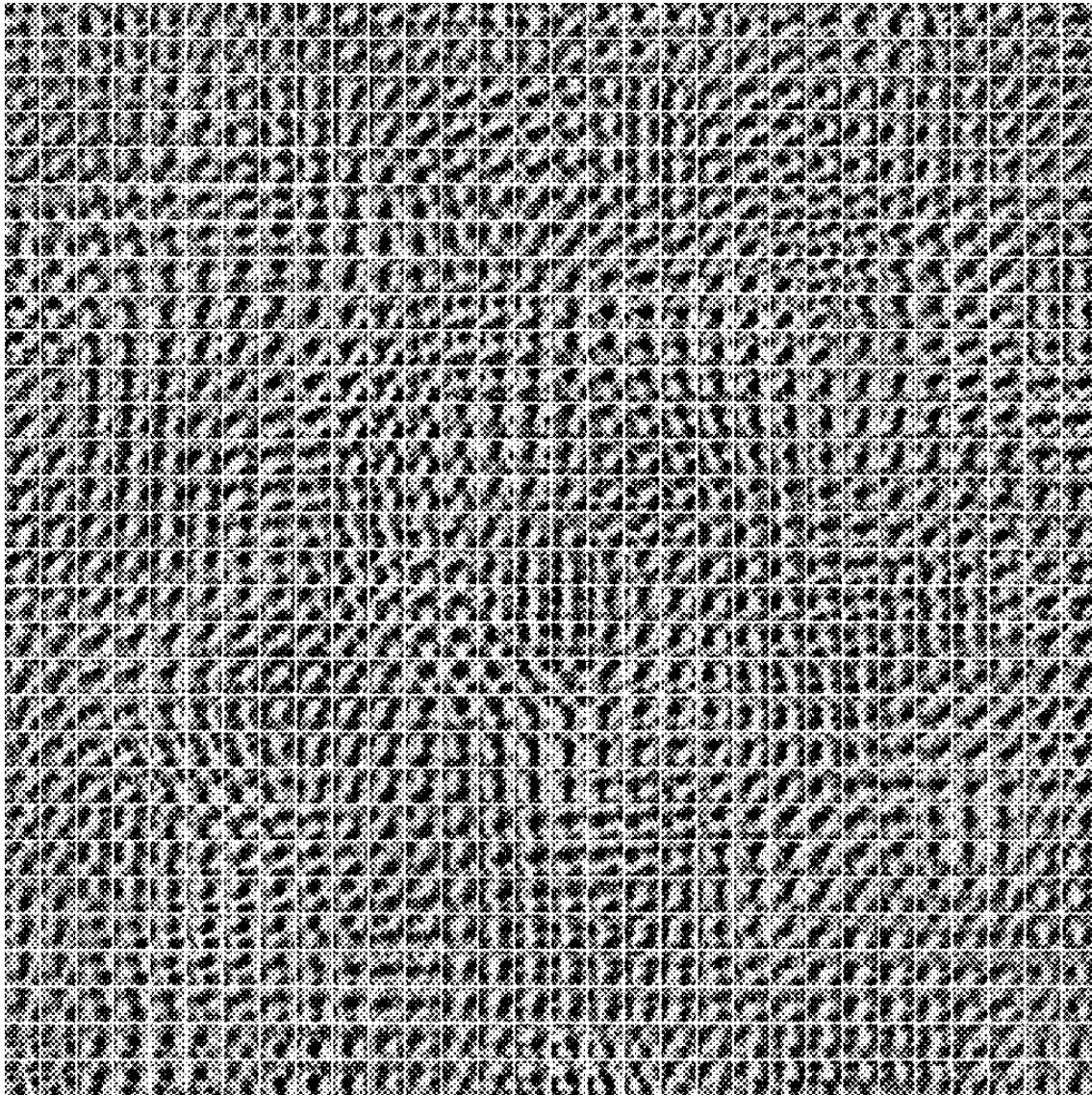


Figure 11: Orientation map after training on natural data. The receptive fields have been gathered together in a montage. There is a clear swirl-like pattern that is characteristic of orientation maps. Each local clockwise or anticlockwise swirl typically circulates around an unoriented region. The contrast in this figure has been manually adjusted in order to make the fine details easier to see.

only a small fraction of the neurons participate in encoding a given input (this type of transformation of the input is central to the work that was reported in Webber [10]). This type of encoding is very convenient because it has effectively transformed the input into a small number of constituents each of which corresponds to a small patch of neural activity (an “activity bubble”), rather than transforming the input into a representation where the output activity is spread over all of the neurons, which would not be easily interpretable as arising from a small number of constituents.

The reconstruction has a lower resolution than the input because there are insufficient neurons to faithfully record all the information that is required to reconstruct the input exactly (e.g. probability leakage causes neigh-

bouring neurons to have a correlated response, thus reducing the effective number of neurons that are available). The featureless region around the edge of the reconstruction is an edge effect, which occurs because fewer neurons (per unit area) contribute to the reconstruction near the edge of the input array. The quality of the reconstruction may be improved by increasing the size of the network (to reduce the proportion of the network that is distorted by the edge effect), or using more neurons per unit area of the image (to increase the resolution of the reconstruction).

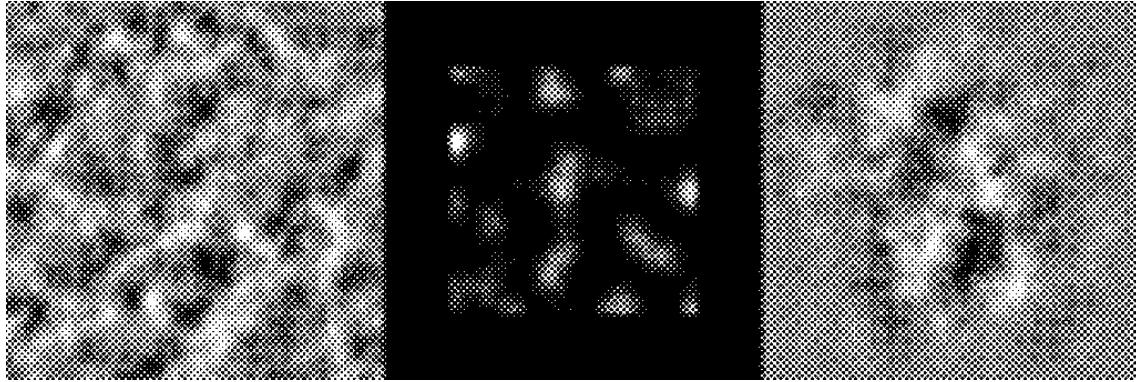


Figure 12: Typical input, output and reconstruction produced by the orientation map. The output consists of a number of isolated ‘‘activity bubbles’’ of posterior probability, and the reconstruction is a low-resolution version of the original input.

## V. CONCLUSIONS

This chapter has shown that the theory of self-organising networks introduced in Luttrell [4] yields a class of self-organising neural networks (Luttrell [6]) which has many of the properties that are observed in the mammalian visual cortex (Swindale [9]). This type of network will thus be called a VIsual COortex Network (VICON). These neural networks differ from previous models of the visual cortex, insofar as they model the neuron behaviour in terms of their individual firing events, and also operate in the real space of input images. When the neural network structure (e.g. receptive field size) parameters are chosen to match the correlation properties of the training data, dominance stripes and orientation maps emerge when the network is trained on a natural image (e.g. a Brodatz texture image).

These results show that if this type of network is trained on data from multiple sources, then its internal parameters self-organise into the expected patterns, such as dominance stripes. Only one or two sources have been used in this study, but the same network objective function could be used if an arbitrary number of sources is used, and it is anticipated that it would lead to analogous

results.

Although VICON has many emergent properties (such as dominance stripes and orientation maps) that are similar to observed properties of the mammalian visual cortex, VICON is nevertheless derived from the soft encoder theory in Luttrell [4], which is not a physiologically motivated theory. It is not yet clear whether VICON is a useful model of the visual cortex.

The main limitation of the results that have been presented in this chapter is that the network structure is hand-crafted (e.g. the receptive field and output layer neighbourhood sizes). This restriction has been deliberately imposed for simplicity; it is not a fundamental limitation of the approach. For instance, it is possible to do more sophisticated simulations in which the network objective function is also optimised with respect to both the receptive field and output layer neighbourhood sizes (Luttrell [8]).

## VI. ACKNOWLEDGEMENTS

I thank Chris Webber for many useful discussions that we had during the course of this research.

---

The derivatives of the objective function with respect to its parameters are presented and interpreted.

### 1. Derivatives of the Objective Function

In order to minimise the network objective function in Equation 1 (using the exact expression for  $\Pr(y|\mathbf{x})$  in Equation 2, rather than its approximation in Equation 3) its derivatives must be calculated. First of all, define some

convenient notation (Luttrell [6]):

$$\begin{aligned}
L_{y,y'} &\equiv \Pr(y'|y) & P_{y,y'} &\equiv \Pr(y'|\mathbf{x}; y) \equiv \frac{Q(\mathbf{x}|y') \delta_{y' \in \mathcal{N}(y)}}{\sum_{y'' \in \mathcal{N}(y)} Q(\mathbf{x}|y'')} \\
p_y &\equiv \sum_{y' \in \mathcal{N}^{-1}(y)} P_{y',y} & (L^T p)_y &\equiv \sum_{y'=1}^M L_{y',y} p_{y'} \\
e_y &\equiv \|\mathbf{x} - \mathbf{x}'(y)\|^2 & (L e)_y &\equiv \sum_{y'=1}^M L_{y,y'} e_{y'} \\
(P L e)_y &\equiv \sum_{y' \in \mathcal{N}(y)} P_{y,y'} (L e)_{y'} & (P^T P L e)_y &\equiv \sum_{y' \in \mathcal{N}^{-1}(y)} P_{y',y} (P L e)_{y'} \tag{5}
\end{aligned}$$

whence  $\Pr(y|\mathbf{x})$  in Equation 2 may be written as  $\Pr(y|\mathbf{x}) = \frac{1}{M} (L^T p)_y$ , and the derivatives of  $D$  may be obtained in the form (Luttrell [6])

$$\begin{aligned}
\frac{\partial D}{\mathbf{x}'(y)} &= -\frac{4}{M} \int d\mathbf{x} \Pr(\mathbf{x}) (L^T p)_y (\mathbf{x} - \mathbf{x}'(y)) \\
\frac{\delta D}{\delta \log Q(\mathbf{x}|y)} &= \frac{2}{M} \Pr(\mathbf{x}) \left( p_y (L e)_y - (P^T P L e)_y \right) \tag{6}
\end{aligned}$$

Assume that the raw neuron firing rates may be modelled using a sigmoid function

$$Q(\mathbf{x}|y) = \frac{1}{1 + \exp(-\mathbf{w}(y) \cdot \mathbf{x} - b(y))} \tag{7}$$

then the two derivatives  $\frac{\partial D}{\partial b(y)}$  and  $\frac{\partial D}{\partial \mathbf{w}(y)}$  may be obtained in the form

$$\frac{\partial D}{\partial \begin{pmatrix} b(y) \\ \mathbf{w}(y) \end{pmatrix}} = \frac{2}{M} \int d\mathbf{x} \Pr(\mathbf{x}) \left( p_y (L e)_y - (P^T P L e)_y \right) (1 - Q(\mathbf{x}|y)) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \tag{8}$$

## 2. Interpretation of the Derivatives of the Objective Function

In Equation 6,  $\frac{\partial D}{\partial \mathbf{x}'(y)}$  may be interpreted as follows. Thus  $D$  is reduced by moving the reference vector  $\mathbf{x}'(y)$  directly towards the input vector  $\mathbf{x}$  by an amount that is proportional to the probability  $\Pr(y|\mathbf{x})$  that neuron  $y$  fires first; all the reference vectors  $\mathbf{x}'(y)$  (for  $y = 1, 2, \dots, M$ ) are simultaneously updated in this fashion. This type of update prescription is a soft version of the winner-take-all prescription proposed by Kohonen [2], and it similarly leads to topographic ordering of the reference vectors.

In Equation 6,  $\frac{\delta D}{\delta \log Q(\mathbf{x}|y)}$  may be most easily interpreted by replacing  $\Pr(y|\mathbf{x})$  in Equation 2 by the approximation in Equation 3. Thus

$$\frac{\delta D}{\delta Q(\mathbf{x}|z)} \approx \frac{2}{M q_0} \Pr(\mathbf{x}) \sum_{y=1}^M \left( \Pr(y|z) - \frac{1}{N^2} \sum_{\substack{y'' \in \mathcal{N}^{-1}(z) \\ y''' \in \mathcal{N}(y'')}} \Delta(y|y''') \right) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \tag{9}$$

Thus  $D$  is reduced by reducing  $Q(\mathbf{x}|y)$  by an amount that is proportional to the sum of the following two terms:

1.  $\sum_{y'=1}^M \Pr(y'|y) \|\mathbf{x} - \mathbf{x}'(y')\|^2$ : a local excitatory term, which requires that those neurons that have a larger (or smaller) than average contribution to the reconstruction error to reduce (or increase) their raw firing rate. The exact definition of this average is given below in the interpretation of the inhibitory term.

2.  $-\frac{1}{N^2} \sum_{y=1}^M \left( \sum_{\substack{y'' \in \mathcal{N}^{-1}(y) \\ y''' \in \mathcal{N}(y'')}} \Pr(y'|y''') \right) \|\mathbf{x} - \mathbf{x}'(y')\|^2$ :

a longer-range inhibitory term, which is the average value (as measured over the neighbourhood  $a$  of the reconstruction error).

Overall, this update prescription ensures that the raw

firing rates  $Q(\mathbf{x}|y)$  and the reference vectors  $\mathbf{x}'(y)$  (for  $y = 1, 2, \dots, M$ ) are adjusted in such a way that the objective function  $D$  is reduced.

Although the neurons fire independently, the update prescription involves local excitatory and longer-range inhibitory terms. These excitatory and inhibitory terms are not included in the underlying neural firing model. Rather, they arise as a side-effect of constructing an objective function for network optimisation that depends on the posterior probability  $\Pr(y|\mathbf{x})$  that neuron  $y$  fires

first. The model used for  $\Pr(y|\mathbf{x})$ , taken together with the normalisation property  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$ , then automatically leads to local excitatory and longer-range inhibitory terms, as discussed in Section II. These excitatory and inhibitory terms do not affect the raw firing rate  $Q(\mathbf{x}|y)$  of the neuron  $y$ , but they do affect the posterior probability  $\Pr(y|\mathbf{x})$  that neuron  $y$  fires first, which in turn affects the value of the network objective function  $D$ , and hence the update prescription.

- [1] P Brodatz, *Textures - a photographic album for artists and designers*, Dover, New York, 1966.
- [2] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [3] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [4] S P Luttrell, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
- [5] ———, *Partitioned mixture distribution: an adaptive Bayesian network for low-level image processing*, IEE Proceedings on Vision, Image, and Signal Processing **141** (1994), no. 4, 251–260.
- [6] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
- [7] ———, *Partitioned mixture distributions: a first order perturbation analysis*, Proceedings of IEE international conference on artificial neural networks (Cambridge), IEE, 1997, pp. 280–284.
- [8] ———, *Self-organisation of multiple winner-take-all neural networks*, Connection Science **9** (1997), no. 1, 11–30.
- [9] N V Swindale, *The development of topography in the visual cortex: a review of models*, Network: Computation in Neural Systems **7** (1996), no. 2, 161–247.
- [10] C J S Webber, *Self-organisation of transformation-invariant detectors for constituents of perceptual patterns*, Network: Computation in Neural Systems **5** (1994), no. 4, 471–496.



# Self-Organised Discovery of Structure in Signal Manifolds \*

S P Luttrell

The theory of stochastic vector quantisers (SVQ) has been extended to allow the quantiser to develop invariances, so that only “large” degrees of freedom in the input vector are represented in the code. This has been applied to the problem of encoding data vectors which are a superposition of a “large” jammer and a “small” signal, so that only the jammer is represented in the code. This allows the jammer to be subtracted from the total input vector (i.e. the jammer is nulled), leaving a residual that contains only the underlying signal. The main advantage of this approach to jammer nulling is that little prior knowledge of the jammer is assumed, because these properties are automatically discovered by the SVQ as it is trained on examples of input vectors.

## I. INTRODUCTION

In vector quantisation a code book is used to encode each input vector as a corresponding code index, which is then decoded (again, using the codebook) to produce an approximate reconstruction of the original input vector [2, 3]. The standard approach to vector quantiser (VQ) design [6] may be generalised [8] so that each input vector is encoded as a *vector* of code indices that are stochastically sampled from a probability distribution that depends on the input vector, rather than as a *single* code index that is the deterministic outcome of finding which entry in a code book is closest to the input vector. This will be called a stochastic VQ (SVQ), and it includes the standard VQ as a special case.

One advantage of using the stochastic approach is that it automates the process of splitting high-dimensional input vectors into low-dimensional blocks before encoding them [14], because minimising the mean Euclidean reconstruction error can encourage different stochastically sampled code indices to become associated with different input subspaces [9, 11]. Another advantage is that it is very easy to connect SVQs together, by using the vector of code index probabilities computed by one SVQ as the input vector to another SVQ [10].

In Section II the underlying theory of SVQs is developed, and in Section III some simulations illustrating the application of SVQs to the nulling of jammers are presented.

## II. STOCHASTIC VECTOR QUANTISER THEORY

In Section II A the folded Markov chain (FMC) theory of adaptive encoders is developed [7], in Section II B

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

Published in the Proceedings of the 2nd DERA Workshop on Sensor Array Signal Processing (DERA Technical Report,

the theory is cast into a form that is scales well to high-dimensional input vectors [8], in Section II C the theory extended to allow invariant codes in which only the “large” subspaces of the input are encoded [13], and finally in Section II D this invariant encoder theory is applied to the problem of encoding and subsequently nulling “large” jammers that obscure “small” signals.

### A. Folded Markov Chains

The basic building block of the SVQ used in this paper is the folded Markov chain (FMC) [7]. An input vector  $\mathbf{x}$  is encoded as a code index vector  $\mathbf{y}$ , which is then subsequently decoded as a reconstruction  $\mathbf{x}'$  of the input vector. Both the encoding and decoding operations are allowed to be probabilistic, in the sense that  $\mathbf{y}$  is a sample drawn from  $\text{Pr}(\mathbf{y}|\mathbf{x})$ , and  $\mathbf{x}'$  is a sample drawn from  $\text{Pr}(\mathbf{x}'|\mathbf{y})$ , where  $\text{Pr}(\mathbf{y}|\mathbf{x})$  and  $\text{Pr}(\mathbf{x}'|\mathbf{y})$  are Bayes’ inverses of each other, as given by  $\text{Pr}(\mathbf{x}'|\mathbf{y}) = \frac{\text{Pr}(\mathbf{y}|\mathbf{x}) \text{Pr}(\mathbf{x})}{\int d\mathbf{z} \text{Pr}(\mathbf{y}|\mathbf{z}) \text{Pr}(\mathbf{z})}$ , and  $\text{Pr}(\mathbf{x})$  is the prior probability from which  $\mathbf{x}$  is sampled.

In order to ensure that the FMC encodes the input vector optimally, a measure of the reconstruction error must be minimised. There are many possible ways to define this measure, but one that is consistent with many previous results, and which also leads to many new results, is the mean Euclidean reconstruction error measure  $D$ , which is defined as [7]

DERA/S&E/SPI/TR000349, pp 88-93), DERA Malvern, 17-18 May 2000. © British Crown Copyright 2000 / DERA

$$D \equiv \int d\mathbf{x} \text{Pr}(\mathbf{x}) \sum_{y_1=1}^M \sum_{y_2=1}^M \cdots \sum_{y_n=1}^M \text{Pr}(\mathbf{y}|\mathbf{x}) \int d\mathbf{x}' \text{Pr}(\mathbf{x}'|\mathbf{y}) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (2.1)$$

where  $\mathbf{y} = (y_1, y_2, \dots, y_n)$ ,  $1 \leq y_i \leq M$  is assumed,  $\Pr(\mathbf{x}) \Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x}'|\mathbf{y})$  is the joint probability that the FMC has state  $(\mathbf{x}, \mathbf{y}, \mathbf{x}')$ ,  $\|\mathbf{x} - \mathbf{x}'\|^2$  is the Euclidean reconstruction error, and  $\int d\mathbf{x} \sum_{y_1=1}^M \sum_{y_2=1}^M \dots \sum_{y_n=1}^M \int d\mathbf{x}' (\dots)$  sums over all possible states of the FMC (weighted by the joint probability).

The Bayes' inverse probability  $\Pr(\mathbf{x}'|\mathbf{y})$  may be integrated out of this expression for  $D$  to yield [7]

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1=1}^M \sum_{y_2=1}^M \dots \sum_{y_n=1}^M \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (2.2)$$

where the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  is defined as  $\mathbf{x}'(\mathbf{y}) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ . Because of the quadratic form of the objective function, it turns out that  $\mathbf{x}'(\mathbf{y})$  may be treated as a free parameter whose optimum value (i.e. the solution of  $\frac{\partial D}{\partial \mathbf{x}'(\mathbf{y})} = 0$ ) is  $\int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ , as required.

It was shown in [7] that the standard VQ [6] and topographic mappings [4] automatically emerge as special cases when  $D$  is minimised. In this approach, topographic mappings emerge as the optimal coding scheme when the code is to be transmitted along a noisy communication channel before being decoded [1, 5].

## B. High Dimensional Input Spaces

A problem with the standard VQ is that its code book grows exponentially in size as the dimensionality of the input vector is increased, assuming that the contribution to the reconstruction error from each input dimension is held constant. This means that such VQs are useless for encoding extremely high dimensional input vectors, such as images. The usual solution to this problem is to manually partition the input space into a number of lower dimensional subspaces, and then to encode each of these subspaces separately. However, it would be very useful if this partitioning could be done automatically, in such a way that typically the correlations *within* each subspace were much stronger than the correlations *between* subspaces, so that the subspaces were approximately statistically independent of each other [14].

The key step in solving this problem is to constrain the minimisation of  $D$  in such a way as to encourage the formation of code schemes in which each component of the code vector  $\mathbf{y}$  codes a different subspace of the input vector  $\mathbf{x}$ . There are two related constraints that may be imposed on  $\Pr(\mathbf{y}|\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  which may be summarised as

$$\begin{aligned} \Pr(\mathbf{y}|\mathbf{x}) &= \Pr(y_1|\mathbf{x}) \Pr(y_2|\mathbf{x}) \dots \Pr(y_n|\mathbf{x}) \\ \mathbf{x}'(\mathbf{y}) &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}'(y_i) \end{aligned} \quad (2.3)$$

Thus each component  $y_i$  (for  $i = 1, 2, \dots, n$  and  $1 \leq y_i \leq M$ ) is an *independent* sample drawn from the

codebook using  $\Pr(y_i|\mathbf{x})$  (which is assumed to be the same function for all  $i$ ), and the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  (vector argument  $\mathbf{y}$ ) is assumed to be a *superposition* of  $n$  contributions  $\mathbf{x}'(y_i)$  (scalar argument  $y_i$ ) for  $i = 1, 2, \dots, n$ . Taken together, these constraints encourage the formation of coding schemes in which independent subspaces are separately coded, as required.

The constraints in Equation 2.3 prevent the full space of possible values of  $\Pr(\mathbf{y}|\mathbf{x})$  or  $\mathbf{x}'(\mathbf{y})$  from being explored as  $D$  is minimised, so they lead to only an *upper bound*  $D_1 + D_2$  on the FMC objective function  $D$  (i.e.  $D \leq D_1 + D_2$ ). However,  $D_1 + D_2$  may be used as the *definition* of the objective function of a stochastic vector quantiser (SVQ), where [8]

$$\begin{aligned} D_1 &\equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \\ D_2 &\equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 \end{aligned} \quad (2.4)$$

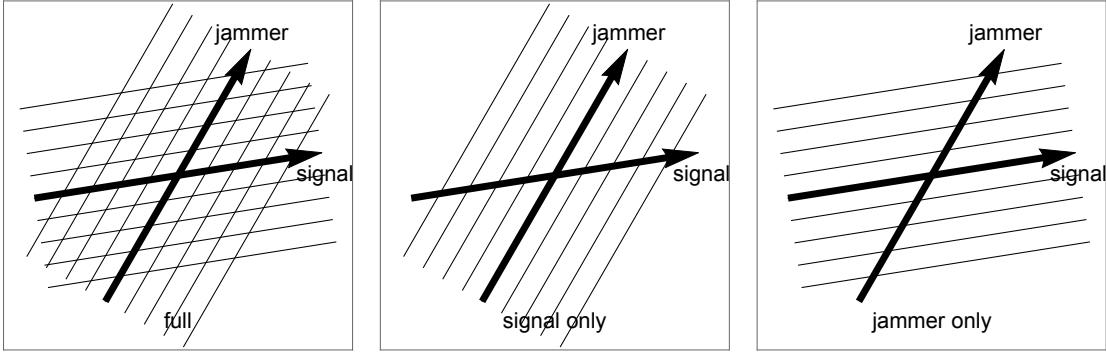
Note that  $M$  (size of codebook) and  $n$  (number of samples drawn from codebook using  $\Pr(y|\mathbf{x})$ ) are effectively model order parameters, whose values need to be chosen appropriately for each encoder optimisation problem. The properties of the optimum solution depend critically on the interplay between the statistical properties of the training data and the model order parameters  $M$  and  $n$ .

## C. Encoding Subspaces

Further constraints on the minimisation of  $D$  may be introduced to encourage  $\Pr(y|\mathbf{x})$  in Equation 2.4 to limit its dependence on  $\mathbf{x}$ . Based on an idea originally proposed in [15], it is shown in [13] how the  $\Pr(y|\mathbf{x})$  may be parameterised, so that after optimisation  $\Pr(y|\mathbf{x})$  responds only to the "large" degrees of freedom in  $\mathbf{x}$ , and is invariant with respect to the "small" degrees of freedom in  $\mathbf{x}$ . In effect, the input space is automatically split into complementary subspaces, one of which contains the "large" components of  $\mathbf{x}$ , and the other the "small" components of  $\mathbf{x}$ .

Define an explicitly normalised  $\Pr(y|\mathbf{x})$  as  $\Pr(y|\mathbf{x}) = \frac{Q(y|\mathbf{x})}{\sum_{y'=1}^M Q(y'|\mathbf{x})}$  where  $Q(y|\mathbf{x}) \geq 0$ . However,  $Q(y|\mathbf{x})$  itself needs to be modelled using a finite number of parameters in order that the values that minimise  $D_1 + D_2$  may be approximated from a finite amount of training data. It can be shown that the optimal form of  $\Pr(y|\mathbf{x})$  is piecewise linear in  $\mathbf{x}$  [10], and that for training data that lie on smooth curved manifolds the form of this solution is well approximated by a piecewise linear  $Q(y|\mathbf{x})$  of the form [11, 12]

$$Q(y|\mathbf{x}) = \begin{cases} \mathbf{w}(y) \cdot \mathbf{x} - a(y) & \mathbf{w}(y) \cdot \mathbf{x} \geq a(y) \\ 0 & \mathbf{w}(y) \cdot \mathbf{x} \leq a(y) \end{cases} \quad (2.5)$$

Figure 1: Examples of the response of  $\Pr(y|\mathbf{x})$  to signal and jammer subspaces.

which is the same as the functional form used for the neural response in [15]. However, the precise functional form of  $Q(y|\mathbf{x})$  needs to exhibit this behaviour only in the vicinity of the data manifold, so in particular it can be allowed to saturate (i.e.  $Q(y|\mathbf{x}) \rightarrow 1$ ) as  $\mathbf{w}(y) \cdot \mathbf{x} \rightarrow \infty$ . A convenient functional form that achieves this is the sigmoid, which is defined as

$$Q(y|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}(y) \cdot \mathbf{x} - b(y))} \quad (2.6)$$

This reduces the problem of minimising  $D_1 + D_2$  to one of finding the optimal values of the  $\mathbf{w}(y)$ ,  $b(y)$  and  $\mathbf{x}'(y)$ , which may be done by using the gradient descent procedure described in [8].

To encourage  $\Pr(y|\mathbf{x})$  to respond to only the “large” degrees of freedom in  $\mathbf{x}$ , it is sufficient to further constrain  $Q(y|\mathbf{x})$  as [13]

$$Q(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\hat{\mathbf{w}}(y) \cdot \mathbf{x} - \theta) / \|\mathbf{w}(y)\|)} \quad (2.7)$$

where  $\|\mathbf{w}(y)\| \equiv \sqrt{\mathbf{w}(y) \cdot \mathbf{w}(y)}$ ,  $\hat{\mathbf{w}}(y) \equiv \frac{\mathbf{w}(y)}{\|\mathbf{w}(y)\|}$ , and  $\theta$  is a threshold parameter. Thus  $\|\mathbf{w}(y)\|$  controls the slope of the sigmoid where it makes its transition from 0 to 1, and  $\theta$  specifies the radial distance from the origin where this transition occurs. As  $\theta$  is varied, the dimensionality of the two complementary subspaces (for “large” and “small” degrees of freedom in  $\mathbf{x}$ ) vary, although their sum remains fixed at  $\dim \mathbf{x}$ .

#### D. Jammer Nulling

A number of examples of typical behaviours of  $\Pr(y|\mathbf{x})$  are shown in Figure 1.

In Figure 1 the signal and jammer degrees of freedom generate a pair of non-orthogonal subspaces, whose axes are indicated in bold. The response contours of a variety of possible  $\Pr(y|\mathbf{x})$  are shown. In the “full” case a pair of  $\Pr(y|\mathbf{x})$  respond to the signal and jammer subspaces respectively. In the “signal” case a  $\Pr(y|\mathbf{x})$  responds to only the signal subspace, and is thus invariant over the

jammer subspace. In the “jammer” case the situation is the reverse of the “signal” case. This argument may readily be generalised to any number of  $\Pr(y|\mathbf{x})$ .

If it is assumed that the jammer is the “large” degree of freedom and the signal is the “small” degree of freedom, the signal and jammer subspaces may be separated by adjusting the threshold parameter  $\theta$  so that in Figure 1 the “jammer” case is obtained, in which case the  $\Pr(y|\mathbf{x})$  for  $y = 1, 2, \dots, M$  will all become invariant over the signal subspace. The jammer subspace is then spanned by the set of gradient vectors  $\nabla \Pr(y|\mathbf{x})$  for  $y = 1, 2, \dots, M$ , which can thus be used to construct a projection operator  $\mathbf{J}$  onto the jammer subspace, and a projection operator  $\mathbf{1} - \mathbf{J}$  onto the signal subspace. This definition of the projection operator may also be used in cases where the jammer and signal subspaces are curved, so that the directions of their axes are functions of  $\mathbf{x}$ , and all of the straight lines in Figure 1 are replaced by curves defining a curvilinear coordinate system and its coordinate surfaces. Note that curved subspaces are the norm rather than the exception.

### III. JAMMER NULLING SIMULATIONS

The optimisation of the encoder may be done by minimising  $D_1 + D_2$  in Equation 2.4 using gradient descent [8], using the sigmoid function in Equation 2.7 to constrain the optimisation so that it encodes only the jammer subspace [13].

In these simulations the input vector  $\mathbf{x}$  is 100-dimensional so that  $\mathbf{x} = (x_1, x_2, \dots, x_{100})$ , and each vector in the training set is independently generated as a superposition of a pair of response functions

$$x_i = a_s \frac{\sin(\frac{i-i_s}{\sigma})}{\frac{i-i_s}{\sigma}} + a_j \frac{\sin(\frac{i-i_j}{\sigma})}{\frac{i-i_j}{\sigma}} \quad (3.1)$$

where  $a_s$  is the signal amplitude that is uniformly distributed in the interval  $[-\sqrt{10^{-3}}, \sqrt{10^{-3}}]$  (this corresponds to a signal level of -30dB),  $a_j$  is the jammer amplitude that is uniformly distributed in the interval  $[-1, 1]$

(this corresponds to a jammer level of 0dB),  $i_s$  is the signal location that is chosen to be 50,  $i_j$  is the jammer location that is uniformly distributed in the interval  $[38 - \Delta, 38 + \Delta]$  ( $\Delta = 0, 2, 4$  is used in the simulations), and  $\sigma$  is the width of the response function that is chosen to be 2. The peak and the first zero of the sinc function are separated by  $\pi\sigma$ , which defines the resolution cell size. The mean jammer position and the signal position satisfy  $i_s - <i_j> = 12$ , which corresponds to a separation of  $\frac{12}{\pi\sigma} \approx 2$  resolution cells. Random noise uniformly distributed in the interval  $[-\sqrt{10^{-5}}, \sqrt{10^{-5}}]$  (this corresponds to a noise level of -50dB) is also added to each component of the training vector.

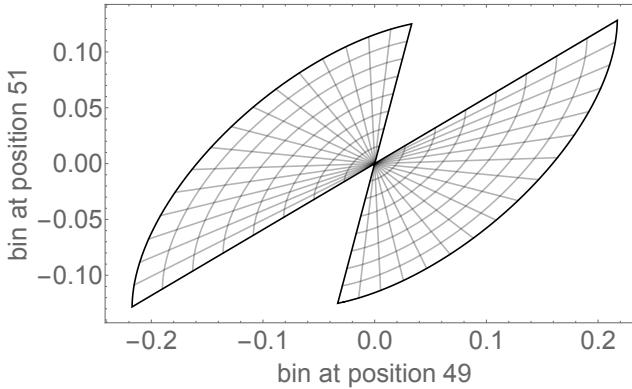


Figure 2: A two-dimensional projection of the curved manifold generated by the jammer when  $\Delta = 2$ .

In Figure 2 the 2-dimensional manifold generated by varying the jammer position over the interval  $[38 - \Delta, 38 + \Delta]$  (for  $\Delta = 2$ ), and varying the jammer amplitude over the interval  $[-1, 1]$ , is shown. Because the input vector  $\mathbf{x}$  is 100-dimensional, only a low-dimensional projection can be visualised, and the 2-dimensional vector  $(x_{49}, x_{51})$  is displayed here. The curvilinear grid traces out the coordinate surfaces of jammer position  $i_j$  and jammer amplitude  $a_j$ , and the whole diagram shows how this grid is embedded in  $(x_{49}, x_{51})$ -space. Note that the  $a_j$  dimension behaves as a “radial” coordinate (straight lines), whereas the  $i_j$  dimension behaves as an “angular” coordinate (curved lines).

In Figure 3 an encoder is trained on three different jammer scenarios  $\Delta = 0, 2, 4$ . After training the encoder is tested for how well it can be used to null a pure jammer (i.e. with no signal or noise added), where the degree of nulling is defined as the ratio of the squared lengths of the nulled input vector and the original input vector. This is a good test of the ability of the encoder to simultaneously learn the profile of the jammer and the shape of the jammer manifold which is generated by sweeping this profile over the interval  $[38 - \Delta, 38 + \Delta]$ . When  $\Delta = 0$  there is a sharp minimum at the jammer location  $i_j = 38$ , as

expected. When  $\Delta = 2$  the minimum becomes spread over the jammer locations  $i_j \in [36, 40]$ , and when  $\Delta = 4$  the minimum becomes spread even more broadly over the jammer locations  $i_j \in [34, 42]$ . All of these results are as expected.

In Figure 4 typical examples of an input vector together with how it appears after jammer nulling are shown for each of the jammer scenarios considered in Figure 3. In every case the signal is clearly revealed at its correct location after nulling the jammer.

In all of these training scenarios, one could envisage further constraining some of the properties of the encoder, in order to introduce prior knowledge of the form of the jammer and/or signal subspaces, and to thereby reduce the computational complexity of the jammer nulling. For instance, the signal subspace could be predefined, as in conventional algorithms which hold constant the response in a predefined “look direction”. Similarly, the jammer subspace could be built out of prefined subspaces which are optimised so as to maximally null the jammer(s), as in conventional algorithms in which a number of jammer “templates” are used to remove the jammer(s). In general, by choosing appropriate additional constraints, the SVQ approach to jammer nulling can be made backwardly compatible with conventional approaches.

#### IV. CONCLUSIONS

The theory of stochastic vector quantisers (SVQ) [8] has been extended to allow the quantiser to develop invariances [13], so that only “large” degrees of freedom in the input vector are represented in the code. This has been applied to the problem of encoding data vectors which are a superposition of a “large” jammer and a “small” signal, so that only the jammer is represented in the code. This allows the jammer to be subtracted from the total input vector (i.e. the jammer is nulled), leaving a residual that contains only the underlying signal. Several numerical simulations have shown how that idea works in practice, even when the jammer subspace is curved.

The main advantage of this approach to jammer nulling is that little prior knowledge of the jammer is assumed, because these properties are automatically discovered by the SVQ as it is trained on examples of input vectors. Provided that the signal is much weaker than the jammer, the SVQ acquires an internal representation of the jammer and signal manifolds, in which its code is invariant with respect to the signal. In a sense, the SVQ regards the “large” jammer as the normal type of input that it expects to receive, whereas it regards the “small” signal as an anomaly.

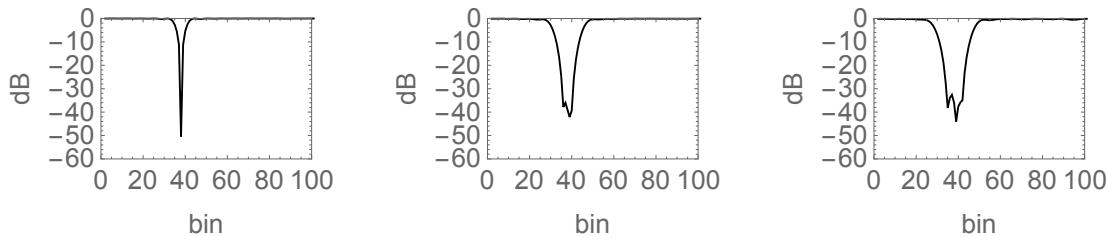


Figure 3: Plot of degree of nulling against nominal jammer location, for jammer locations that are spread over the intervals [38, 38] using  $M = 2$ , [36, 40] using  $M = 4$ , and [34, 42] using  $M = 6$ .

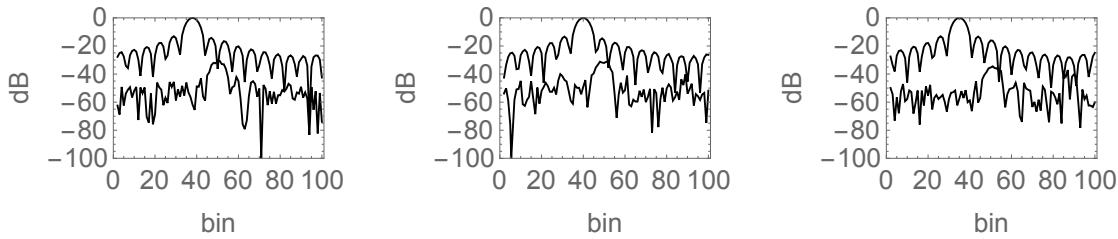


Figure 4: Plot of a typical input vector before and after jammer nulling for each of the scenarios in Figure 3.

- [1] N Farvardin, *A study of vector quantisation for noisy channels*, IEEE Transactions on Information Theory **36** (1990), no. 4, 799–809.
- [2] A Gersho and R M Gray, *Vector quantisation and signal compression*, Kluwer, Norwell, 1992.
- [3] R M Gray, *Vector quantisation*, IEEE Transactions on Acoustics, Speech, and Signal Processing Magazine **1** (1984), no. 2, 4–29.
- [4] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
- [5] H Kumazawa, M Kasahara, and T Namekawa, *A construction of vector quantisers for noisy channels*, Electronics and Engineering in Japan **67** (1984), no. 4, 39–47.
- [6] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [7] S P Luttrell, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
- [8] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
- [9] ———, *Self-organisation of multiple winner-take-all neural networks*, Connection Science **9** (1997), no. 1, 11–30.
- [10] ———, *An adaptive network for encoding data using piecewise linear functions*, Proceedings of international conference on artificial neural networks (Edinburgh), IEE, 1999, pp. 198–203.
- [11] ———, *Combining artificial neural nets: Ensemble and modular multi-net systems*, Perspectives in Neural Computing, ch. Self-organised modular neural networks for encoding data, pp. 235–263, Springer-Verlag, London, 1999.
- [12] ———, *Encoding data from a curved manifold*, Technical Report DERA/S&P/SPI/TR98116, Defence Evaluation and Research Agency, Malvern, 1999.
- [13] ———, *Invariance discovery by vector quantisation of noisy data*, Technical Report DERA/S&P/SPI/TR000070, Defence Evaluation and Research Agency, Malvern, 2000.
- [14] ———, *Stochastic vector quantisers*, (2000).
- [15] C J S Webber, *Self-organisation of transformation-invariant detectors for constituents of perceptual patterns*, Network: Computation in Neural Systems **5** (1994), no. 4, 471–496.



# Using Stochastic Encoders to Discover Structure in Data \*

S P Luttrell<sup>†</sup>

DERA, St Andrews Road, Malvern, Worcs, WR14 3PS, UK

In this paper a stochastic generalisation of the standard Linde-Buzo-Gray (LBG) approach to vector quantiser (VQ) design is presented, in which the encoder is implemented as the sampling of a vector of code indices from a probability distribution derived from the input vector, and the decoder is implemented as a superposition of reconstruction vectors. This stochastic VQ (SVQ) is optimised using a minimum mean Euclidean reconstruction distortion criterion, as in the LBG case. Numerical simulations are used to demonstrate how this leads to self-organisation of the SVQ, where different stochastically sampled code indices become associated with different input subspaces.

## I. INTRODUCTION

In vector quantisation a code book is used to encode each input vector as a corresponding code index, which is then decoded (again, using the codebook) to produce an approximate reconstruction of the original input vector. The purpose of this paper is to generalise the standard approach to vector quantiser (VQ) design, so that each input vector is encoded as a *vector* of code indices that are stochastically sampled from a probability distribution that depends on the input vector, rather than as a *single* code index that is the deterministic outcome of finding which entry in a code book is closest to the input vector. This will be called a stochastic VQ (SVQ), and it includes the standard VQ as a special case. Note that this approach is different from the various stochastic approaches that are used to train VQs, because here the codebook itself is stochastic, so the use of probability distributions is essential both during and after training.

One advantage of using the stochastic approach, which will be demonstrated in this paper, is that it automates the process of splitting high-dimensional input vectors into low-dimensional blocks before encoding them, because minimising the mean Euclidean reconstruction error can encourage different stochastically sampled code indices to become associated with different input subspaces. Another advantage is that it is very easy to connect SVQs together, by using the vector of code index probabilities computed by one SVQ as the input vector to another SVQ.

## II. THEORY

### A. Folded Markov Chains

The basic building block of the encoder/decoder model used in this paper is the folded Markov chain (FMC).

Thus an input vector  $\mathbf{x}$  is encoded as a code index vector  $\mathbf{y}$ , which is then subsequently decoded as a reconstruction  $\mathbf{x}'$  of the input vector. Both the encoding and decoding operations are allowed to be probabilistic, in the sense that  $\mathbf{y}$  is a sample drawn from  $\Pr(\mathbf{y}|\mathbf{x})$ , and  $\mathbf{x}'$  is a sample drawn from  $\Pr(\mathbf{x}'|\mathbf{y})$ , where  $\Pr(\mathbf{y}|\mathbf{x})$  and  $\Pr(\mathbf{x}'|\mathbf{y})$  are Bayes inverses of each other.

Because the chain of dependences in passing from  $\mathbf{x}$  to  $\mathbf{y}$  and then to  $\mathbf{x}'$  is first order Markov (i.e. it is described by the directed graph  $(\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}')$ , and because the two ends of this Markov chain (i.e.  $\mathbf{x}$  and  $\mathbf{x}'$ ) live in the same vector space, it is called a *folded* Markov chain (FMC). The operations that occur in an FMC are summarised in Figure 1.

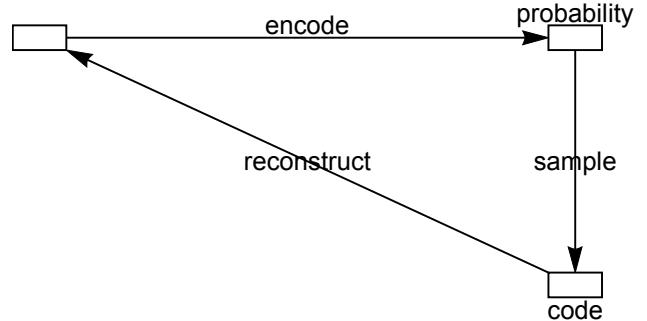


Figure 1: A folded Markov chain (FMC) in which an input vector  $\mathbf{x}$  is encoded as a code index vector  $\mathbf{y}$  that is drawn from a conditional probability  $\Pr(\mathbf{y}|\mathbf{x})$ , which is then decoded as a reconstruction vector  $\mathbf{x}'$  drawn from the Bayes inverse conditional probability  $\Pr(\mathbf{x}'|\mathbf{y})$ .

In order to ensure that the FMC encodes the input vector optimally, a measure of the reconstruction error must be minimised. There are many possible ways to define this measure, but one that is consistent with many previous results, and which also leads to many new results, is the mean Euclidean reconstruction error measure  $D$ , which may be expressed as

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1=1}^M \sum_{y_2=1}^M \cdots \sum_{y_n=1}^M \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (2.1)$$

where the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  is defined as  $\mathbf{x}'(\mathbf{y}) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ . Because of the quadratic form

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

Published in the Digest of the 5th IMA International Conference on Mathematics in Signal Processing, 18-20 December 2000, Warwick University, UK. © British Crown Copyright 2000 / DERA

<sup>†</sup>Electronic address: luttrell@signal.dera.gov.uk

of the objective function, it turns out that  $\mathbf{x}'(\mathbf{y})$  may be treated as a free parameter whose optimum value (i.e. the solution of  $\frac{\partial D}{\partial \mathbf{x}'(\mathbf{y})} = 0$ ) is  $\int d\mathbf{x} \Pr(\mathbf{x}) \mathbf{x}$ .

It has been shown that the standard VQ and topographic mappings automatically emerge as special cases when  $D$  is minimised. In this approach, topographic mappings emerge as the optimal coding scheme when the code is to be transmitted along a noisy communication channel before being decoded.

## B. High Dimensional Input Spaces

A problem with the standard VQ is that its code book grows exponentially in size as the dimensionality of the input vector is increased, so such VQs are useless for encoding extremely high dimensional input vectors. The usual solution to this problem is to manually partition the input space into a number of lower dimensional subspaces, and then to encode each of these subspaces separately. However, it would be very useful if this partitioning could be done automatically, in such a way that typically the correlations *within* each subspace were much stronger than the correlations *between* subspaces, so that the subspaces were approximately statistically independent of each other.

The key step in solving this problem is to constrain the minimisation of  $D$  in such a way as to encourage the formation of code schemes in which each component of the code vector  $\mathbf{y}$  codes a different subspace of the input vector  $\mathbf{x}$ . There are two related constraints that may be imposed on  $\Pr(\mathbf{y}|\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  which may be summarised as

$$\begin{aligned}\Pr(\mathbf{y}|\mathbf{x}) &= \Pr(y_1|\mathbf{x}) \Pr(y_2|\mathbf{x}) \cdots \Pr(y_n|\mathbf{x}) \\ \mathbf{x}'(\mathbf{y}) &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}'(y_i)\end{aligned}\quad (2.2)$$

Thus each component  $y_i$  (for  $i = 1, 2, \dots, n$  and  $1 \leq y_i \leq M$ ) is an *independent* sample drawn from the codebook using  $\Pr(y_i|\mathbf{x})$  (which is assumed to be the same function for all  $i$ ), and the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  (vector argument) is assumed to be a *superposition* of  $n$  contributions  $\mathbf{x}'(y_i)$  (scalar argument) for  $i = 1, 2, \dots, n$ . Taken together, these constraints encourage the formation of coding schemes in which independent subspaces are separately coded, as required.

This leads to an *upper bound*  $D_1 + D_2$  on the FMC objective function  $D$  (i.e.  $D \leq D_1 + D_2$ ), which may be derived as

$$\begin{aligned}D_1 &= \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \\ D_2 &= \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2\end{aligned}\quad (2.3)$$

## Using Stochastic Encoders to Discover Structure in Data

Note that  $M$  (size of codebook) and  $n$  (number of samples drawn from codebook using  $\Pr(y|\mathbf{x})$ ) are effectively model order parameters, whose values need to be chosen appropriately for each encoder optimisation problem. The properties of the optimum solution depend critically on the interplay between the statistical properties of the training data and the model order parameters  $M$  and  $n$ .

## C. Chains of Linked FMCs

The FMC illustrated in Figure 1 may be generalised to a chain of linked FMCs as shown in Figure 2.

Each stage in this chain is an FMC of the type shown in Figure 1, and the vector of probabilities (for all values of the code index) computed by each stage is used as the input vector to the next stage; there are other ways of linking the stages together, but this is the simplest possibility. The overall objective function is a weighted sum of the FMC objective functions derived from each stage. The total number of free parameters in an  $L$  stage chain is  $3L - 1$ , which is the sum of 2 free parameters for each of the  $L$  stages, plus  $L - 1$  weighting coefficients; there are  $L - 1$  rather than  $L$  weighting coefficients because the overall normalisation of the objective function does not affect the optimum solution.

## III. TYPES OF ENCODER

If the positions of a pair of objects are mutually correlated, then they can be encoded in three fundamentally different ways:

1. Factorial encoder. This encoder ignores the correlations between the objects, and encodes them as if they were two independent objects. Each code index thus encodes a single object position, so many code indices must be sampled in order to virtually guarantee that both object positions are encoded. This result is a type of independent component analysis (ICA).

2. Joint encoder. This encoder regards each possible joint placement of the two objects as a distinct configuration. Each code index thus encodes a pair of object positions, so only one code index needs to be sampled in order to guarantee that both object positions are encoded. This result is basically the same as what would be obtained by using a standard VQ.

3. Invariant encoder. This encoder regards each possible placement of the centroid of the two objects as a distinct configuration, but regards all possible object separations (for a given centroid) as being equivalent. Each code index thus encodes only the centroid of the pair of objects. This type of encoder does not arise when the objects are independent.

Each of these three possibilities is shown in Figure 3, where the diagrams are meant only to be illustrative. The correlated variables live in the large 2-dimensional rectangular region extending from bottom-left to top-right of

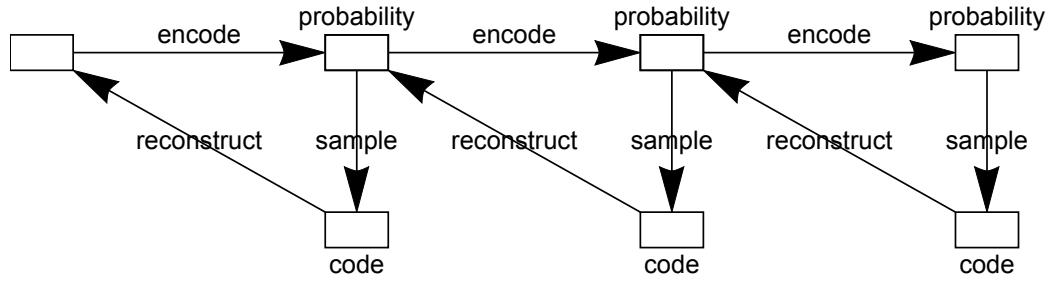


Figure 2: A chain of linked FMCs, in which the output from each stage is its vector of posterior probabilities (for all values of the code index), which is then used as the input to the next stage. Only 3 stages are shown, but any number may be used. More generally, any acyclically linked network of FMCs may be used.

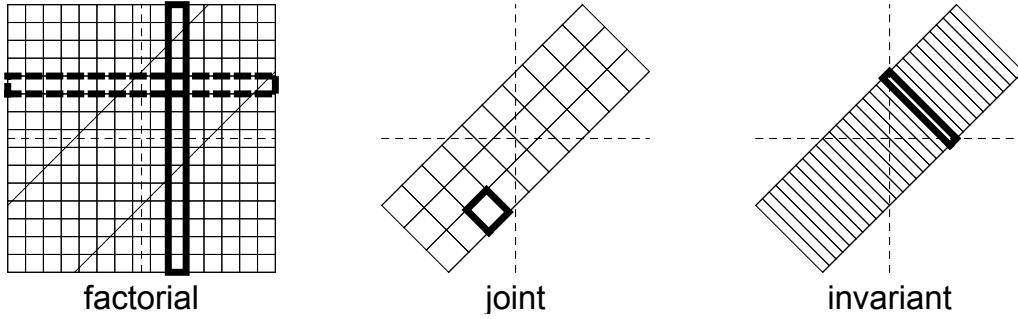


Figure 3: Three alternative ways of using 30 code indices to encode a pair of correlated variables. The typical code cells are shown in bold.

each diagram.

The factorial encoder has two orthogonal sets of long thin rectangular code cells, and the diagram shows how a pair of such cells intersect to define a small square code cell. The joint encoder behaves as a standard vector quantiser, and is illustrated as having a set of square code cells, although their shapes will not be as simple as this in practice. The invariant encoder ideally has a set of long thin rectangular code cells that encode only the long diagonal dimension.

In all three cases there is overlap between code cells. In the case of the factorial and joint encoders the overlap tends to be only between nearby code cells, whereas in the case of an invariant encoder the range of the overlap is usually much greater. In practice the optimum encoder may not be a clean example of one of the types illustrated in Figure 3.

#### IV. CONCLUSIONS

The numerical results presented in this paper show that a stochastic vector quantiser (SVQ) can self-organise to find a variety of different types of way of encoding high-

dimensional input vectors. Three fundamentally different types of encoder have been demonstrated, which differ in the way that they build a reconstruction that approximates the input vector:

1. A factorial encoder uses a reconstruction that is superposition of a *number* of vectors that each lives in a well defined input subspace, which is useful for discovering constituent objects in the input vector. This result is a type of independent component analysis (ICA).

2. A joint encoder uses a reconstruction that is a *single* vector that lives in the whole input space. This result is basically the same as what would be obtained by using a standard VQ.

3. An invariant encoder uses a reconstruction that is a *single* vector that lives in a subspace of the whole input space, so it ignores some dimensions of the input vector, which is therefore useful for discovering correlated objects whilst rejecting uninteresting fluctuations in their relative coordinates. This is similar to self-organising transformation invariant detectors described in.

More generally, the encoder will be a hybrid of these basic types, depending on the interplay between the statistical properties of the input vector and the parameter settings of the SVQ.



# Invariant Stochastic Encoders \*

S P Luttrell<sup>†</sup>

DERA, St Andrews Road, Malvern, Worcs, WR14 3PS, UK

The theory of stochastic vector quantisers (SVQ) has been extended to allow the quantiser to develop invariances, so that only “large” degrees of freedom in the input vector are represented in the code. This has been applied to the problem of encoding data vectors which are a superposition of a “large” jammer and a “small” signal, so that only the jammer is represented in the code. This allows the jammer to be subtracted from the total input vector (i.e. the jammer is nulled), leaving a residual that contains only the underlying signal. The main advantage of this approach to jammer nulling is that little prior knowledge of the jammer is assumed, because these properties are automatically discovered by the SVQ as it is trained on examples of input vectors.

## I. INTRODUCTION

In vector quantisation a code book is used to encode each input vector as a corresponding code index, which is then decoded (again, using the codebook) to produce an approximate reconstruction of the original input vector. The standard approach to vector quantiser (VQ) design may be generalised so that each input vector is encoded as a *vector* of code indices that are stochastically sampled from a probability distribution that depends on the input vector, rather than as a *single* code index that is the deterministic outcome of finding which entry in a code book is closest to the input vector. This will be called a stochastic VQ (SVQ), and it includes the standard VQ as a special case.

One advantage of using the stochastic approach is that it automates the process of splitting high-dimensional input vectors into low-dimensional blocks before encoding them, because minimising the mean Euclidean reconstruction error can encourage different stochastically sampled code indices to become associated with different input subspaces. Another advantage is that it is very easy to connect SVQs together, by using the vector of code index probabilities computed by one SVQ as the input vector to another SVQ.

## II. STOCHASTIC VECTOR QUANTISER THEORY

### A. Folded Markov Chains

The basic building block of the SVQ used in this paper is the folded Markov chain (FMC). An input vector  $\mathbf{x}$  is encoded as a code index vector  $\mathbf{y}$ , which is then subsequently decoded as a reconstruction  $\mathbf{x}'$  of the input vector. Both the encoding and decoding operations are

allowed to be probabilistic, in the sense that  $\mathbf{y}$  is a sample drawn from  $\Pr(\mathbf{y}|\mathbf{x})$ , and  $\mathbf{x}'$  is a sample drawn from  $\Pr(\mathbf{x}'|\mathbf{y})$ , where  $\Pr(\mathbf{y}|\mathbf{x})$  and  $\Pr(\mathbf{x}'|\mathbf{y})$  are Bayes' inverses of each other.

In order to ensure that the FMC encodes the input vector optimally, a measure of the reconstruction error must be minimised. There are many possible ways to define this measure, but one that is consistent with many previous results, and which also leads to many new results, is the mean Euclidean reconstruction error measure  $D$ , which may be expressed as

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1=1}^M \sum_{y_2=1}^M \cdots \sum_{y_n=1}^M \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (2.1)$$

where the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  is defined as  $\mathbf{x}'(\mathbf{y}) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ . Because of the quadratic form of the objective function, it turns out that  $\mathbf{x}'(\mathbf{y})$  may be treated as a free parameter whose optimum value (i.e. the solution of  $\frac{\partial D}{\partial \mathbf{x}'(\mathbf{y})} = 0$ ) is  $\int d\mathbf{x} \Pr(\mathbf{x}) \mathbf{x}$ .

There are two related constraints that may be imposed on  $\Pr(\mathbf{y}|\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$ . Thus each component  $y_i$  will be assumed to be an *independent* sample drawn from the codebook, and the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  will be assumed to be a *superposition* of  $n$  contributions  $\mathbf{x}'(y_i)$ . This leads to an *upper bound*  $D_1 + D_2$  on the FMC objective function  $D$  (i.e.  $D \leq D_1 + D_2$ ), which may be derived as

$$D_1 \equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \quad (2.2)$$

$$D_2 \equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2$$

### B. Encoding Subspaces

Further constraints on the minimisation of  $D$  may be introduced to encourage  $\Pr(y|\mathbf{x})$  in Equation 2.2 to limit its dependence on  $\mathbf{x}$ . It is shown how the  $\Pr(y|\mathbf{x})$  may be parameterised, so that after optimisation  $\Pr(y|\mathbf{x})$  responds only to the “large” degrees of freedom in  $\mathbf{x}$ , and

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

Published in the Digest of the 5th IMA International Conference on Mathematics in Signal Processing, 18-20 December 2000, Warwick University, UK. © British Crown Copyright 2000 / DERA

<sup>†</sup>Electronic address: luttrell@signal.dera.gov.uk

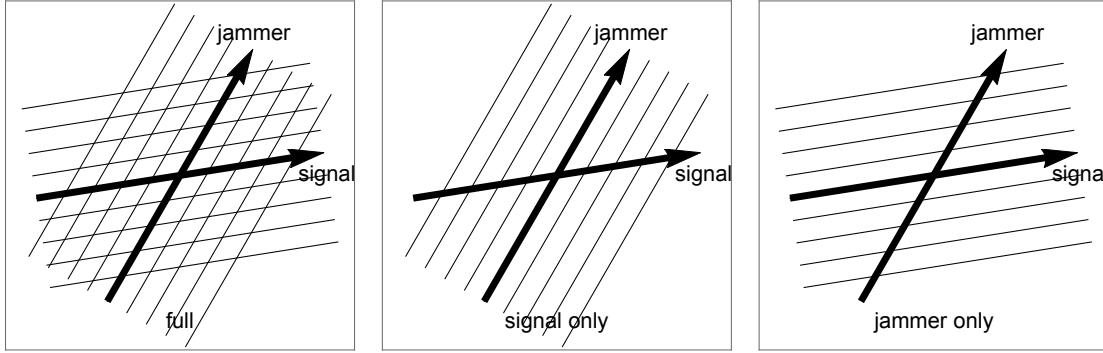


Figure 1: Examples of the response of  $\Pr(y|\mathbf{x})$  to signal and jammer subspaces.

is invariant with respect to the “small” degrees of freedom in  $\mathbf{x}$ . In effect, the input space is automatically split into complementary subspaces, one of which contains the “large” components of  $\mathbf{x}$ , and the other the “small” components of  $\mathbf{x}$ .

Define an explicitly normalised  $\Pr(y|\mathbf{x})$  as  $\Pr(y|\mathbf{x}) = \frac{Q(y|\mathbf{x})}{\sum_{y'=1}^M Q(y'|\mathbf{x})}$  where  $Q(y|\mathbf{x}) \geq 0$ . However,  $Q(y|\mathbf{x})$  itself needs to be modelled using a finite number of parameters in order that the values that minimise  $D_1 + D_2$  may be approximated from a finite amount of training data. It can be shown that the optimal form of  $\Pr(y|\mathbf{x})$  is piecewise linear in  $\mathbf{x}$ , and that for training data that lie on smooth curved manifolds the form of this solution is well approximated by a piecewise linear  $Q(y|\mathbf{x})$  of the form

$$Q(y|\mathbf{x}) = \begin{cases} \mathbf{w}(y) \cdot \mathbf{x} - a(y) & \mathbf{w}(y) \cdot \mathbf{x} \geq a(y) \\ 0 & \mathbf{w}(y) \cdot \mathbf{x} \leq a(y) \end{cases} \quad (2.3)$$

However, the precise functional form of  $Q(y|\mathbf{x})$  needs to exhibit this behaviour only in the vicinity of the data manifold, so in particular it can be allowed to saturate (i.e.  $Q(y|\mathbf{x}) \rightarrow 1$ ) as  $\mathbf{w}(y) \cdot \mathbf{x} \rightarrow \infty$ . A convenient functional form that achieves this is the sigmoid, which is defined as

$$Q(y|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}(y) \cdot \mathbf{x} - b(y))} \quad (2.4)$$

This reduces the problem of minimising  $D_1 + D_2$  to one of finding the optimal values of the  $\mathbf{w}(y)$ ,  $b(y)$  and  $\mathbf{x}'(y)$ .

To encourage  $\Pr(y|\mathbf{x})$  to respond to only the “large” degrees of freedom in  $\mathbf{x}$ , it is sufficient to further constrain  $Q(y|\mathbf{x})$  as

$$Q(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\hat{\mathbf{w}}(y) \cdot \mathbf{x} - \theta) \|\mathbf{w}(y)\|)} \quad (2.5)$$

where  $\|\mathbf{w}(y)\| \equiv \sqrt{\mathbf{w}(y) \cdot \mathbf{w}(y)}$ ,  $\hat{\mathbf{w}}(y) \equiv \frac{\mathbf{w}(y)}{\|\mathbf{w}(y)\|}$ , and  $\theta$  is a threshold parameter. Thus  $\|\mathbf{w}(y)\|$  controls the slope of the sigmoid where it makes its transition from 0 to 1, and  $\theta$  specifies the radial distance from the origin where this transition occurs. As  $\theta$  is varied, the dimensionality of the two complementary subspaces (for “large” and “small” degrees of freedom in  $\mathbf{x}$ ) vary, although their sum remains fixed at  $\dim \mathbf{x}$ .

### C. Jammer Nulling

A number of examples of typical behaviours of  $\Pr(y|\mathbf{x})$  are shown in Figure 1.

In Figure 1 the signal and jammer degrees of freedom generate a pair of non-orthogonal subspaces, whose axes are indicated in bold. The response contours of a variety of possible  $\Pr(y|\mathbf{x})$  are shown. In the “full” case a pair of  $\Pr(y|\mathbf{x})$  respond to the signal and jammer subspaces respectively. In the “signal” case a  $\Pr(y|\mathbf{x})$  responds to only the signal subspace, and is thus invariant over the jammer subspace. In the “jammer” case the situation is the reverse of the “signal” case. This argument may readily be generalised to any number of  $\Pr(y|\mathbf{x})$ .

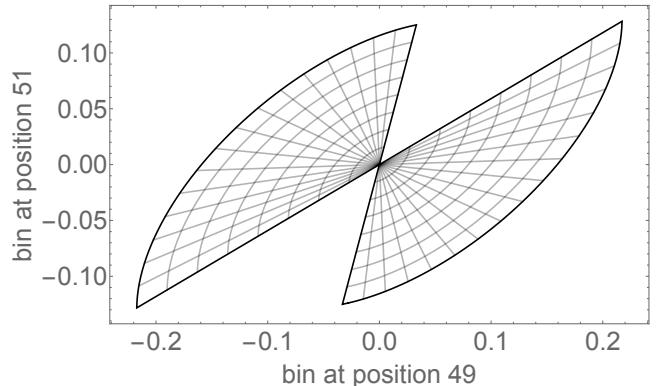


Figure 2: A two-dimensional projection of the curved manifold generated by the jammer when  $\Delta = 2$ .

If it is assumed that the jammer is the “large” degree of freedom and the signal is the “small” degree of freedom, the signal and jammer subspaces may be separated by adjusting the threshold parameter  $\theta$  so that in Figure 1 the “jammer” case is obtained, in which case the  $\Pr(y|\mathbf{x})$  for  $y = 1, 2, \dots, M$  will all become invariant over the signal subspace. The jammer subspace is then spanned by the set of gradient vectors  $\nabla \Pr(y|\mathbf{x})$  for  $y = 1, 2, \dots, M$ , which can thus be used to construct a projection operator

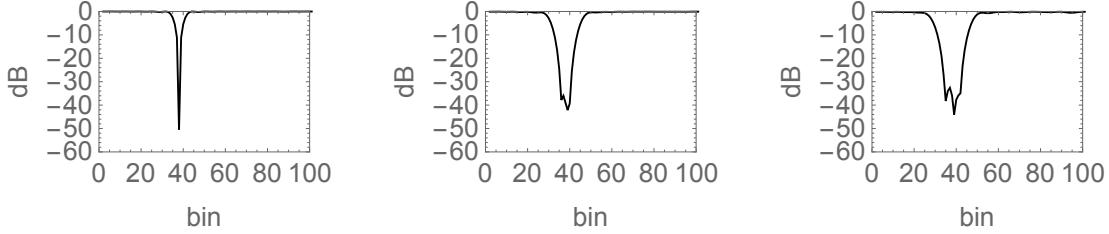


Figure 3: Plot of degree of nulling against nominal jammer location, for jammer locations that are spread over the intervals [38, 38] using  $M = 2$ , [36, 40] using  $M = 4$ , and [34, 42] using  $M = 6$ .

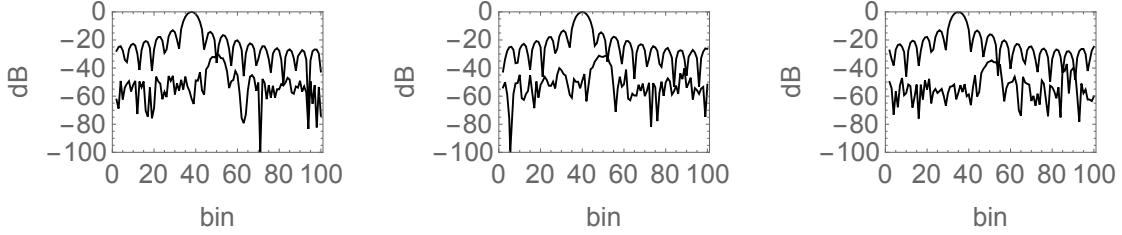


Figure 4: Plot of a typical input vector before and after jammer nulling for each of the scenarios in Figure 3.

$\mathbf{J}$  onto the jammer subspace, and a projection operator  $\mathbf{1} - \mathbf{J}$  onto the signal subspace. This definition of the projection operator may also be used in cases where the jammer and signal subspaces are curved, so that the directions of their axes are functions of  $\mathbf{x}$ , and all of the straight lines in Figure 1 are replaced by curves defining a curvilinear coordinate system and its coordinate surfaces. Note that curved subspaces are the norm rather than the exception.

### III. JAMMER NULLING SIMULATIONS

In these simulations the input vector  $\mathbf{x}$  is 100-dimensional so that  $\mathbf{x} = (x_1, x_2, \dots, x_{100})$ , and each vector in the training set is independently generated as a superposition of a pair of response functions

$$x_i = a_s \frac{\sin(\frac{i-i_s}{\sigma})}{\frac{i-i_s}{\sigma}} + a_j \frac{\sin(\frac{i-i_j}{\sigma})}{\frac{i-i_j}{\sigma}} \quad (3.1)$$

where  $a_s$  is the signal amplitude,  $a_j$  is the jammer amplitude,  $i_s$  is the signal location,  $i_j$  is the jammer location, and  $\sigma$  is the width of the response function that is chosen to be 2.

In Figure 2 the 2-dimensional manifold generated by varying the jammer position over the interval  $[38-\Delta, 38+\Delta]$  (for  $\Delta = 2$ ), and varying the jammer amplitude over the interval  $[-1, 1]$ , is shown. Because the input vector  $\mathbf{x}$  is 100-dimensional, only a low-dimensional projection can be visualised, and the 2-dimensional vector  $(x_{49}, x_{51})$  is displayed here. The curvilinear grid traces out the coordinate surfaces of jammer position  $i_j$  and jammer amplitude  $a_j$ , and the whole diagram shows how this grid

is embedded in  $(x_{49}, x_{51})$ -space. Note that the  $a_j$  dimension behaves as a “radial” coordinate (straight lines), whereas the  $i_j$  dimension behaves as an “angular” coordinate (curved lines).

In Figure 3 an encoder is trained on three different jammer scenarios  $\Delta = 0, 2, 4$ . After training the encoder is tested for how well it can be used to null a pure jammer (i.e. with no signal or noise added), where the degree of nulling is defined as the ratio of the squared lengths of the nulled input vector and the original input vector. This is a good test of the ability of the encoder to simultaneously learn the profile of the jammer and the shape of the jammer manifold which is generated by sweeping this profile over the interval  $[38 - \Delta, 38 + \Delta]$ . When  $\Delta = 0$  there is a sharp minimum at the jammer location  $i_j = 38$ , as expected. When  $\Delta = 2$  the minimum becomes spread over the jammer locations  $i_j \in [36, 40]$ , and when  $\Delta = 4$  the minimum becomes spread even more broadly over the jammer locations  $i_j \in [34, 42]$ . All of these results are as expected.

In Figure 4 typical examples of an input vector together with how it appears after jammer nulling are shown for each of the jammer scenarios considered in Figure 3. In every case the signal is clearly revealed at its correct location after nulling the jammer.

In all of these training scenarios, one could envisage further constraining some of the properties of the encoder, in order to introduce prior knowledge of the form of the jammer and/or signal subspaces, and to thereby reduce the computational complexity of the jammer nulling. For instance, the signal subspace could be predefined, as in conventional algorithms which hold constant the response in a predefined “look direction”. Simi-

larly, the jammer subspace could be built out of prefined subspaces which are optimised so as to maximally null the jammer(s), as in conventional algorithms in which a number of jammer “templates” are used to remove the jammer(s). In general, by choosing appropriate additional constraints, the SVQ approach to jammer nulling can be made backwardly compatible with conventional approaches.

#### IV. CONCLUSIONS

The theory of stochastic vector quantisers (SVQ) has been extended to allow the quantiser to develop invariances, so that only “large” degrees of freedom in the input

vector are represented in the code. This has been applied to the problem of encoding data vectors which are a superposition of a “large” jammer and a “small” signal, so that only the jammer is represented in the code. This allows the jammer to be subtracted from the total input vector (i.e. the jammer is nulled), leaving a residual that contains only the underlying signal.

The main advantage of this approach to jammer nulling is that little prior knowledge of the jammer is assumed, because these properties are automatically discovered by the SVQ as it is trained on examples of input vectors. Provided that the signal is much weaker than the jammer, the SVQ acquires an internal representation of the jammer and signal manifolds, in which its code is invariant with respect to the signal.

# Adaptive Subspace Encoders Using Stochastic Vector Quantisers \*

S P Luttrell

Defence Evaluation and Research Agency, Malvern, U.K.

The theory of stochastic vector quantisers (SVQ) has been extended to allow the quantiser to develop invariances, so that only “large” degrees of freedom in the input vector are represented in the code. In particular, this may be applied to the problem of encoding data that predominantly lies on a manifold whose dimensionality is much less than the full dimensionality of the data space. In effect, the approach learns a subspace that approximates the local manifold subspace at each point on the manifold, so it is an example of an adaptive subspace approach. To illustrate this approach, the technique has been applied to the problem of encoding data vectors that are a superposition of a “large” jammer and a “small” signal, so that only the jammer is represented in the code. This allows the jammer to be subtracted from the total input vector (i.e. the jammer is nulled), leaving a residual that contains only the underlying signal.

## I. INTRODUCTION

In vector quantisation a code book is used to encode each input vector as a corresponding code index, which is then decoded (again, using the codebook) to produce an approximate reconstruction of the original input vector [4, 5]. The standard approach to vector quantiser (VQ) design [7] may be generalised [9] so that each input vector is encoded as a *vector* of code indices that are stochastically sampled from a probability distribution that depends on the input vector, rather than as a *single* code index that is the deterministic outcome of finding which entry in a code book is closest to the input vector. This will be called a stochastic vector quantiser (SVQ), and it includes the standard VQ as a special case.

One advantage of using the stochastic approach is that it automates the process of splitting high-dimensional input vectors into low-dimensional blocks before encoding them, because minimising the mean Euclidean reconstruction error can encourage different stochastically sampled code indices to become associated with different input subspaces [10, 12]. Another advantage is that it is very easy to connect SVQs together, by using the vector of code index probabilities computed by one SVQ as the input vector to another SVQ [11].

SVQ theory will be extended to the case of encoding noisy (or distorted) data, with the intention of subsequently reconstructing an approximation to the noiseless data. In particular, this may be applied to the problem of encoding data that predominantly lie on a manifold whose dimensionality is much less than the full dimensionality of the data space. In effect, the approach learns a subspace that approximates the local manifold subspace at each point on the manifold, so it is an example of an adaptive subspace approach [1, 2, 6]. This theory is then applied to the problem of encoding data

vectors which are a superposition of a “large” jammer and a “small” signal, where the signal is regarded as a distortion superimposed on the jammer, rather than the other way around. The reconstruction is then an approximation to the jammer, which can thus be subtracted from the original data to reveal the underlying signal of interest.

In section II the underlying theory of SVQs is developed together with its extension to the encoding of noisy data, and in section III some simulations illustrating the application of SVQs to the nulling of jammers are presented.

## II. STOCHASTIC VECTOR QUANTISER THEORY

In section II A the basic theory of folded Markov chains (FMC) is given [8], in section II B FMC theory is extended to the case of encoding noisy or distorted data with the intention of eventually recovering the undistorted data, in section II C this extended theory is applied to the problem of encoding data that contain unwanted “nuisance degrees of freedom”, and in section II D some constraints on the optimisation of the encoder are introduced to encourage the encoder to disregard the nuisance degrees of freedom (i.e. discover invariances).

### A. Folded Markov Chains

The basic building block of the SVQ used in this paper is the folded Markov chain (FMC) [8]. An input vector  $\mathbf{x}$  is encoded as a code index vector  $\mathbf{y}$ , which is then subsequently decoded as a reconstruction  $\mathbf{x}'$  of the input vector. Both the encoding and decoding operations are allowed to be probabilistic, in the sense that  $\mathbf{y}$  is a sample drawn from  $\text{Pr}(\mathbf{y}|\mathbf{x})$ , and  $\mathbf{x}'$  is a sample drawn from  $\text{Pr}(\mathbf{x}'|\mathbf{y})$ , where  $\text{Pr}(\mathbf{y}|\mathbf{x})$  and  $\text{Pr}(\mathbf{x}'|\mathbf{y})$  are Bayes’ inverses of each other, as given by  $\text{Pr}(\mathbf{x}'|\mathbf{y}) = \frac{\text{Pr}(\mathbf{y}|\mathbf{x}) \text{Pr}(\mathbf{x})}{\int dz \text{Pr}(\mathbf{y}|z) \text{Pr}(z)}$ , and  $\text{Pr}(\mathbf{x})$  is the prior probability from which  $\mathbf{x}$  is sampled. In order to ensure that the FMC encodes the input

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

This appeared in Advances in Self-Organising Maps, Springer, N Allinson and H Yin H and L Allinson and J Slack (eds.), 102-109, 2001. ©British Crown Copyright 2001/DERA.

vector optimally, a measure of the reconstruction error must be minimised. There are many possible ways to define this measure, but one that is consistent with many previous results, and which also leads to many new results, is the mean Euclidean reconstruction error measure  $D$ , which is defined as [8]

$$D \equiv \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \int d\mathbf{x}' \Pr(\mathbf{x}'|\mathbf{y}) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (1)$$

where  $\mathbf{y} = (y_1, y_2, \dots, y_n)$ ,  $1 \leq y_i \leq M$  is assumed,  $\Pr(\mathbf{x}) \Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x}'|\mathbf{y})$  is the joint probability that the FMC has state  $(\mathbf{x}, \mathbf{y}, \mathbf{x}')$ ,  $\|\mathbf{x} - \mathbf{x}'\|^2$  is the Euclidean reconstruction error, and  $\int d\mathbf{x} \sum_{y_1}^M \sum_{y_2}^M \dots \sum_{y_n}^M \int d\mathbf{x}' (\dots)$  sums over all possible states of the FMC (weighted by the joint probability). The Bayes' inverse probability  $\Pr(\mathbf{x}'|\mathbf{y})$  may be integrated out of this expression for  $D$  to yield [8]

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (2)$$

where the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  is defined as  $\mathbf{x}'(\mathbf{y}) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ . Because of the quadratic form

of the objective function, it turns out that  $\mathbf{x}'(\mathbf{y})$  may be treated as a free parameter whose optimum value (i.e. the solution of  $\frac{\partial D}{\partial \mathbf{x}'(\mathbf{y})} = 0$ ) is  $\int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ , as required.

## B. Noisy Data

The FMC approach can be generalised to the problem of encoding noisy or distorted data, with the intention of eventually recovering the undistorted data. This generalisation is based on the results reported in [3]. The input vector is  $\mathbf{x}_0$ , which is converted into the distorted input vector  $\mathbf{x}$  by a distortion process  $\Pr(\mathbf{x}|\mathbf{x}_0)$ , which is then encoded as a code index vector  $\mathbf{y}$ , which is then subsequently decoded as a reconstruction  $\mathbf{x}'_0$  of the original input vector. This is described by the directed graph  $\mathbf{x}_0 \rightarrow \mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}'_0$ . The mean Euclidean reconstruction error measure  $D$  becomes (compare equation 1)

---


$$D = \int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{x}_0) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \int d\mathbf{x}'_0 \Pr(\mathbf{x}'_0|\mathbf{y}) \|\mathbf{x}_0 - \mathbf{x}'_0\|^2 \quad (3)$$

The Bayes' inverse probability  $\Pr(\mathbf{x}'_0|\mathbf{y})$  may be integrated out of this expression for  $D$  to yield (compare equation 2)

$$D = 2 \int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{x}_0) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{y})\|^2 \quad (4)$$


---

where the reconstruction vector  $\mathbf{x}'_0(\mathbf{y})$  is defined as  $\mathbf{x}'_0(\mathbf{y}) \equiv \int d\mathbf{x}_0 \Pr(\mathbf{x}_0|\mathbf{y}) \mathbf{x}_0$ , which may be treated as a free parameter. Bayes' theorem  $\Pr(\mathbf{x}_0) \Pr(\mathbf{x}|\mathbf{x}_0) = \Pr(\mathbf{x}) \Pr(\mathbf{x}_0|\mathbf{x})$  may be used to integrate out  $\mathbf{x}_0$  to yield

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x}_0(\mathbf{x}) - \mathbf{x}'_0(\mathbf{y})\|^2 + \text{constant} \quad (5)$$

where  $\mathbf{x}_0(\mathbf{x})$  is defined as  $\mathbf{x}_0(\mathbf{x}) \equiv \int d\mathbf{x}_0 \Pr(\mathbf{x}_0|\mathbf{x}) \mathbf{x}_0$ .

---

## C. Nuisance Degrees of Freedom

For convenience, split up the input space into (possibly non-orthogonal) subspaces as  $(\mathbf{x}_0, \mathbf{x}_\perp)$ , where all of the distortion is contained in  $\mathbf{x}_\perp$ , which requires that any distortion that lies in the  $\mathbf{x}_0$  subspace is regarded as part of the undistorted input. The expression for  $D$  becomes (compare equation 4).

$$D = 2 \int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \int d\mathbf{x}_\perp \Pr(\mathbf{x}_\perp|\mathbf{x}_0) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\perp) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{y})\|^2 \quad (6)$$

Consider the related optimisation problem in which an attempt to reconstruct  $(\mathbf{x}_0, \mathbf{x}_\perp)$  is made. The corresponding objective function may be obtained by modifying equation 6, where the cross-term arising from non-orthogonal

$(\mathbf{x}_0, \mathbf{x}_\perp)$  is omitted.

$$D = 2 \int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \int d\mathbf{x}_\perp \Pr(\mathbf{x}_\perp | \mathbf{x}_0) \sum_y \Pr(y | \mathbf{x}_0, \mathbf{x}_\perp) \left( \|\mathbf{x}_0 - \mathbf{x}'_0(y)\|^2 + \|\mathbf{x}_\perp - \mathbf{x}'_\perp(y)\|^2 \right) \quad (7)$$

Now introduce the following assumptions

$$\begin{aligned} \Pr(y | \mathbf{x}_0, \mathbf{x}_\perp) &= \Pr(y | \mathbf{x}_0) \\ \Pr(\mathbf{x}_\perp | \mathbf{x}_0) &= \Pr(\mathbf{x}_\perp) \end{aligned} \quad (8)$$

which allow the objective function  $D$  (see equation 7) to be replaced by the equivalent objective function

$$D = 2 \int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \sum_y \Pr(y | \mathbf{x}_0) \|\mathbf{x}_0 - \mathbf{x}'_0(y)\|^2 + \text{constant} \quad (9)$$

This is the standard FMC objective function (compare equation 2) for encoding and reconstructing the undistorted input, for which the directed graph is  $\mathbf{x}_0 \rightarrow \mathbf{y} \rightarrow \mathbf{x}'_0$ . Note that, under the stated assumptions, the simplification in equation 9 occurs even if the two subspaces are not orthogonal to each other, the potential cross-term  $\int d\mathbf{x}_\perp \Pr(\mathbf{x}_\perp | \mathbf{x}_0) (\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{y})) \cdot (\mathbf{x}_\perp - \mathbf{x}'_\perp(\mathbf{y}))$  in equation 7 is zero.

#### D. Optimisation Constraints

Henceforth, only the scalar case will be considered, so the vector  $\mathbf{y}$  is now replaced by the scalar  $y$  ( $1 \leq y \leq M$ ). In order to implement a practical optimisation procedure for minimising  $D$  it is necessary to introduce a variety of assumptions and constraints. Because  $\Pr(y | \mathbf{x})$  is a probability it satisfies  $\Pr(y | \mathbf{x}) \geq 0$  and  $\sum_{y=1}^M \Pr(y | \mathbf{x}) = 1$ , which is guaranteed if  $\Pr(y | \mathbf{x})$  is written as  $\Pr(y | \mathbf{x}) = \frac{Q(y | \mathbf{x})}{\sum_{y'=1}^M Q(y' | \mathbf{x})}$ , where  $Q(y | \mathbf{x}) \geq 0$ . This removes the need to explicitly impose the constraint  $\sum_{y=1}^M \Pr(y | \mathbf{x}) = 1$  during optimisation. The  $Q(y | \mathbf{x})$  are the unnormalised likelihoods of sampling code index  $y$  from the code book. However,  $Q(y | \mathbf{x})$  itself needs to be described by a finite number of parameters in order that the values that minimise  $D$  may be derived from a finite amount of training data. It can be shown that the optimal form of  $\Pr(y | \mathbf{x})$  is piecewise linear in  $\mathbf{x}$  [11], and that for training data that lie on smooth curved manifolds the form of this solution is well approximated by a piecewise linear  $Q(y | \mathbf{x})$  of the form [12]

$$Q(y | \mathbf{x}) = \begin{cases} \mathbf{w}(y) \cdot \mathbf{x} - a(y) & \mathbf{w}(y) \cdot \mathbf{x} \geq a(y) \\ 0 & \mathbf{w}(y) \cdot \mathbf{x} \leq a(y) \end{cases} \quad (10)$$

However, the precise functional form of  $Q(y | \mathbf{x})$  needs to exhibit this behaviour only in the vicinity of the data manifold, so in particular it can be allowed to saturate (i.e.  $Q(y | \mathbf{x}) \rightarrow 1$ ) as  $\mathbf{w}(y) \cdot \mathbf{x} \rightarrow \infty$ . A convenient

functional form that achieves this is the sigmoid, which is defined as  $Q(y | \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}(y) \cdot \mathbf{x} - b(y))}$ . This reduces the problem of minimising  $D$  to one of finding the optimal values of the  $\mathbf{w}(y)$ ,  $b(y)$  and  $\mathbf{x}'(y)$ . This may be done by using the gradient descent procedure described in [9].

If the input is an *undistorted* signal (i.e.  $\mathbf{x} = (\mathbf{x}_0, \mathbf{0})$ ) which lies on a smooth curved manifold, then the sigmoids can cooperate in encoding this input as illustrated in figure 1, where the sigmoid threshold planes  $\mathbf{w}(y) \cdot \mathbf{x} + b(y) = 0$  are shown slicing pieces off the curved manifold [12].

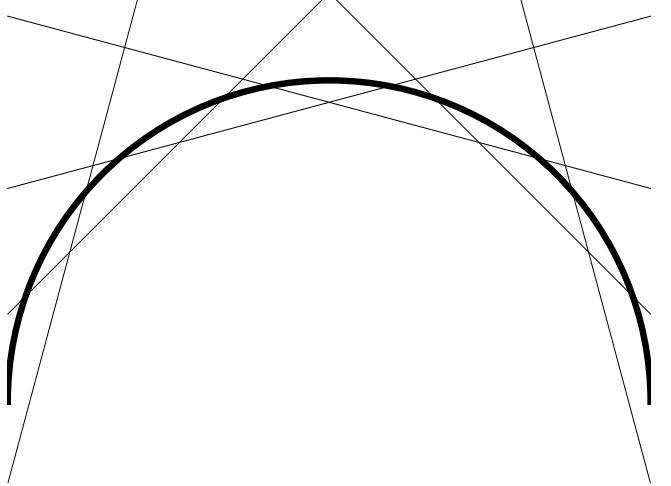


Figure 1: Illustration of how a number of sigmoids can cooperate to slice pieces off a signal manifold.

The additional constraints that are required in order to implement the behaviour described in section II C will now be described. Thus the constraints must be such that the encoder disregards (or is invariant with respect to) the nuisance degrees of freedom  $\mathbf{x}_\perp$  in the full input vector  $(\mathbf{x}_0, \mathbf{x}_\perp)$ . However, without knowing  $\Pr(\mathbf{x}_0, \mathbf{x}_\perp)$  in advance (which would allow  $\mathbf{x}_0(\mathbf{x})$  in equation 5 to be calculated), it is not possible to give a general approach that works in all cases. At best, an empirical approach must be used. A very simple and useful constraint is to impose a threshold constraint on the sigmoid function, which forces the value of the sigmoid to lie exactly halfway up its slope when the norm of its input vector is  $\theta$ . This is achieved by choosing  $b(y) = -\theta \|\mathbf{w}(y)\|$ , so that

$$Q(y | \mathbf{x}) = \frac{1}{1 + \exp(-(\hat{\mathbf{w}}(y) \cdot \mathbf{x} - \theta) \|\mathbf{w}(y)\|)} \quad (11)$$

where  $\|\mathbf{w}(y)\| \equiv \sqrt{\mathbf{w}(y) \cdot \mathbf{w}(y)}$  and  $\hat{\mathbf{w}}(y) \equiv \frac{\mathbf{w}(y)}{\|\mathbf{w}(y)\|}$ .

If the input is a *distorted* signal (i.e.  $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_\perp)$ ) which lies on a “thickened” version of the smooth curved manifold of figure 1 (the thickness represents the nuisance degrees of freedom), then the sigmoids can cooperate in encoding this input as illustrated in figure 2, where the sigmoid threshold planes  $\hat{\mathbf{w}}(y).\mathbf{x} = \theta$  are shown slicing pieces off the curved manifold in a way that disregards the nuisance degrees of freedom.

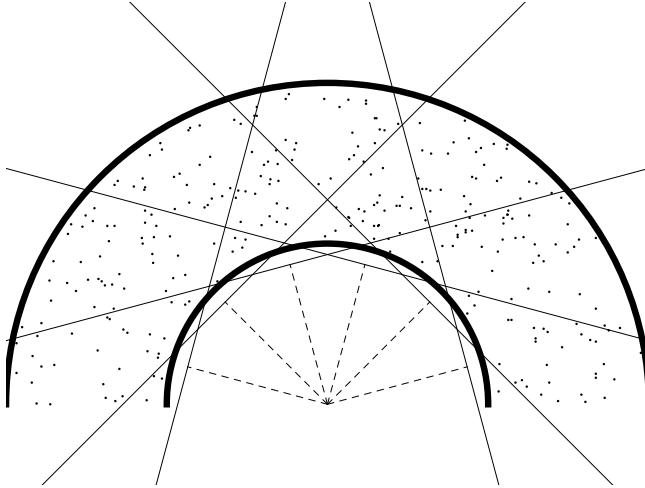


Figure 2: Illustration of how a number of sigmoids can cooperate to slice pieces off a signal manifold thickened by nuisance degrees of freedom.

Note that in this figure 2 the representation of thickening is not complete, because it can actually occur in any direction orthogonal to the manifold, including directions orthogonal to the space in which the manifold is embedded; the radial direction in figure 2 does not include this latter possibility.

### III. JAMMER NULLING SIMULATIONS

Consider the case of a weak signal that is masked by a strong jammer. If it is assumed that the jammer is the “large” degree of freedom and the signal is the “small” degree of freedom, the signal and jammer subspaces may be separated by adjusting the threshold parameter  $\theta$ , in which case the  $\Pr(y|\mathbf{x})$  for  $y = 1, 2, \dots, M$  will all become invariant over the signal subspace. The jammer subspace is then spanned by the set of gradient vectors  $\nabla \Pr(y|\mathbf{x})$  for  $y = 1, 2, \dots, M$ , which can thus be used to construct a projection operator  $\mathbf{J}$  onto the jammer subspace, and a projection operator  $\mathbf{1} - \mathbf{J}$  onto the signal subspace. This definition of the projection operator may also be used in cases where the jammer and signal subspaces are curved, so that the directions of their axes are functions of  $\mathbf{x}$  defining a curvilinear coordinate system and its coordinate surfaces. Note that curved subspaces are the norm rather than the exception.

In these simulations the input vector  $\mathbf{x}$  is 100-dimensional so that  $\mathbf{x} = (x_1, x_2, \dots, x_{100})$ , and each

vector in the training set is independently generated as a superposition of a pair of response functions  $x_i = a_s \frac{\sin(\frac{i-i_s}{\sigma})}{\frac{i-i_s}{\sigma}} + a_j \frac{\sin(\frac{i-i_j}{\sigma})}{\frac{i-i_j}{\sigma}}$ , where  $a_s$  is the signal amplitude that is uniformly distributed in the interval  $[-\sqrt{10^{-3}}, \sqrt{10^{-3}}]$  (this corresponds to a signal level of -30dB),  $a_j$  is the jammer amplitude that is uniformly distributed in the interval  $[-1, 1]$  (this corresponds to a jammer level of 0dB),  $i_s$  is the signal location that is chosen to be 50,  $i_j$  is the jammer location that is uniformly distributed in the interval  $[38 - \Delta, 38 + \Delta]$  ( $\Delta = 0, 2, 4$  is used in the simulations), and  $\sigma$  is the width of the response function that is chosen to be 2. The peak and the first zero of the sinc function are separated by  $\pi\sigma$ , which defines the resolution cell size. The mean jammer position and the signal position satisfy  $i_s - \langle i_j \rangle \geq 12$ , which corresponds to a separation of  $\frac{12}{\pi\sigma} \approx 2$  resolution cells. Random noise uniformly distributed in the interval  $[-\sqrt{10^{-5}}, \sqrt{10^{-5}}]$  (this corresponds to a noise level of -50dB) is also added to each component of the training vector.

In figure 3 typical examples of an input vector together with how it appears after jammer nulling are shown for each of the jammer scenarios  $\Delta = 0, 2, 4$ . In every case the signal is clearly revealed at its correct location after nulling the jammer.

### IV. CONCLUSIONS

The theory of stochastic vector quantisers (SVQ) [9] has been extended to allow the quantiser to develop invariances, so that only “large” degrees of freedom in the input vector are represented in the code. In particular, this may be applied to the problem of encoding data that predominantly lies on a manifold whose dimensionality is much less than the full dimensionality of the data space. In effect, the approach learns a subspace that approximates the local manifold subspace at each point on the manifold, so it is an example of an adaptive subspace approach. A nice property of the SVQ approach is that it is a natural generalisation of the standard approach to VQ design [7], in which the deterministic code book is replaced by a stochastic code book. This apparently innocuous change leads to wealth of new results.

To illustrate the SVQ approach, the technique has been applied to the problem of encoding data vectors that are a superposition of a “large” jammer and a “small” signal, so that only the jammer is represented in the code. This allows the jammer to be subtracted from the total input vector (i.e. the jammer is nulled), leaving a residual that contains only the underlying signal. Several numerical simulations have shown how that idea works in practice, even when the jammer location is uncertain so that the jammer subspace is curved.

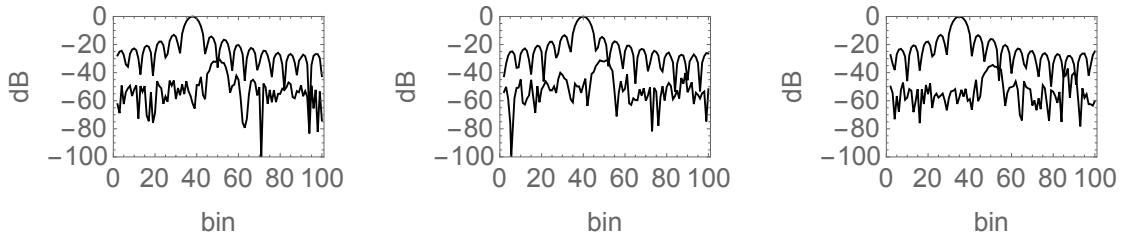


Figure 3: Plot of a typical input vector before and after jammer nulling for each of the scenarios  $\Delta = 0, 2, 4$ .

## V. ACKNOWLEDGEMENT

This work was carried out as part of Technology Group 10 of the MoD Corporate Research Programme.

- [1] D de Ridder, J Kittler, O Lemmers, and R W Duin, *Proceedings of international conference on pattern recognition*, ch. The adaptive subspace map for texture segmentation, pp. 216–220, IEEE Computer Society Press, Los Alamitos, 2000.
- [2] D de Ridder, O Lemmers, R W Duin, and J Kittler, *Proceedings of joint IAPR international workshops SSPR2000 and SPR2000*, Lecture notes in computer science, vol. 1876, ch. The adaptive subspace map for image description and image database retrieval, pp. 94–103, Springer-Verlag, Berlin, 2000.
- [3] Y Ephraim and R M Gray, *A unified approach for encoding clean and noisy sources by means of waveform and autoregressive model vector quantisation*, IEEE Transactions on Information Theory **34** (1988), no. 4, 826–834.
- [4] A Gersho and R M Gray, *Vector quantisation and signal compression*, Kluwer, Norwell, 1992.
- [5] R M Gray, *Vector quantisation*, IEEE Transactions on Acoustics, Speech, and Signal Processing Magazine **1** (1984), no. 2, 4–29.
- [6] T Kohonen, S Kaski, and H Lappalainen, *Self-organised formation of various invariant-feature filters in the adaptive-subspace SOM*, Neural Computation **9** (1997), no. 6, 1321–1344.
- [7] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [8] S P Luttrell, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
- [9] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
- [10] ———, *Self-organisation of multiple winner-take-all neural networks*, Connection Science **9** (1997), no. 1, 11–30.
- [11] ———, *An adaptive network for encoding data using piecewise linear functions*, Proceedings of international conference on artificial neural networks (Edinburgh), IEE, 1999, pp. 198–203.
- [12] ———, *Combining artificial neural nets: Ensemble and modular multi-net systems*, Perspectives in Neural Computing, ch. Self-organised modular neural networks for encoding data, pp. 235–263, Springer-Verlag, London, 1999.



# Adaptive Sensor Fusion Using Stochastic Vector Quantisers \*

S P Luttrell<sup>†</sup>  
DERA, Malvern.

In this paper a stochastic generalisation of the standard Linde-Buzo-Gray (LBG) approach to vector quantiser (VQ) design is presented, in which the encoder is implemented as the sampling of a vector of code indices from a probability distribution derived from the input vector, and the decoder is implemented as a superposition of reconstruction vectors. This stochastic VQ (SVQ) is optimised using a minimum mean Euclidean reconstruction distortion criterion, as in the LBG case. Numerical simulations with stereo pairs of images are used to demonstrate how this can lead to various types of self-organisation of the SVQ, each of which encodes and fuses the information in the stereo pair in a characteristic way.

## I. INTRODUCTION

In vector quantisation a code book is used to encode each input vector as a corresponding code index, which is then decoded (again, using the codebook) to produce an approximate reconstruction of the original input vector [2, 3]. The purpose of this paper is to generalise the standard approach to vector quantiser (VQ) design [6], so that each input vector is encoded as a vector of code indices that are stochastically sampled from a probability distribution that depends on the input vector, rather than as a single code index that is the deterministic outcome of finding which entry in a code book is closest to the input vector. This will be called a stochastic vector quantiser (SVQ), and it includes the standard VQ as a special case.

One advantage of using the stochastic approach, that will be demonstrated in this paper, is that it automates the process of discovering structure in data, such as the presence of recurring patterns or correlations across multiple sensors. This point may be illustrated by training an SVQ on stereo pairs of images, in which a single scene is imaged by two spatially separated sensors, so that parallax leads to stereo disparity between the two images. This stereo disparity cannot be recovered by processing data from a single sensor, but it can be recovered by suitably comparing the data received by the two sensors. Stereo disparity can either be manually extracted by using our understanding of the geometry of the imaging process, or can be automatically extracted by using an SVQ to discover the stereo disparity degree of freedom in the data.

In section II various pieces of previously published theory are unified to give a coherent account of SVQs. In section III the results of some numerical simulations are presented, that demonstrate how the code indices in a SVQ can become associated with stereo pairs of images.

## II. THEORY

In this section various pieces of the previously published theory of folded Markov chains (FMC) are unified to establish a coherent framework for modelling SVQs. In section II A the basic theory of FMCs is given [7], and in section II B it is extended to the case of high-dimensional input data [8].

### A. Folded Markov Chains

The basic building block of the encoder/decoder model used in this paper is the folded Markov chain (FMC) [7]. Thus an input vector  $\mathbf{x}$  is encoded as a code index vector  $\mathbf{y}$ , which is then subsequently decoded as a reconstruction  $\mathbf{x}'$  of the input vector. Both the encoding and decoding operations are allowed to be probabilistic, in the sense that  $\mathbf{y}$  is a sample drawn from  $\Pr(\mathbf{y}|\mathbf{x})$ , and  $\mathbf{x}'$  is a sample drawn from  $\Pr(\mathbf{x}'|\mathbf{y})$ , where  $\Pr(\mathbf{y}|\mathbf{x})$  and  $\Pr(\mathbf{x}'|\mathbf{y})$  are Bayes' inverses of each other, as given by  $\Pr(\mathbf{x}'|\mathbf{y}) = \frac{\Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x})}{\int dz \Pr(\mathbf{y}|z) \Pr(z)}$ , and  $\Pr(\mathbf{x})$  is the prior probability from which  $\mathbf{x}$  was sampled. Because the chain of dependences in passing from  $\mathbf{x}$  to  $\mathbf{y}$  and then to  $\mathbf{x}'$  is first order Markov (i.e. it is described by the directed graph  $(\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}')$ , and because the two ends of this Markov chain (i.e.  $\mathbf{x}$  and  $\mathbf{x}'$ ) live in the same vector space, it is called a *folded* Markov chain (FMC). The operations that occur in an FMC are summarised in Figure 1.

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

This appeared in Proceedings of the DERA/IEE Workshop on Intelligent Sensor Processing, Birmingham, 2/1-2/6, 2001. © British Crown Copyright 2001/DERA.

<sup>†</sup>Electronic address: luttrell@signal.dera.gov.uk

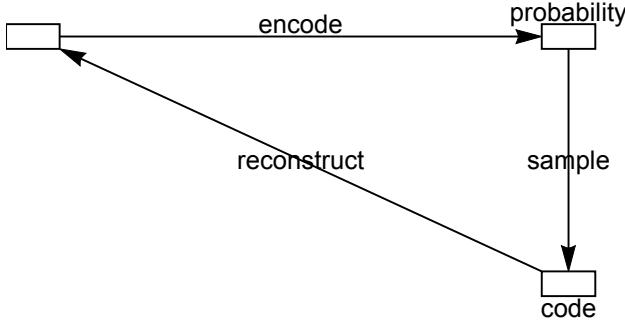


Figure 1: A folded Markov chain (FMC) in which an input vector  $\mathbf{x}$  is encoded as a code index vector  $\mathbf{y}$  that is drawn from a conditional probability  $\Pr(\mathbf{y}|\mathbf{x})$ , which is then decoded as a reconstruction vector  $\mathbf{x}'$  drawn from the Bayes' inverse conditional probability  $\Pr(\mathbf{x}'|\mathbf{y})$ .

In order to ensure that the FMC encodes the input vector optimally, a measure of the reconstruction error must be minimised. There are many possible ways to define this measure, but one that is consistent with many previous results, and which also leads to many new results, is the mean Euclidean reconstruction error measure  $D$ , which is defined as

$$D \equiv \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1=1}^M \sum_{y_2=1}^M \cdots \sum_{y_n=1}^M \Pr(\mathbf{y}|\mathbf{x}) \int d\mathbf{x}' \Pr(\mathbf{x}'|\mathbf{y}) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (2.1)$$

where  $\mathbf{y} = (y_1, y_2, \dots, y_n)$   $1 \leq y_i \leq M$  is assumed,  $\Pr(\mathbf{x}) \Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x}'|\mathbf{y})$  is the joint probability that the FMC has state  $(\mathbf{x}, \mathbf{y}, \mathbf{x}')$ ,  $\|\mathbf{x} - \mathbf{x}'\|^2$  is the Euclidean reconstruction error, and  $\int d\mathbf{x} \sum_{y_1=1}^M \sum_{y_2=1}^M \cdots \sum_{y_n=1}^M \int d\mathbf{x}' (\dots)$  sums over all possible states of the FMC (weighted by the joint probability).

The Bayes' inverse probability  $\Pr(\mathbf{x}'|\mathbf{y})$  may be integrated out of this expression for  $D$  to yield

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1=1}^M \sum_{y_2=1}^M \cdots \sum_{y_n=1}^M \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (2.2)$$

where the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  is defined as  $\mathbf{x}'(\mathbf{y}) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ . Because of the quadratic form of the objective function, it turns out that  $\mathbf{x}'(\mathbf{y})$  may be treated as a free parameter whose optimum value (i.e. the solution of  $\frac{\partial D}{\partial \mathbf{x}'(\mathbf{y})} = 0$ ) is  $\int d\mathbf{x} \Pr(\mathbf{x}) \mathbf{x}$ , as required.

It was shown in [7] that the standard VQ [6] and topographic mappings [4] automatically emerge as special cases when  $D$  is minimised. In this approach, topographic mappings emerge as the optimal coding scheme when the code is to be transmitted along a noisy communication channel before being decoded [1, 5].

## B. High Dimensional Input Spaces

A problem with the standard VQ is that its code book grows exponentially in size as the dimensionality of the input vector is increased, assuming that the contribution

to the reconstruction error from each input dimension is held constant. This means that such VQs are useless for encoding extremely high dimensional input vectors, such as images. The usual solution to this problem is to manually partition the input space into a number of lower dimensional subspaces, and then to encode each of these subspaces separately. However, it would be very useful if this partitioning could be done automatically, in such a way that typically the correlations *within* each subspace were much stronger than the correlations *between* subspaces, so that the subspaces were approximately statistically independent of each other.

The key step in solving this problem is to constrain the minimisation of  $D$  in such a way as to encourage the formation of code schemes in which each component of the code vector  $\mathbf{y}$  codes a different subspace of the input vector  $\mathbf{x}$ . There are two related constraints that may be imposed on  $\Pr(\mathbf{y}|\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  which may be summarised as

$$\begin{aligned} \Pr(\mathbf{y}|\mathbf{x}) &= \Pr(y_1|\mathbf{x}) \Pr(y_2|\mathbf{x}) \cdots \Pr(y_n|\mathbf{x}) \\ \mathbf{x}'(\mathbf{y}) &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}'(y_i) \end{aligned} \quad (2.3)$$

Thus each component  $y_i$  (for  $i = 1, 2, \dots, n$  and  $1 \leq y_i \leq M$ ) is an *independent* sample drawn from the codebook using  $\Pr(y_i|\mathbf{x})$  (which is assumed to be the same function for all  $i$ ), and the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  (vector argument) is assumed to be a *superposition* of  $n$  contributions  $\mathbf{x}'(y_i)$  (scalar argument) for  $i = 1, 2, \dots, n$ . Taken together, these constraints encourage the forma-

tion of coding schemes in which independent subspaces are separately coded, as required.

The constraints in Equation 2.3 prevent the full space of possible values of  $\Pr(y|\mathbf{x})$  or  $\mathbf{x}'(y)$  from being explored as  $D$  is minimised, so they lead to an *upper bound*  $D_1 + D_2$  on the FMC objective function  $D$  (i.e.  $D \leq D_1 + D_2$ ), which may be derived as [8]

$$\begin{aligned} D_1 &\equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \\ D_2 &\equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2 \end{aligned} \quad (2.4)$$

Note that  $M$  (size of codebook) and  $n$  (number of samples drawn from codebook using  $\Pr(y|\mathbf{x})$ ) are effectively model order parameters, whose values need to be chosen appropriately for each encoder optimisation problem. The properties of the optimum solution depend critically on the interplay between the statistical properties of the training data and the model order parameters  $M$  and  $n$ , as will be seen in the simulations in Section III.

In numerical simulations it is convenient to parameterise (i.e. constrain)  $\Pr(y|\mathbf{x})$  thus

$$\begin{aligned} \Pr(y|\mathbf{x}) &= \frac{Q(y|\mathbf{x})}{\sum_{y'=1}^M Q(y'|\mathbf{x})} \\ Q(y|\mathbf{x}) &= \frac{1}{1 + \exp(-\mathbf{w}(y) \cdot \mathbf{x} - b(y))} \end{aligned} \quad (2.5)$$

where  $Q(y|\mathbf{x})$  is a sigmoid function of  $\mathbf{x}$ , having weight vector  $\mathbf{w}(y)$  and bias  $b(y)$ . This form for  $\Pr(y|\mathbf{x})$  may be further constrained to obtain encoders that have particular desired properties. For instance, a very simple and useful constraint is to impose a threshold constraint on the sigmoid function, which forces the value of the sigmoid to lie exactly halfway up its slope when the norm of its input vector is  $\theta$ . This is achieved by choosing  $b(y) = -\theta |\mathbf{w}(y)|$ , so that

$$Q(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\hat{\mathbf{w}}(y) \cdot \mathbf{x} - \theta) \|\mathbf{w}(y)\|)} \quad (2.6)$$

where  $\|\mathbf{w}(y)\| \equiv \sqrt{\mathbf{w}(y) \cdot \mathbf{w}(y)}$  and  $\hat{\mathbf{w}}(y) \equiv \frac{\mathbf{w}(y)}{\|\mathbf{w}(y)\|}$ . This type of constraint may be used to force the optimum SVQ to ignore the “small” degrees of freedom in the input [10], and thus exhibit invariance with respect to these degrees of freedom.

### III. STEREO DISPARITY SIMULATIONS

The simulations described here were originally reported in much greater detail in [9]. The problem is to encode a pair of 1-dimensional images of a target which derive from the two sensors of a stereoscopic imaging system. The location of the target on a sensor is specified

by a single intrinsic coordinate, and the pair of such coordinates (one for each of the two sensors) are correlated with each other, because the target appears in similar positions on each sensor in the stereoscopic imaging system. Also, each image pixel independently has a large amount of multiplicative noise and a small amount of additive noise.

Each 1-dimensional image has 18 pixels, so the total number of pixels is 36, the code book has 24 entries ( $M = 24$ ), two separate runs are done sampling either 2 or 50 code indices for each input vector ( $n = 2$  or  $n = 50$ ), the update step size is 0.1 ( $\varepsilon = 0.1$ ), the Gaussian target half-width is 2, the half-width of the uniform distribution of stereo disparities is 4 pixels ( $a = 4$ ), the background noise level is 0.2 ( $\sigma = 0.2$ ), the elements of the weight vectors  $\mathbf{w}(y)$  and the biasses  $b(y)$  used in the  $Q(y|\mathbf{x})$  (see equation 2.5), and the reconstruction vectors  $\mathbf{x}'(y)$  are initialised to uniformly distributed random numbers in the interval  $[-0.05, +0.05]$ .

The steps needed to generate a stereo pair of images are:

1. Locate a Gaussian target on one of the sensors and generate a randomly shifted copy (the shift is uniformly distributed in  $[-a, a]$ ) on the other sensor.
2. Rotate the stereo image of the target to a randomised position on the pair of sensors (using circular wraparound).
3. Multiply each pixel value by a random number uniformly distributed in  $[0, 1]$  (this is *multiplicative noise*).
4. Add an independent uniformly distributed random number in the range  $[0, \sigma]$  to each pixel value (this is *additive noise*).

Some examples of stereo pairs thus generated are shown in Figure 2.

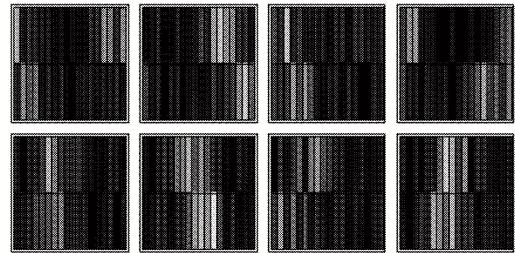


Figure 2: 8 typical stereo pairs of noisy 1-dimensional images.

Very good convergence to a local minimum of  $D_1 + D_2$  was achieved by 2000 iterations of a simple gradient descent update prescription (i.e. based on  $\frac{\partial(D_1+D_2)}{\partial \text{parameters}}$ ), although good convergence occurs after only a few hundred such updates.

The posterior probabilities that each code index is selected as a function of the mean position of the two images (horizontal axis) and stereo disparity (vertical axis)

of a test target are shown in Figure 3 (for  $n = 50$ ) and Figure 4 (for  $n = 2$ ).

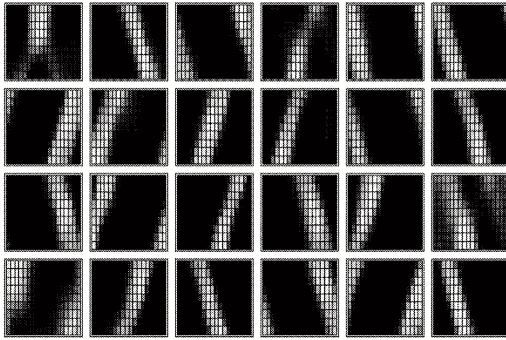


Figure 3: Typical posterior probabilities obtained using  $n = 50$  and  $M = 24$ .

The results shown in Figure 3 show that if  $n = 50$  (i.e. large) then the code book operates as a factorial encoder, because each code index encodes only one of the stereo pair of images. However there are 2 subsets of code indices, one of which has a negative slope and the other of which has a positive slope, each of which encodes a different image from the stereo pair. The intersections between these two subsets may be used to define localised resolution cells. Thus the measurement of stereo disparity requires that a minimum of two code indices be observed, which must belong to oppositely sloping subsets in Figure 3. In practice, many more than two code indices must be observed to virtually guarantee that there is at least one in each of the two subsets, which is why the large value  $n = 50$  was used in this simulation. Because the images in the stereo pair are separately encoded, this type of encoder has not discovered a way of fusing the information in the stereo pair of images.

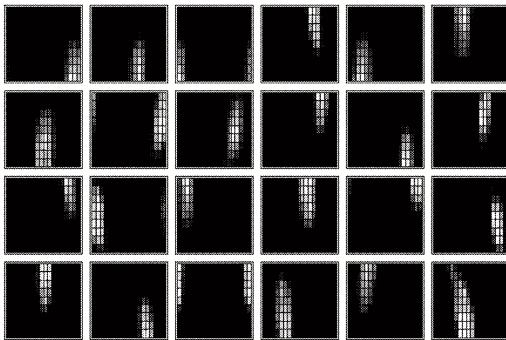


Figure 4: Typical posterior probabilities obtained using  $n = 2$  and  $M = 24$ .

The results shown in Figure 4 show that if  $n = 2$  (i.e. small) then the code book operates as a joint encoder, because each code index jointly encodes position and stereo disparity. The disparity direction is resolved into approximately 3 disparities (positive, zero, and negative disparity), whereas the position direction is resolved into approximately 8 positions, giving a total of 24 ( $= 3 * 8$ ) different possible codes. The measurement of stereo disparity (and position) to this resolution requires that only one code index is observed. Because the images in the stereo pair are jointly encoded, this type of encoder has adaptively discovered a way of fusing the information in the stereo pair of images.

It is possible to further improve the results in Figure 4 for the extraction of stereo disparity. Thus if the constrained version of  $Q(y|\mathbf{x})$  (see equation 2.6) is used during training, then the threshold parameter  $\theta$  can be adjusted so that the (assumed) “small” stereo disparity degree of freedom is ignored by the optimum SVQ, so that the results shown in Figure 4 would be invariant with respect to stereo disparity (i.e. vertical stripes rather than blobs). This type of invariant encoder could be used to automatically extract stereo disparity, by computing the residual obtained when the original stereo pair of images is subtracted from their reconstruction obtained from the invariant SVQ.

#### IV. CONCLUSIONS

The theory presented in this paper shows how a standard vector quantiser (VQ) [6] may be generalised to a stochastic vector quantiser (SVQ) [7, 8], which may be used to automate the process of discovering structure in data, such as the presence of recurring patterns or correlations across multiple sensors. This is illustrated by training an SVQ on stereo pairs of images, to demonstrate how various types of encoder may be obtained, and how the stereo disparity degree of freedom may be extracted, for instance.

#### V. ACKNOWLEDGEMENT

This work was carried out as part of Technology Group 10 of the MoD Corporate Research Programme.

[1] N Farvardin, *A study of vector quantisation for noisy channels*, IEEE Transactions on Information Theory **36**

(1990), no. 4, 799–809.  
[2] A Gersho and R M Gray, *Vector quantisation and signal*

- compression*, Kluwer, Norwell, 1992.
- [3] R M Gray, *Vector quantisation*, IEEE Transactions on Acoustics, Speech, and Signal Processing Magazine **1** (1984), no. 2, 4–29.
  - [4] T Kohonen, *Self-organisation and associative memory*, Springer-Verlag, Berlin, 1984.
  - [5] H Kumazawa, M Kasahara, and T Namekawa, *A construction of vector quantisers for noisy channels*, Electronics and Engineering in Japan **67** (1984), no. 4, 39–47.
  - [6] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
  - [7] S P Luttrell, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
  - [8] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
  - [9] ———, *A user's guide to stochastic encoder/decoders*, Technical Report DERA/S&P/SPI/TR990290, Defence Evaluation and Research Agency, Malvern, 1999.
  - [10] ———, *Invariance discovery by vector quantisation of noisy data*, Technical Report DERA/S&P/SPI/TR000070, Defence Evaluation and Research Agency, Malvern, 2000.



# A Self-Organising Approach to Multiple Classifier Fusion \*

S P Luttrell<sup>†</sup>  
DERA, Malvern, Worcestershire, U.K.

In this paper the theory of unsupervised multi-layer stochastic vector quantiser (SVQ) networks is reviewed, and then extended to the supervised case where the network is to be used as a classifier. This leads to a hybrid approach, in which training is governed both by unsupervised and supervised pieces in the network objective function. The unsupervised piece aims to preserve enough information in the network to be able to accurately reconstruct the input (i.e. the network serves as an encoder), whereas the supervised piece aims to reproduce the classification output supplied by an external teacher (i.e. the network serves as a classifier). The tension between these two pieces of the objective function leads to an optimal network, in which typically the lower layers (near to the input) act as faithful encoders of the input, whereas the higher layers (near to the output) act as faithful classifiers. The results of some simulations are presented to illustrate these properties.

## I. INTRODUCTION

For a review of the subject of combining classifiers see the introduction to [2], where it is stated that the two main reasons for combining classifiers are efficiency and accuracy. Efficiency gains may be obtained when the classifier is implemented using a network of simple processing operations, and accuracy may be improved when results from two or more classifiers (with different strengths and weaknesses) are combined [1]. Typically, both of these strategies are simultaneously employed. Thus a classifier ensemble is implemented, where each classifier has a different set of simple operations, so each has its own peculiar strengths and weaknesses. Then the classifiers in this ensemble are combined to produce the overall classifier.

The question that will be addressed in this paper is how to simultaneously solve the two problems of designing the separate classifiers and combining them together to produce the overall classifier. A novel approach will be used, in which the overall classifier will be allowed to emerge by a process of self-organisation, that is driven by the minimisation of a suitably chosen objective function. This approach implements the overall classifier as a multi-layer network with full interconnections between adjacent layers. However, self-organisation typically discovers optimal solutions in which the connections implement a set of simple processing operations using only a subset of the connections, followed by combination of their outputs to produce the overall classifier.

The basic unit of computation in the self-organising network is a generalisation of the standard vector quantiser (VQ) [3], called a stochastic vector quantiser (SVQ) [5, 7], in which samples are drawn probabilistically from a codebook. The objective function used to optimise a

SVQ is the mean Euclidean error that occurs when using the SVQ to encode an input as multiple probabilistic samples, followed by reconstruction from these samples to estimate the input. If multiple samples are allowed, then an SVQ can use much cleverer coding schemes than a standard VQ. For instance, self-organisation can cause the codebook to split into several smaller codebooks, each of which specialises in encoding only part of the input. This propensity for the codebook to split is the key to using self-organisation to form a classifier ensemble, in which the different classifiers have different strengths and weaknesses.

Thus far the SVQ objective function is unsupervised, because it makes no provision for an external teacher to influence the way in which the SVQ encodes its input. This is easily rectified by adding a term to the SVQ objective function that attempts to steer the SVQ output towards some desired target output. This external supervision is readily put to use in designing a classifier network, where it may be used to force the final output of the network to be the required overall classifier.

In Section II the underlying theory of SVQs is presented, and in Section III the results of simulations are presented to demonstrate the potential use of SVQs to classification.

## II. THEORY

In this section various pieces of the previously published theory of stochastic vector quantisers (SVQ) are unified to establish a coherent framework for modelling SVQs. In Section II A the basic theory of SVQs is given (which is equivalent to the theory of FMCs reported in [4]), and in Section II B it is extended to the case of high-dimensional input data [5]. In Section II C the theory is further generalised to chains of linked SVQs [6], and the use of an external teacher to supervise the chain is explained in Section II D.

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

This appeared in Proceedings of the 2nd International Workshop on Multiple Classifier Systems, Cambridge, 2-4 July 2001, pp 319-328. © British Crown Copyright 2001.

<sup>†</sup>Electronic address: luttrell@signal.dera.gov.uk

### A. Stochastic Vector Quantisers

The basic building block of the encoder/decoder model used in this paper is the folded Markov chain (FMC) [4], which is equivalent to the SVQ discussed in Section I. Thus an input vector  $\mathbf{x}$  is encoded as a code index vector  $\mathbf{y}$ , which is then subsequently decoded as a reconstruction  $\mathbf{x}'$  of the input vector. Both the encoding and decoding operations are allowed to be probabilistic, in the sense that  $\mathbf{y}$  is a sample drawn from  $\Pr(\mathbf{y}|\mathbf{x})$ , and  $\mathbf{x}'$  is a sample drawn from  $\Pr(\mathbf{x}'|\mathbf{y})$ , where  $\Pr(\mathbf{y}|\mathbf{x})$  and  $\Pr(\mathbf{x}'|\mathbf{y})$  are Bayes' inverses of each other, as given by  $\Pr(\mathbf{x}'|\mathbf{y}) = \frac{\Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x})}{\int d\mathbf{x}'' \Pr(\mathbf{y}|\mathbf{x}'') \Pr(\mathbf{x}'')}$ , and  $\Pr(\mathbf{x})$  is the prior

probability from which  $\mathbf{x}$  was sampled. Because the chain of dependences in passing from  $\mathbf{x}$  to  $\mathbf{y}$  and then to  $\mathbf{x}'$  is first order Markov (i.e. it is described by the directed graph  $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}'$ , and because the two ends of this Markov chain (i.e.  $\mathbf{x}$  and  $\mathbf{x}'$ ) live in the same vector space, it is called a *folded* Markov chain [4].

In order to ensure that the SVQ encodes the input vector optimally, a measure of the reconstruction error must be minimised. There are many possible ways to define this measure, but one that is consistent with many previous results, and which also leads to many new results, is the mean Euclidean reconstruction error measure  $D$ , which is defined as

$$D \equiv \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1=1}^M \sum_{y_2=1}^M \cdots \sum_{y_n=1}^M \Pr(\mathbf{y}|\mathbf{x}) \int d\mathbf{x}' \Pr(\mathbf{x}'|\mathbf{y}) \|\mathbf{x} - \mathbf{x}'\|^2 \quad (2.1)$$

where  $\mathbf{y} = (y_1, y_2, \dots, y_n)$ ,  $1 \leq y_i \leq M$  is assumed,  $\Pr(\mathbf{x}) \Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x}'|\mathbf{y})$  is the joint probability that the SVQ has state  $(\mathbf{x}, \mathbf{y}, \mathbf{x}')$ ,  $\|\mathbf{x} - \mathbf{x}'\|^2$  is the Euclidean reconstruction error, and  $\int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1=1}^M \sum_{y_2=1}^M \cdots \sum_{y_n=1}^M \int d\mathbf{x}' (\dots)$  sums over all possible states of the SVQ (weighted by the joint probability).

The Bayes' inverse probability  $\Pr(\mathbf{x}'|\mathbf{y})$  may be integrated out of this expression for  $D$  to yield

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y_1=1}^M \sum_{y_2=1}^M \cdots \sum_{y_n=1}^M \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (2.2)$$

where the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  is defined as  $\mathbf{x}'(\mathbf{y}) = \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ . Because of the quadratic form of the objective function, it turns out that  $\mathbf{x}'(\mathbf{y})$  may be treated as a free parameter, whose optimum value (i.e. the solution of  $\frac{\partial D}{\partial \mathbf{x}'(\mathbf{y})} = 0$ ) is  $\int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ , as required.

### B. High Dimensional Input Spaces

A problem with the standard VQ is that its code book grows exponentially in size as the dimensionality of the input vector is increased, assuming that the contribution to the reconstruction error from each input dimension is held constant. This means that such VQs are useless for encoding extremely high dimensional input vectors, such as images. The usual solution to this problem is to manually partition the input space into a number of lower dimensional subspaces, and then to encode each of these subspaces separately. However, it would be very useful if this partitioning could be done automatically, in such a way that typically the correlations *within* each sub-

space were much stronger than the correlations *between* subspaces, so that the subspaces were approximately statistically independent of each other. This is an example of the self-organised discovery of a classifier ensemble, in which each classifier focusses on only a subspace of the input.

The key step in solving this problem is to constrain the minimisation of  $D$  in such a way as to encourage the formation of code schemes in which each component of the code vector  $\mathbf{y}$  codes a different subspace of the input vector  $\mathbf{x}$ . There are two related constraints that may be imposed on  $\Pr(\mathbf{y}|\mathbf{x})$  and  $\mathbf{x}'(\mathbf{y})$  which may be summarised as

$$\begin{aligned} \Pr(\mathbf{y}|\mathbf{x}) &= \Pr(y_1|\mathbf{x}) \Pr(y_2|\mathbf{x}) \cdots \Pr(y_n|\mathbf{x}) \\ \mathbf{x}'(\mathbf{y}) &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}'(y_i) \end{aligned} \quad (2.3)$$

Thus, for  $i = 1, 2, \dots, n$  and  $1 \leq y_i \leq M$ , each component  $y_i$  is an *independent* sample drawn from the codebook using  $\Pr(y_i|\mathbf{x})$  (which is assumed to be the same function for all  $i$ ), and the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  (vector argument) is assumed to be a *superposition* of  $n$  contributions  $\mathbf{x}'(y_i)$  (scalar argument). Taken together, these constraints encourage the formation of coding schemes in which independent subspaces are separately coded, as required.

The constraints in Equation 2.3 prevent the full space of possible values of  $\Pr(\mathbf{y}|\mathbf{x})$  or  $\mathbf{x}'(\mathbf{y})$  from being explored as  $D$  is minimised, so they lead to an upper bound  $D_1 + D_2$  on the SVQ objective function  $D$  (i.e.  $D \leq D_1 + D_2$ ), which may be derived as (the details of this derivation,

including the derivatives of  $D_1 + D_2$ , are reported in [5])

$$D_1 \equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \quad (2.4)$$

$$D_2 \equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2$$

Note that  $M$  (size of codebook) and  $n$  (number of samples drawn from codebook using  $\Pr(y|\mathbf{x})$ ) are effectively model order parameters, whose values need to be chosen appropriately for each encoder optimisation problem. The properties of the optimum encoder depend critically on the interplay between the statistical properties of the training data and the model order parameters  $M$  and  $n$ .

In numerical simulations it is convenient to parameterise (i.e. constrain)  $\Pr(y|\mathbf{x})$  thus

$$\begin{aligned} \Pr(y|\mathbf{x}) &= \frac{Q(y|\mathbf{x})}{\sum_{y'=1}^M Q(y'|\mathbf{x})} \\ Q(y|\mathbf{x}) &\equiv \frac{1}{1 + \exp(-\mathbf{w}(y) \cdot \mathbf{x} - b(y))} \end{aligned} \quad (2.5)$$

where  $Q(y|\mathbf{x})$  is a sigmoid function of  $\mathbf{x}$ , with weight vector  $\mathbf{w}(y)$  and bias  $b(y)$ .

### C. Chain of Linked Stochastic Vector Quantisers

An SVQ may be generalised to a chain of linked SVQs, and further generalisation to any acyclically linked network of SVQs is also readily achieved. The vector of probabilities (for all values of the code index) computed by each stage in the chain is used as the input vector to the next stage, and the overall objective function is a weighted sum of the SVQ objective functions derived from each stage. There are other ways of linking the stages together and defining an overall objective function, but the above prescription is the simplest possibility. The total number of free parameters in an  $L$  stage chain is  $3L - 1$ , which is the sum of 2 free parameters for each of the  $L$  stages, plus  $L - 1$  weighting coefficients. There are  $L - 1$  rather than  $L$  weighting coefficients because the overall normalisation of the objective function does not affect the optimum solution.

The chain of linked SVQs will now be expressed mathematically. Firstly, an index  $l$  (where  $1 \leq l \leq L$ ) is introduced to allow different stages of the chain to be

distinguished thus

$$\begin{aligned} M &\longrightarrow M^{(l)} \\ n &\longrightarrow n^{(l)} \\ \mathbf{x} &\longrightarrow \mathbf{x}^{(l)} \\ \mathbf{x}' &\longrightarrow \mathbf{x}'^{(l)} \\ y &\longrightarrow y^{(l)} \\ D &\longrightarrow D^{(l)} \\ D_1 &\longrightarrow D_1^{(l)} \\ D_2 &\longrightarrow D_2^{(l)} \end{aligned} \quad (2.6)$$

Then the stages are then defined and linked together thus

$$\begin{aligned} \mathbf{x}^{(l)} &\longrightarrow y^{(l)} \longrightarrow \mathbf{x}'^{(l)} \\ \mathbf{x}^{(l+1)} &= (x_1^{(l+1)}, x_2^{(l+1)}, \dots, x_{M^{(l)}}^{(l+1)}) \\ x_i^{(l+1)} &= \Pr(y^{(l)} = i|\mathbf{x}^{(l)}), \quad 1 \leq i \leq M^{(l)} \end{aligned} \quad (2.7)$$

Finally, the objective function and its upper bound are given by

$$\begin{aligned} D &= \sum_{l=1}^L s^{(l)} D^{(l)} \\ &\leq D_1 + D_2 \\ &= \sum_{l=1}^L s^{(l)} (D_1^{(l)} + D_2^{(l)}) \end{aligned} \quad (2.8)$$

where  $s^{(l)} \geq 0$  is the weighting that is applied to the contribution  $D_1^{(l)} + D_2^{(l)}$  of stage  $l$  of the chain to the overall objective function  $D$ .

### D. Supervision by an External Teacher

The objective function  $D$  in Equation 2.8 can readily be extended to allow an external teacher to supervise the chain of linked SVQs. Thus make the following replacement in Equation 2.8

$$D_1^{(l)} + D_2^{(l)} \longrightarrow D_1^{(l)} + D_2^{(l)} + D_{\text{supervise}}^{(l)} \quad (2.9)$$

where  $D_{\text{supervise}}^{(l)}$  is any convenient objective function that the external teacher wishes to apply to stage  $l$  of the chain. For instance  $D_{\text{supervise}}^{(l)}$  might measure the Euclidean error between the output of stage  $l$  (whose components are  $\Pr(y^{(l)} = i|\mathbf{x}^{(l)})$ ,  $1 \leq i \leq M^{(l)}$ ) and some externally supplied reference vector.

Note that supervision can be applied to any or all of the stages, and is not limited to only the final stage, as is conventional in supervised training. In the context of combining classifiers, not only the design of the combined classifier, but also the design of the individual classifiers may be supervised.

### III. SIMULATIONS

In this section the results of several simulations are presented to illustrate the behaviour of SVQs in both unsupervised and supervised training scenarios. Circular (i.e.  $S^1$ ) and 2-toroidal (i.e.  $S^1 \times S^1$ ) input manifolds are used.

#### A. Circular Input Manifold

The simplest demonstration of a SVQ is to train it (unsupervised) with data that lives on a circular manifold.

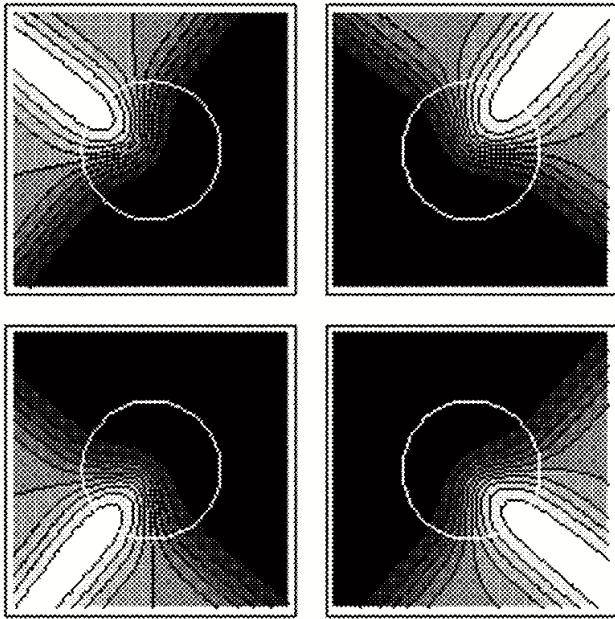


Figure 1: Contours of  $\Pr(y|\mathbf{x})$  for data lying on a circle (represented by the white circle in each plot), trained using  $M = 4$  and  $n = 10$ .

Figure 1 shows contour plots of the posterior probabilities  $\Pr(y|\mathbf{x})$  for  $y = 1, 2, 3, 4$  in a SVQ with  $M = 4$  and  $n = 10$ . The circular manifold is chopped up by the  $\Pr(y|\mathbf{x})$  into four softly overlapping arcs.

#### B. 2-Toroidal Input Manifold

A more sophisticated demonstration of a SVQ is to train it (unsupervised) with data that lives on a toroidal manifold.

In Figure 2(a) each of the  $\Pr(y|\mathbf{x})$  for  $M = 8$  and  $n = 5$  has a localised response region on the 2-torus, and the 2-torus is thus chopped up into eight softly overlapping regions. This result (2-dimensional manifold) may be compared with the simpler result (1-dimensional manifold) in Figure 1. In both cases, sampling a single code

### A Self-Organising Approach to Multiple Classifier Fusion

index from the code book is sufficient to determine the location of the input vector to an accuracy corresponding to a single localised response region. A major disadvantage of this type of encoder is that the size of the code book that is required to guarantee a given resolution (in each dimension of the input manifold) increases *exponentially* with input manifold dimensionality. This would be completely useless for very high dimensional applications, such as image processing. This general type of encoder will be called a *joint* encoder, because it simultaneously encodes all of the dimensions of the input manifold.

In Figure 2(b) the results shown are analogous to those shown in Figure 2(a), except that  $M = 8$  and  $n = 50$ , so 10 times as many samples are now drawn from the code book. There is a marked difference in the shape of the response region of each of the  $\Pr(y|\mathbf{x})$ . The set of  $\Pr(y|\mathbf{x})$  for  $y = 1, 2, \dots, 8$  has split into two subsets. In one subset the response regions are elongated vertically, and in the other subset they are elongated horizontally. In all cases there is approximate invariance of  $\Pr(y|\mathbf{x})$  with respect to variations of  $\mathbf{x}$  along the direction of elongation of the response region. In effect, one subset encodes one of the circular  $S^1$  subspaces of the 2-torus  $S^1 \times S^1$ , and the other subset encodes the other circular  $S^1$  subspace. In this type of encoder it is necessary to sample many times from the code book in order to guarantee that at least one sample is drawn from each of these subsets, so that both of the circular subspaces are represented in the code. In this case the location of the input vector may be determined to an accuracy corresponding to the region of intersection of an orthogonal pair of elongated response regions.

A major advantage of this type of encoder is that the size of the code book that is required to guarantee a given resolution (in each dimension of the input manifold) increases *linearly* with input manifold dimensionality, and the price that has to be paid for this is the need to sample many times from the code book. This general type of encoder will be called a factorial encoder, because it separately encodes each of the dimensions of the input manifold. This propensity for the codebook to split into a number of smaller code books is the key to using self-organisation to form a classifier ensemble, in which the different classifiers have different strengths and weaknesses.

#### C. 2-Toroidal Input Manifold with Supervision

A yet more sophisticated demonstration of an SVQ is to train it with 2-toroidal data (see Section III B), but this time introduce some supervision by an external teacher.

To allow a reasonable amount of flexibility a 2-stage encoder will be used (see Section II C), where zero weight will be assigned to the  $D_1^{(2)} + D_2^{(2)}$  contribution to the overall objective function  $D$ , so that the second stage is devoted entirely to dealing with the supervision that

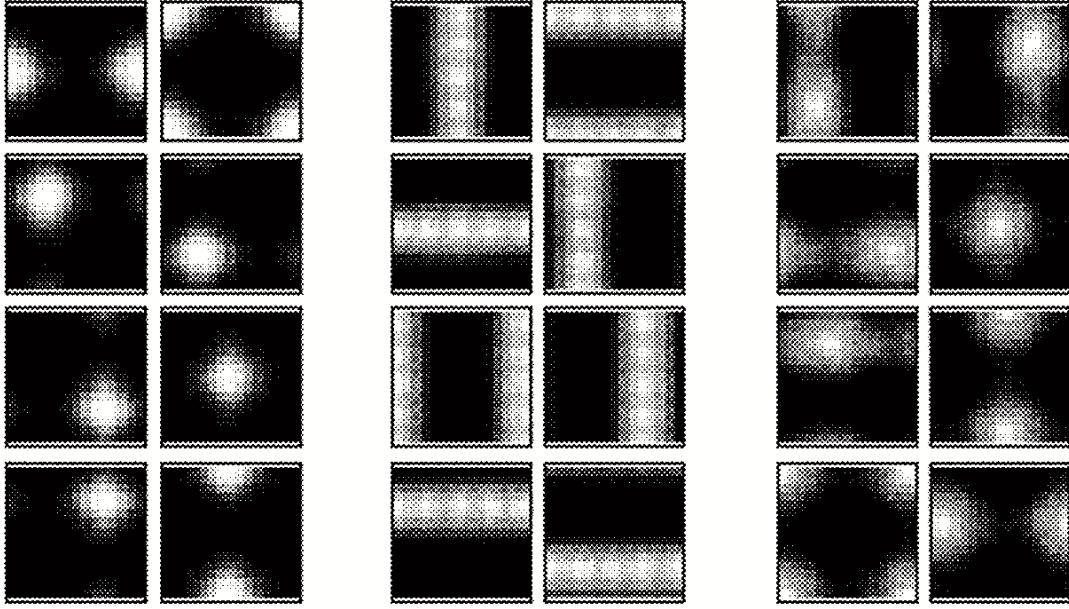


Figure 2: All of these plots use toroidal boundary conditions. (a) (left)  $\Pr(y|\mathbf{x})$  for data lying on a 2-torus, trained using  $M = 8$  and  $n = 5$ . (b) (middle)  $\Pr(y|\mathbf{x})$  for data lying on a 2-torus, trained using  $M = 8$  and  $n = 50$ . (c) (right)  $\Pr(y|\mathbf{x})$  for data lying on a 2-torus, trained using  $M = (8, 2)$  and  $n = (50, \text{N/A})$ , and using *strong* supervision.

is introduced via the  $D_{\text{supervise}}^{(l)}$  contribution. Back-propagation of derivatives of the objective function will ensure that this supervision also influences the first stage of the 2-stage encoder.

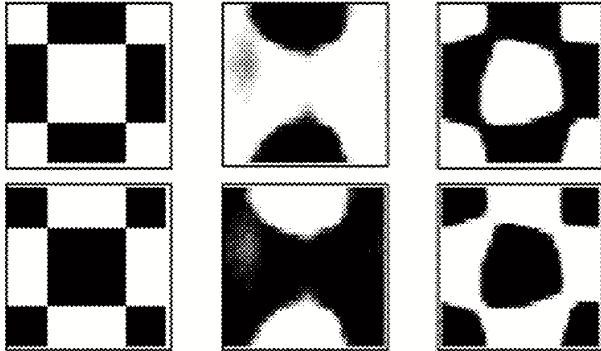


Figure 3: All of these plots use toroidal boundary conditions. (a) (left) Target outputs used for supervision of a 2-stage encoder. (b) (middle) Output produced when trained using *weak* supervision. (c) (right) Output produced when trained using *strong* supervision.

The first and second stages will use code books with parameters ( $M = 8, n = 50$ ) and ( $M = 8, n = \text{N/A}$ ), respectively, and the external teacher will attempt to make the pair of outputs from the second stage equal to the pair of target outputs shown in Figure 3(a). These target outputs form oppositely signed checkerboard pattern (of 0's and 1's) on the 2-toroidal input manifold, so that

they have the properties of a posterior probability (i.e. non-negative and sum to unity at each point on the 2-torus). This particular form of target output has been chosen to be more difficult to produce using a factorial encoder (see Figure 2(b)) than a joint encoder (see Figure 2(a)), because the latter uses response regions that are similar in shape to the squares in the checkerboard pattern used by the external teacher (see Figure 3(a)).

When the 2-stage encoder is trained using supervision that is weak enough not to significantly influence the first stage, the results are the same as those shown in Figure 2(b). Figure 3(b) then shows the output of the second stage, which should be compared with the required output shown in Figure 3(a). The results in Figure 3(b) are poor because the weak supervision signal cannot produce large enough back-propagated derivatives to override the unsupervised training of the first stage, which insists on producing a factorial code in which the two circular subspaces of the 2-toroidal input manifold are separately encoded (see Figure 2(b)). In effect, these results are produced by combining the outputs of two badly designed classifiers, each of which concentrates on only one of the  $S^1$  subspaces of the 2-torus.

When the supervision is strong enough to significantly influence the first stage, the results are as shown in Figure 2(c) and Figure 3(c) (which are analogous to Figure 2(b) and Figure 3(b), respectively). Figure 2(c) shows that the factorial encoder that arose when weak supervision was used (see Figure 2(b)) has now been modified by the use of strong supervision (via back-propagation of the correspondingly large derivatives) to resemble a joint

encoder (compare Figure 2(a)). This is as expected, because the response regions of a joint encoder resemble the shape of the squares in the checkerboard pattern used by the supervisor (see Figure 3(a)). Figure 3(c) shows the output of the second stage, which now much more closely resembles the required output (see Figure 3(a)) than when the supervision was weak (see Figure 3(b)). In effect, these results are produced by a single classifier that makes optimal use of the  $S^1 \times S^1$  2-torus, rather than concentrating on only one of its  $S^1$  subspaces at a time, as was the case in Figure 3(b).

These results obtained from a 2-stage network of linked SVQs are illustrative of the more general possibilities offered by acyclically linked networks of SVQs.

#### IV. DISCUSSION

A key behaviour of 2-stage encoders is exemplified by the results obtained from a 2-toroidal input manifold with supervision (see Section III C).

When only weak supervision is used, the first stage optimises itself to encode the input so that it can reconstruct it with minimum distortion (i.e. minimise  $D_1 + D_2$  in Equation 2.4). There is no guarantee that this encoder will be any good for accurately producing the output required by the weak supervision. Effectively, the first stage preprocesses the input in a way that is heedless of the nature of the task that the second stage has to do. This is analogous to the situation where an overall classifier is constructed by combining the outputs of a set of individual classifiers, that are designed in ignorance of the overall classification problem that is to be solved, so it does not perform very well.

When strong supervision is used, the first stage is forced to heed the requirements of the second stage, and adjusts the way in which it preprocesses the input, so that the second stage is able to accurately produce the output required by the strong supervision. This is analogous to

#### A Self-Organising Approach to Multiple Classifier Fusion

the situation where an overall classifier is constructed by combining the outputs of a set of individual classifiers, that are designed in way that is mindful of the overall classification problem that is to be solved, so it performs very well.

There are many ways in which the results of Section III C can be generalised. The multi-stage encoder could have more than two stages, and in general it could be an acyclically linked network of encoders having multiple input and multiple output stages. The basic properties of each encoder are specified by three parameters: size of the code book  $N$ , number of samples  $n$  drawn from the code book, and objective function weighting  $s$ . Supervision by an external teacher could be added to any or all of the encoder outputs.

#### V. CONCLUSIONS

In this paper SVQs have been shown to be a flexible tool for self-organised classifier fusion. The unsupervised part of the network objective function tries to preserve information about the input data as it is processed and passed through the network. The supervised part of the objective function tries to ensure that the required classifier output is produced at the network output, by judiciously discarding irrelevant information and massaging what is left into the required output form. The tension between these two pieces of the objective function leads to an optimal network, in which typically the lower layers (near to the input) act as faithful encoders of the input, whereas the higher layers (near to the output) act as faithful classifiers.

#### VI. ACKNOWLEDGEMENT

This work was carried out as part of Technology Group 10 of the MoD Corporate Research Programme.

- 
- [1] K M Ali and M J Pazzani, *On the link between error correlation and error reduction in decision tree ensembles*, Tech. Report 95-38, ICS-UCI, California University, Irvine, 1985.
  - [2] J Kittler, M Hatef, R P Duin, and J Matas, *On combining classifiers*, IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998), no. 3, 226–239.
  - [3] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
  - [4] S P Luttrell, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
  - [5] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
  - [6] ———, *An adaptive network for encoding data using piecewise linear functions*, Proceedings of international conference on artificial neural networks (Edinburgh), IEE, 1999, pp. 198–203.
  - [7] ———, *A user’s guide to stochastic encoder/decoders*, Technical Report DERA/S&P/SPI/TR990290, Defence Evaluation and Research Agency, Malvern, 1999.

# Using Stochastic Vector Quantisers to Characterise Signal and Noise Subspaces \*

S. P. Luttrell

Signal and Information Processing Department, Defence Evaluation  
and Research Agency, St. Andrews Rd, Malvern WR14 3PS, UK

**Abstract:** In this paper a stochastic generalisation of the standard Linde-Buzo-Gray (LBG) approach to vector quantiser (VQ) design is presented, in which the encoder is implemented as the sampling of a vector of code indices from a probability distribution derived from the input vector, and the decoder is implemented as a superposition of reconstruction vectors. This stochastic VQ (SVQ) is optimised using a minimum mean Euclidean reconstruction distortion criterion, as in the LBG case. Numerical simulations are used to demonstrate how this leads to self-organisation of the SVQ, where different stochastically sampled code indices become associated with different input subspaces.

## I. INTRODUCTION

In vector quantisation a code book is used to encode each input vector as a corresponding code index, which is then decoded (again, using the codebook) to produce an approximate reconstruction of the original input vector [3, 4]. The purpose of this paper is to generalise the standard approach to vector quantiser (VQ) design [5], so that each input vector is encoded as a *vector* of code indices that are stochastically sampled from a probability distribution that depends on the input vector, rather than as a *single* code index that is the deterministic outcome of finding which entry in a code book is closest to the input vector. This will be called a stochastic VQ (SVQ), and it includes the standard VQ as a special case.

One advantage of using the stochastic approach is that it automates the process of splitting high-dimensional input vectors into low-dimensional blocks before encoding them, because minimising the mean Euclidean reconstruction error can encourage different stochastically sampled code indices to become associated with different input subspaces [8, 10]. Another advantage is that it is very easy to connect SVQs together, by using the vector of code index probabilities computed by one SVQ as the input vector to another SVQ [9].

SVQ theory will be extended to the case of encoding noisy (or distorted) data, with the intention of subsequently reconstructing an approximation to the noiseless data. This theory is then applied to the problem of encoding data vectors which are a superposition of a “large” jammer and a “small” signal, where the signal is regarded as a distortion superimposed on the jammer, rather than the other way around. The reconstruction is then an approximation to the jammer, which can thus be subtracted from the original data to reveal the underlying signal of interest.

In Section II the underlying theory of SVQs is devel-

oped together with its extension to the encoding of noisy data, and in Section III some simulations illustrating the application of SVQs to the nulling of jammers are presented.

## II. THEORY

In Section II A the basic theory of SVQs is summarised, in Section II B SVQ theory is extended to the case of encoding noisy or distorted data with the intention of eventually recovering the undistorted data, in Section II C this extended theory is applied to the problem of encoding data that contain unwanted “nuisance degrees of freedom”, in Section II D some constraints on the optimisation of the encoder are introduced to encourage the encoder to disregard the nuisance degrees of freedom (i.e. discover invariances), and finally in Section II E this invariant encoder theory is applied to the problem of encoding and subsequently nulling “large” jammers that obscure “small” signals.

### A. Stochastic Vector Quantisers

The basic building block of the encoder/decoder model used in this paper is the folded Markov chain (FMC) [6], which is equivalent to the SVQ discussed in Section I, and analysed in detail in [7]. The operations that occur in a SVQ are summarised in Figure 1(a).

Thus an input vector  $\mathbf{x}$  is encoded as a code index vector  $\mathbf{y}$ , which is then subsequently decoded as a reconstruction  $\mathbf{x}'$  of the input vector. Both the encoding and decoding operations are allowed to be probabilistic, in the sense that  $\mathbf{y}$  is a sample drawn from  $\text{Pr}(\mathbf{y}|\mathbf{x})$ , and  $\mathbf{x}'$  is a sample drawn from  $\text{Pr}(\mathbf{x}'|\mathbf{y})$ , where  $\text{Pr}(\mathbf{y}|\mathbf{x})$  and  $\text{Pr}(\mathbf{x}|\mathbf{y})$  are Bayes' inverses of each other, as given by

$$\text{Pr}(\mathbf{x}|\mathbf{y}) = \frac{\text{Pr}(\mathbf{y}|\mathbf{x})\text{Pr}(\mathbf{x})}{\int d\mathbf{x}' \text{Pr}(\mathbf{y}|\mathbf{x}')\text{Pr}(\mathbf{x}')} \quad (1)$$

and  $\text{Pr}(\mathbf{x})$  is the prior probability from which  $\mathbf{x}$  was sampled. Because the chain of dependences in passing from  $\mathbf{x}$  to  $\mathbf{y}$  and then to  $\mathbf{x}'$  is first order Markov (i.e. it is

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

This appeared in Mathematics in Signal Processing V, Clarendon Press, J. G. McWhirter and I. K. Proudler (eds.), 193-204, 2002.  
© British Crown Copyright 2001.

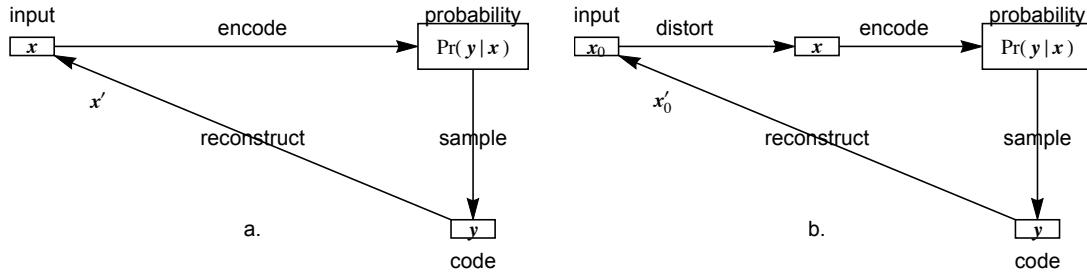


Figure 1: (a) A SVQ in which an input vector  $\mathbf{x}$  is encoded as a code index vector  $\mathbf{y}$  that is drawn from a conditional probability  $\Pr(\mathbf{y}|\mathbf{x})$ , which is then decoded as a reconstruction vector  $\mathbf{x}'$  drawn from the Bayes' inverse conditional probability  $\Pr(\mathbf{x}'|\mathbf{y})$ . (b) A SVQ in which an input vector  $\mathbf{x}_0$  is first distorted into  $\mathbf{x}$ , which is then encoded as a code index vector  $\mathbf{y}$  that is drawn from a conditional probability  $\Pr(\mathbf{y}|\mathbf{x})$ , which is then decoded as a reconstruction vector  $\mathbf{x}'_0$  drawn from the Bayes' inverse conditional probability  $\Pr(\mathbf{x}'_0|\mathbf{y})$ .

described by the directed graph  $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}'$ , and because the two ends of this Markov chain (i.e.  $\mathbf{x}$  and  $\mathbf{x}'$ ) live in the same vector space, it is called a *folded* Markov chain [6].

In order to ensure that the SVQ encodes the input vector optimally, a measure of the reconstruction error must be minimised. There are many possible ways to define this measure, but one that is consistent with many previous results, and which also leads to many new results, is the mean Euclidean reconstruction error measure  $D$ , which is defined as

$$D \equiv \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \int d\mathbf{x}' \Pr(\mathbf{x}'|\mathbf{y}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (2)$$

where  $\mathbf{y} = (y_1, y_2, \dots, y_n), 1 \leq y_i \leq M$ ,  $\|\mathbf{x} - \mathbf{x}'\|^2$  is the Euclidean reconstruction error, and  $\int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \int d\mathbf{x}' \Pr(\mathbf{x}'|\mathbf{y})$  averages over all possible states of the SVQ.

Using the Markov chain property  $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}'$ , Bayes' theorem may be written in the form  $\Pr(\mathbf{x}) \Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x}'|\mathbf{y}) = \Pr(\mathbf{y}) \Pr(\mathbf{x}|\mathbf{y}) \Pr(\mathbf{x}'|\mathbf{y})$ , which may then be used to express  $D$  in a form in which  $\mathbf{x}$  and  $\mathbf{x}'$  appear symmetrically. This allows the  $\mathbf{x}'$  integrals to be expressed in terms of the corresponding  $\mathbf{x}$  integrals, and after using Bayes' theorem in the form  $\Pr(\mathbf{x}) \Pr(\mathbf{y}|\mathbf{x}) = \Pr(\mathbf{y}) \Pr(\mathbf{x}|\mathbf{y})$  this yields the following

expression for  $D$ :

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(\mathbf{y})\|^2 \quad (3)$$

where the reconstruction vector  $\mathbf{x}'(\mathbf{y})$  is defined as  $\mathbf{x}'(\mathbf{y}) \equiv \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ . Because of the quadratic form of the objective function, it turns out that  $\mathbf{x}'(\mathbf{y})$  may be treated as a vector of free parameters, whose optimum value (i.e. the solution of  $\frac{\partial D}{\partial \mathbf{x}'(\mathbf{y})} = 0$ ) is  $\int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{y}) \mathbf{x}$ , as required.

## B. Noisy Data

By analogy with the results reported in [2], the SVQ approach can be generalised to the problem of encoding noisy or distorted data, with the intention of eventually recovering the undistorted data. The operations that occur in this type of SVQ are summarised in Figure 1(b). The input vector is  $\mathbf{x}_0$ , which is converted into the distorted input vector  $\mathbf{x}$  by a distortion process  $\Pr(\mathbf{x}|\mathbf{x}_0)$ , which is then encoded as a code index vector  $\mathbf{y}$ , which is then subsequently decoded as a reconstruction  $\mathbf{x}'_0$  of the original input vector. This is described by the directed graph  $\mathbf{x}_0 \rightarrow \mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{x}'_0$ .

The mean Euclidean reconstruction error measure  $D$  becomes (compare Equation 2)

---


$$D = \int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{x}_0) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \int d\mathbf{x}' \Pr(\mathbf{x}'_0|\mathbf{y}) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{y})\|^2 \quad (4)$$

The Bayes' inverse probability  $\Pr(\mathbf{x}'_0|\mathbf{y})$  may be integrated out of this expression for  $D$  to yield (compare Equation 3)

$$D = 2 \int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \int d\mathbf{x} \Pr(\mathbf{x}|\mathbf{x}_0) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{y})\|^2 \quad (5)$$

where the reconstruction vector  $\mathbf{x}'_0(\mathbf{y})$  is defined as  $\mathbf{x}'_0(\mathbf{y}) \equiv \int d\mathbf{x}_0 \Pr(\mathbf{x}_0|\mathbf{y}) \mathbf{x}_0$ , which may be treated as a vector of

free parameters. Bayes' theorem  $\Pr(\mathbf{x}_0) \Pr(\mathbf{x}|\mathbf{x}_0) = \Pr(\mathbf{x}) \Pr(\mathbf{x}_0|\mathbf{x})$  may be used to integrate out  $\mathbf{x}_0$  to yield

$$D = 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \|\mathbf{x}_0(\mathbf{x}) - \mathbf{x}'_0(\mathbf{y})\|^2 + \text{constant} \quad (6)$$

where  $\mathbf{x}_0(\mathbf{x})$  is defined as  $\mathbf{x}_0(\mathbf{x}) \equiv \int d\mathbf{x}_0 \Pr(\mathbf{x}_0|\mathbf{x}) \mathbf{x}_0$ ,

It is much more difficult to optimise this version of the objective function than the version in Equation 3, because the  $\mathbf{x}_0(\mathbf{x})$  term is in general a non-linear function of  $\mathbf{x}$ . Worse still, the expression for  $\mathbf{x}_0(\mathbf{x})$  involves  $\Pr(\mathbf{x}_0|\mathbf{x})$ , which depends on the unknown  $\Pr(\mathbf{x}_0)$ , so  $\mathbf{x}_0(\mathbf{x})$  cannot be computed analytically anyway. However, progress can be made by splitting  $\mathbf{x}$  into “signal” and “noise” subspaces, as will be shown in Section II C.

### C. Nuisance Degrees of Freedom

For convenience, split up the input space into (possibly non-orthogonal) subspaces as  $(\mathbf{x}_0, \mathbf{x}_\perp)$ , where the

undistorted input lives entirely in the  $\mathbf{x}_0$  subspace, and the distortion lives principally in the  $\mathbf{x}_\perp$  subspace, and any distortion that lies in the  $\mathbf{x}_0$  subspace is regarded as part of the undistorted input. For simplicity, the part of the distortion that lies in the  $\mathbf{x}_0$  subspace is ignored throughout the rest of this paper. The directed graph becomes  $(\mathbf{x}_0, \mathbf{0}) \rightarrow (\mathbf{x}_0, \mathbf{x}_\perp) \rightarrow \mathbf{y} \rightarrow (\mathbf{x}'_0, \mathbf{0})$  as shown in Figure 2(a).

The expression for  $D$  becomes (compare Equation 5)

$$D = 2 \int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \int d\mathbf{x}_\perp \Pr(\mathbf{x}_\perp|\mathbf{x}_0) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\perp) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{y})\|^2. \quad (7)$$

Now consider the related optimisation problem in which an attempt to reconstruct  $(\mathbf{x}_0, \mathbf{x}_\perp)$  is made, as shown in Figure 2(b). The corresponding objective function may be obtained by modifying Equation 7 (omitting the cross-term arising from  $\mathbf{x}_0 \cdot \mathbf{x}_\perp \neq 0$ , which will later be shown to be zero) to yield

$$D = 2 \int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \int d\mathbf{x}_\perp \Pr(\mathbf{x}_\perp|\mathbf{x}_0) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\perp) \left( \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{y})\|^2 + \|\mathbf{x}_\perp - \mathbf{x}'_\perp(\mathbf{y})\|^2 \right). \quad (8)$$

Assume for now that some of the links in Figure 2(b) are broken as shown in Figure 2(c). Because the distortion subspace is not involved in the computations in Figure 2(c), it may be redrawn as shown in Figure 2(d). This is the same as Figure 2(a), except that the encoder now disregards (or is invariant with respect to) the nuisance degrees of freedom. Note that the simplification of Figure 2(c) to Figure 2(d) ignores differences that arise due to the part of the distortion that lies in the  $\mathbf{x}_0$  subspace.

The following argument explains the conditions under which the links in Figure 2(c) may be broken. Firstly, it is necessary to *assume* that the encoder is independent of  $\mathbf{x}_\perp$ , so that  $\Pr(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\perp) = \Pr(\mathbf{y}|\mathbf{x}_0)$  (this assumption will be discussed in detail later). The  $\|\mathbf{x}_\perp - \mathbf{x}'_\perp(\mathbf{y})\|^2$  term in Equation 8 needs to simplify to a constant, which requires that  $\int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \int d\mathbf{x}_\perp \Pr(\mathbf{x}_\perp|\mathbf{x}_0) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\perp) \|\mathbf{x}_\perp - \mathbf{x}'_\perp(\mathbf{y})\|^2 = \text{constant}$ . However, to obtain this constant result, it is

sufficient to assume  $\int d\mathbf{x}_\perp \Pr(\mathbf{x}_\perp|\mathbf{x}_0) \|\mathbf{x}_\perp - \mathbf{x}'_\perp(\mathbf{y})\|^2 = \text{constant}$  independent of  $\mathbf{x}_0$  and  $\mathbf{y}$ , and to guarantee that this constant is indeed independent of  $\mathbf{x}_0$  and  $\mathbf{y}$ , it is sufficient to assume  $\Pr(\mathbf{x}_\perp|\mathbf{x}_0) = \Pr(\mathbf{x}_\perp)$  and  $\mathbf{x}'_\perp(\mathbf{y}) = \text{constant}$ . Note that the result  $\mathbf{x}'_\perp(\mathbf{y}) = \text{constant}$  follows from the assumptions  $\Pr(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\perp) = \Pr(\mathbf{y}|\mathbf{x}_0)$  and  $\Pr(\mathbf{x}_\perp|\mathbf{x}_0) = \Pr(\mathbf{x}_\perp)$  which imply  $\Pr(\mathbf{x}_\perp|\mathbf{y}) = \Pr(\mathbf{x}_\perp)$ , so it may be omitted as a separate assumption.

These assumptions may be summarised as

$$\begin{aligned} \Pr(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\perp) &= \Pr(\mathbf{y}|\mathbf{x}_0) \\ \Pr(\mathbf{x}_\perp|\mathbf{x}_0) &= \Pr(\mathbf{x}_\perp) \end{aligned} \quad (9)$$

which allow the objective function  $D$  (see Equation 8) to be replaced by the equivalent objective function (dropping a constant term)

$$D = 2 \int d\mathbf{x}_0 \Pr(\mathbf{x}_0) \sum_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}_0) \|\mathbf{x}_0 - \mathbf{x}'_0(\mathbf{y})\|^2 \quad (10)$$

This is the standard SVQ objective function (compare

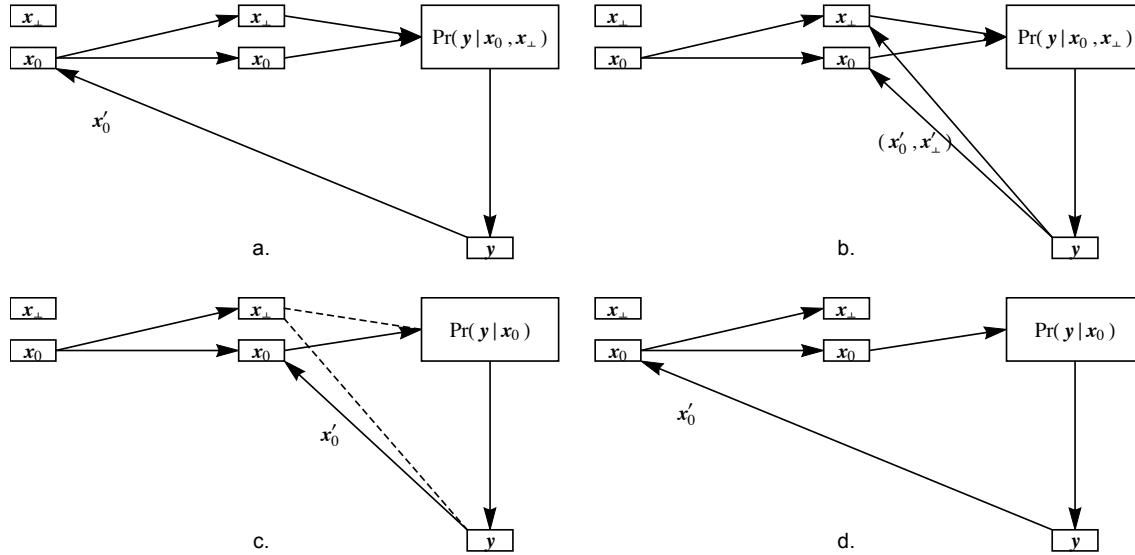


Figure 2: The input  $(\mathbf{x}_0, \mathbf{x}_\perp)$  is represented as a pair of channels, with  $\mathbf{x}_\perp$  lying above  $\mathbf{x}_0$  in each of the diagrams. (a) A SVQ in which an input vector  $(\mathbf{x}_0, \mathbf{0})$  is first distorted into  $(\mathbf{x}_0, \mathbf{x}_\perp)$ , which is then encoded as a code index vector  $\mathbf{y}$  that is drawn from a conditional probability  $\Pr(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\perp)$ , which is then decoded as a reconstruction vector  $\mathbf{x}_0'$  drawn from the Bayes' inverse conditional probability  $\Pr(\mathbf{x}_0'|\mathbf{y})$ . (b) Modified version of Figure 2(a) in which the reconstruction link is switched from the original undistorted signal  $(\mathbf{x}_0, \mathbf{0})$  to the distorted signal  $(\mathbf{x}_0, \mathbf{x}_\perp)$ . (c) Modified version of Figure 2(b) in which the encoder (and reconstruction) links from (and to) the distortion subspace are deleted (as indicated by the dashed lines). (d) Alternative version of Figure 2(c) in which the reconstruction link is moved to an equivalent position, ignoring differences that arise due to the part of the distortion that lies in the  $\mathbf{x}_0$  subspace.

Equation 3) for encoding and reconstructing the undistorted input, for which the directed graph is  $\mathbf{x}_0 \rightarrow \mathbf{y} \rightarrow \mathbf{x}_0'$ .

Note that, under the stated assumptions, the simplification leading from Equation 8 to Equation 10 occurs even if  $\mathbf{x}_0 \cdot \mathbf{x}_\perp \neq 0$ , because the potential cross-term  $\int d\mathbf{x}_\perp \Pr(\mathbf{x}_\perp|\mathbf{x}_0) (\mathbf{x}_0 - \mathbf{x}_0'(\mathbf{y})) \cdot (\mathbf{x}_\perp - \mathbf{x}_\perp'(\mathbf{y}))$  in Equation 8 is zero.

In summary, the encoder  $\Pr(\mathbf{y}|\mathbf{x}_0, \mathbf{x}_\perp)$  has access only to the distorted signal  $(\mathbf{x}_0, \mathbf{x}_\perp)$  (see Figure 2(b) and Equation 8), but the assumptions in Equation 9 force the encoder to disregard the distortion (see Figure 2(d) and Equation 10). In practice, it is not possible to satisfy the assumptions in Equation 9 in general, because it is not known in advance how to separate the distorted signal into signal and distortion  $(\mathbf{x}_0, \mathbf{x}_\perp)$ , given examples of only the distorted signal. However, it turns out that the first assumption in Equation 9 is true if  $D$  (as defined in Equation 8) is minimised under certain constraints, in which case Figure 2(d) and Equation 10 follow automatically from Figure 2(b) and Equation 8, respectively. These constraints are discussed in Section IID.

This type of encoder, in which the signal is encoded whilst the distortion is ignored, can be used as the basis of a so-called ‘‘residual vector quantiser’’ [1], in which (quoting from [1]) ‘‘the quantiser has a sequence of encoding stages, where each stage encodes the residual (error) vector of the prior stage’’. The key identification needed to make this connection is that the reconstruction error

discussed in this paper is the same as the residual (error) vector discussed in [1]. Note that a residual vector quantiser is a special case of the type of multistage encoder discussed in [9].

#### D. Optimisation Constraints

Henceforth, only the scalar code case will be considered, so the vector  $\mathbf{y}$  is now replaced by the scalar  $y$ ,  $1 \leq y \leq M$ . Also, it will be assumed that the input manifold has its centroid at the origin. In order to implement a practical optimisation procedure for minimising  $D$  (as defined in Equation 8) it is necessary to introduce a variety of assumptions and constraints.

Because  $\Pr(y|\mathbf{x})$  is a probability it satisfies  $\Pr(y|\mathbf{x}) \geq 0$  and  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$ , which are guaranteed to be true if  $\Pr(y|\mathbf{x})$  is written as

$$\Pr(y|\mathbf{x}) = \frac{Q(y|\mathbf{x})}{\sum_{y'=1}^M Q(y'|\mathbf{x})} \quad (11)$$

where  $Q(y|\mathbf{x}) \geq 0$  for all  $y$ , and  $Q(y|\mathbf{x}) > 0$  for at least one  $y$ . This removes the need to explicitly impose the constraint  $\sum_{y=1}^M \Pr(y|\mathbf{x}) = 1$  during optimisation. The  $Q(y|\mathbf{x})$  can be viewed as the unnormalised likelihood of sampling code index  $y$  from the code book.

However,  $Q(y|\mathbf{x})$  itself needs to be described by a finite number of parameters in order that the values that minimise  $D$  may be derived from a finite amount of training data. It can be shown that the optimal form of  $\Pr(y|\mathbf{x})$  must have a piecewise linear dependence on  $\mathbf{x}$  [9], and that for training data that lie on smooth curved manifolds this dependence is well approximated by a  $Q(y|\mathbf{x})$  that has the following simple piecewise linear dependence on  $\mathbf{x}$  [10]

$$Q(y|\mathbf{x}) = \begin{cases} \mathbf{u}(y) \cdot \mathbf{x} + v(y) & \mathbf{u}(y) \cdot \mathbf{x} + v(y) \geq 0 \\ 0 & \mathbf{u}(y) \cdot \mathbf{x} + v(y) \leq 0 \end{cases} \quad (12)$$

However,  $Q(y|\mathbf{x})$  needs to approximate this behaviour only in the vicinity of the data manifold, so it can be allowed to depart from this behaviour elsewhere. A convenient functional form that achieves this is the sigmoid function

$$Q(y|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}(y) \cdot \mathbf{x} - b(y))}. \quad (13)$$

This reduces the problem of minimising  $D$  to one of finding the optimal values of the  $\mathbf{w}(y)$ ,  $b(y)$  and  $\mathbf{x}'(y)$ . This may be done by using the gradient descent procedure described in [7].

If the input is an *undistorted* signal (i.e.  $\mathbf{x} = (\mathbf{x}_0, \mathbf{0})$ ) which lies on a smooth curved manifold, then the sigmoids can cooperate in encoding this input as illustrated in Figure 3, where the sigmoid threshold planes  $\mathbf{w}(y) \cdot \mathbf{x} + b(y) = 0$  are shown slicing pieces off a curved 1-dimensional manifold embedded in a 2-dimensional space.

When  $D$  is minimised, the sigmoids adjust themselves so that they partition the 1-dimensional curved manifold into a set of non-overlapping pieces [10], as illustrated in Figure 3(a). If these pieces are required to overlap, then it is sufficient to introduce an additional constraint which pulls the sigmoid threshold planes inwards, as illustrated by the dotted lines in Figure 3(b). This type of constraint is easily introduced by choosing  $b(y) = -\theta \|\mathbf{w}(y)\|$ , so that

$$Q(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\hat{\mathbf{w}}(y) \cdot \mathbf{x} - \theta) \|\mathbf{w}(y)\|)} \quad (14)$$

where  $\|\mathbf{w}(y)\| = \sqrt{\mathbf{w}(y) \cdot \mathbf{w}(y)}$  and  $\hat{\mathbf{w}}(y) = \frac{\mathbf{w}(y)}{\|\mathbf{w}(y)\|}$ . The sigmoid threshold planes are then  $\hat{\mathbf{w}}(y) \cdot \mathbf{x} = \theta$ , which lie at a distance  $\theta$  from the origin.

If the input is a *distorted* signal (i.e.  $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_\perp)$ ) then it lies on a “thickened” version of the smooth curved manifold of Figure 3(b), which is shown in Figure 3(c). The thickness represents the nuisance degrees of freedom, which can lie both in the  $\mathbf{x}_0$  subspace and the  $\mathbf{x}_\perp$  subspace, but only the latter are represented in Figure 3(c). If the threshold constraint  $b(y) = -\theta \|\mathbf{w}(y)\|$  is used (see Equation 14), as illustrated by the dotted lines in Figure 3(c), then the sigmoids slice pieces off the curved manifold in a way that disregards the nuisance degrees of freedom.

The threshold constraint in Equation 14 thus implements the invariant behaviour required by  $\Pr(y|\mathbf{x}_0, \mathbf{x}_\perp) = \Pr(y|\mathbf{x}_0)$  in Equation 9, and is analogous to the threshold trick that was introduced in [11] to encourage the emergence of neural responses that were invariant with respect to various degrees of freedom in their input.

In practice, for numerical efficiency and to encourage the optimisation procedure to locate the global minimum of  $D$ , it is useful to introduce two additional constraints. Firstly, in problems where the input manifold is sliced into overlapping pieces as shown in Figure 3(c), optimal solutions approximately satisfy  $\mathbf{x}'(y) \propto \mathbf{w}(y)$ , so good solutions are obtained if each reconstruction vector  $\mathbf{x}'(y)$  is forced to lie parallel to the corresponding weight vector  $\mathbf{w}(y)$ . Secondly, the norm of the weight vectors  $\|\mathbf{w}(y)\|$  can be constrained as  $\|\mathbf{w}(y)\| = w_0$ , in order to avoid situations where they grow to rather large values which make  $Q(y|\mathbf{x})$  (and hence  $\Pr(y|\mathbf{x})$ ) depend very strongly on  $\mathbf{x}$  in some regions. Both of these constraints speed up convergence to the global minimum of  $D$ , and can finally be lifted in the vicinity of an optimal solution to obtain complete convergence.

## E. Jammer Nulling

A number of examples of typical behaviours of  $\Pr(y|\mathbf{x})$  are shown in Figure 4.

The signal and jammer degrees of freedom generate a pair of non-orthogonal subspaces, whose axes are indicated in bold (only one dimension of each of these subspaces is illustrated in Figure 4). The response contours of a variety of possible  $\Pr(y|\mathbf{x})$  are shown. The “full” case in Figure 4(a) shows the contours of each of a pair of  $\Pr(y|\mathbf{x})$ , one of which responds only to the signal subspace, and the other only to the jammer subspace. The “signal” case in Figure 4(b) shows the contours of a  $\Pr(y|\mathbf{x})$  that responds only to only the signal subspace, and is thus invariant over the jammer subspace. The “jammer” case in Figure 4(c) is the converse of “signal” case, so that  $\Pr(y|\mathbf{x})$  responds to only the jammer subspace, and is thus invariant over the signal subspace. This argument may readily be generalised to any number of  $\Pr(y|\mathbf{x})$ .

If it is assumed that the jammer is the “large” degree of freedom and the signal is the “small” degree of freedom, then the signal and jammer subspaces may be separated by adjusting the threshold parameter  $\theta$  in Equation 14 so that in Figure 4 the “jammer” case (i.e. Figure 4(c)) is obtained, in which case  $\Pr(y|\mathbf{x})$  for  $y = 1, 2, \dots, M$  will all become invariant over the signal subspace. The jammer subspace is then spanned by the set of gradient vectors  $\nabla \Pr(y|\mathbf{x})$  for  $y = 1, 2, \dots, M$ , which can thus be used to construct a projection operator  $\mathbf{J}$  onto the jammer subspace, and a projection operator  $\mathbf{1} - \mathbf{J}$  onto the signal subspace. This definition of the projection operator may also be used in cases where the jammer and signal

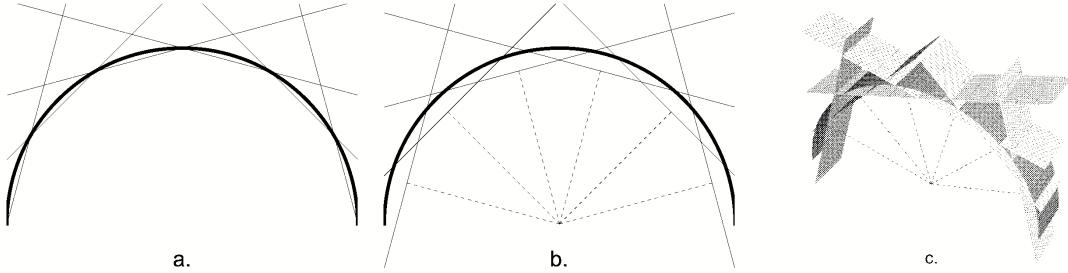


Figure 3: (a) Illustration of how a number of sigmoids slice non-overlapping pieces off a curved signal manifold. (b) Illustration of how a threshold constraint can be used to force a number of sigmoids to slice overlapping pieces off a curved signal manifold. (c) Illustration of how a number of sigmoids slice overlapping pieces off a curved signal manifold thickened by nuisance degrees of freedom.

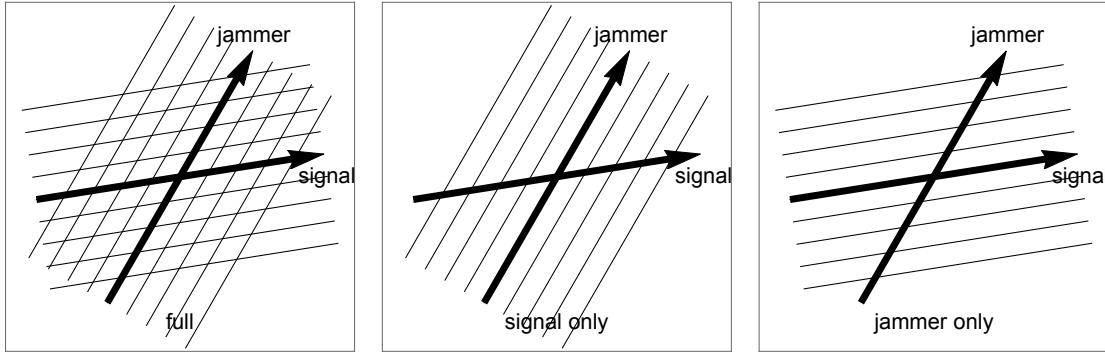


Figure 4: Examples of the response (represented as a contour plot) of  $\Pr(y|\mathbf{x})$  to signal and jammer subspaces. (a) The “full” case shows a pair of  $\Pr(y|\mathbf{x})$ , one of which responds only to the jammer and the other only to the signal. (b) The “signal” case shows a single  $\Pr(y|\mathbf{x})$  responding only to the signal. (c) The “jammer” case shows a single  $\Pr(y|\mathbf{x})$  responding only to the jammer.

subspaces are curved, so that the directions of their axes are functions of  $\mathbf{x}$ , and all of the straight lines in Figure 4 are replaced by curves defining a curvilinear coordinate system and its coordinate surfaces. The projection operator then becomes a function of the input vector  $\mathbf{x}$ , so that  $\mathbf{J} = \mathbf{J}(\mathbf{x})$ . Note that curved subspaces are the norm rather than the exception.

### III. SIMULATIONS

The optimisation of the encoder may be done by minimising  $D$  in Equation 3 using gradient descent [7], using the threshold  $\theta$  in the sigmoid function in Equation 14 to constrain the optimisation so that it encodes only the jammer subspace, as illustrated in Figure 4(c), and discussed in Section II D and Section II E.

In these simulations the input vector  $\mathbf{x}$  is 100-dimensional so that  $\mathbf{x} = (x_1, x_2, \dots, x_{100})$ , and each vector in the training set is independently generated as a superposition of a pair of “sinc” response functions

$$x_i = a_s \frac{\sin(\frac{i-i_s}{\sigma})}{\frac{i-i_s}{\sigma}} + a_j \frac{\sin(\frac{i-i_j}{\sigma})}{\frac{i-i_j}{\sigma}} \quad (15)$$

where  $a_s$  is the signal amplitude that is uniformly distributed in the interval  $[-\sqrt{10^{-3}}, \sqrt{10^{-3}}]$  (this corresponds to a signal level of -30dB),  $a_j$  is the jammer amplitude that is uniformly distributed in the interval  $[-1, 1]$  (this corresponds to a jammer level of 0dB),  $i_s$  is the signal location that is chosen to be 50,  $i_j$  is the jammer location that is uniformly distributed in the interval  $[38 - \Delta, 38 + \Delta]$  ( $\Delta = 0, 2, 4$  is used in the simulations), and  $\sigma$  is the width of the response function that is chosen to be 2. The peak and the first zero of the “sinc” response function are separated by  $\pi\sigma$ , which defines the resolution cell size. The mean jammer location  $\langle i_j \rangle$  and the signal location satisfy  $i_s - \langle i_j \rangle = 12$ , which corresponds to a separation of  $\frac{12}{\pi\sigma} \approx 2$  resolution cells. Random noise uniformly distributed in the interval  $[-\sqrt{10^{-5}}, \sqrt{10^{-5}}]$  (this corresponds to a noise level of -50dB) is also added to each component of the training vector.

In Figure 5 an encoder is trained on three different jammer scenarios  $\Delta = 0, 2, 4$ . After training the encoder is tested for how well it can be used to null a pure jammer (i.e. with no signal or noise added), where the degree of nulling is defined as the ratio of the squared lengths of the nulled input vector and the original input vector.

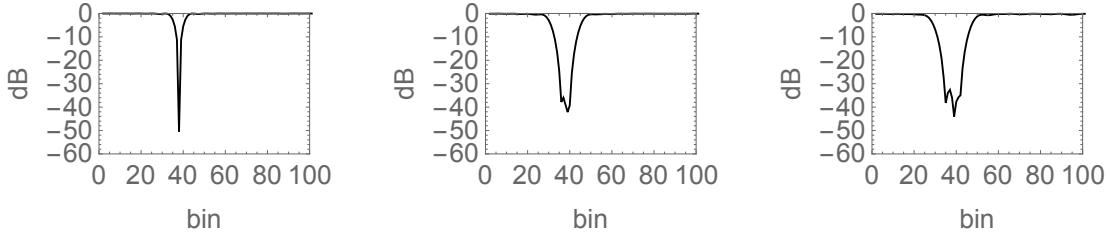


Figure 5: Plot of degree of nulling against jammer location, for jammer locations that are spread over the intervals (2 units is approximately a resolution cell) (a) [38, 38] using  $M = 2$ , (b) [36, 40] using  $M = 4$ , and (c) [34, 42] using  $M = 6$ .

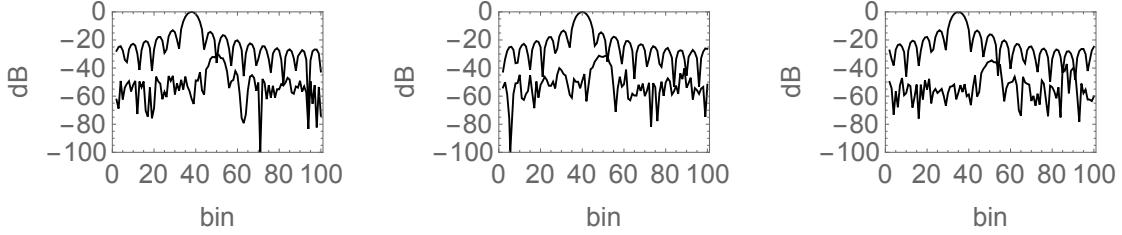


Figure 6: Plot of a typical input vector before (upper curve) and after (lower curve) jammer nulling, for jammer locations that are spread over the intervals (2 units is approximately a resolution cell) (a) [38, 38] using  $M = 2$ , (b) [36, 40] using  $M = 4$ , and (c) [34, 42] using  $M = 6$ . These correspond to the degree of nulling plots in Figure 5.

This is a good test of the ability of the encoder to learn the shape of the jammer manifold which is generated by sweeping this profile over the interval  $[38 - \Delta, 38 + \Delta]$ . When  $\Delta = 0$  there is a sharp minimum at the jammer location  $i_j = 38$ , as expected. When  $\Delta = 2$  the minimum becomes spread over jammer locations  $i_j$  in the interval  $[36, 40]$  (i.e. approximately 2 resolution cells), and when  $\Delta = 4$  the minimum becomes spread even more broadly over jammer locations  $i_j$  in the interval  $[34, 42]$  (i.e. approximately 4 resolution cells). All of these results are as expected.

In Figure 6 typical examples of an input vector together with how it appears after jammer nulling are shown for each of the jammer scenarios considered in Figure 5. In every case the signal is clearly revealed at its correct location after nulling the jammer.

In all of these training scenarios, one could envisage further constraining some of the properties of the encoder, in order to introduce prior knowledge of the form of the jammer and/or signal subspaces, and to thereby reduce the computational complexity of the jammer nulling. For instance, the signal subspace could be predefined, as in conventional algorithms which hold constant the response in a predefined ‘‘look direction’’. Similarly, the jammer subspace could be built out of pre-defined subspaces which are optimised so as to maximally null the jammer(s), as in conventional algorithms in which a number of jammer ‘‘templates’’ are used to remove the jammer(s). In general, by choosing appropriate additional constraints, the SVQ approach to jammer nulling can be made backwardly compatible with conven-

tional approaches.

#### IV. CONCLUSIONS

The theory of stochastic vector quantisers (SVQ) [7] has been extended to allow the quantiser to develop invariances, so that only ‘‘large’’ degrees of freedom in the input vector are represented in the code. This has been applied to the problem of encoding data vectors which are a superposition of a ‘‘large’’ jammer and a ‘‘small’’ signal, so that only the jammer is represented in the code. This allows the jammer to be subtracted from the total input vector (i.e. the jammer is nulled), leaving a residual that contains only the underlying signal. Several numerical simulations have shown how that idea works in practice, even when the jammer location is uncertain so that the jammer subspace is curved.

The main advantage of this approach to jammer nulling is that little prior knowledge of the jammer is assumed, because these properties are automatically discovered by the SVQ as it is trained on examples of input vectors. Provided that the signal is much weaker than the jammer, the SVQ acquires an internal representation of only the jammer manifold, so it is invariant with respect to the signal manifold. In a sense, the SVQ regards the ‘‘large’’ jammer as the normal type of input that it expects to encode, whereas it regards the ‘‘small’’ signal as an anomaly that is not to be encoded.

## V. ACKNOWLEDGEMENT

This work was carried out as part of Technology Group 10 of the MoD Corporate Research Programme.

- [1] C F Barnes, S A Rizvi, and N M Nasrabadi, *Advances in residual vector quantisation: a review*, IEEE Transactions on Image Processing **5** (1996), no. 2, 226–262.
- [2] Y Ephraim and R M Gray, *A unified approach for encoding clean and noisy sources by means of waveform and autoregressive model vector quantisation*, IEEE Transactions on Information Theory **34** (1988), no. 4, 826–834.
- [3] A Gersho and R M Gray, *Vector quantisation and signal compression*, Kluwer, Norwell, 1992.
- [4] R M Gray, *Vector quantisation*, IEEE Transactions on Acoustics, Speech, and Signal Processing Magazine **1** (1984), no. 2, 4–29.
- [5] Y Linde, A Buzo, and R M Gray, *An algorithm for vector quantiser design*, IEEE Transactions on Communications **28** (1980), no. 1, 84–95.
- [6] S P Luttrell, *A Bayesian analysis of self-organising maps*, Neural Computation **6** (1994), no. 5, 767–794.
- [7] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
- [8] ———, *Self-organisation of multiple winner-take-all neural networks*, Connection Science **9** (1997), no. 1, 11–30.
- [9] ———, *An adaptive network for encoding data using piecewise linear functions*, Proceedings of international conference on artificial neural networks (Edinburgh), IEE, 1999, pp. 198–203.
- [10] ———, *Combining artificial neural nets: Ensemble and modular multi-net systems*, Perspectives in Neural Computing, ch. Self-organised modular neural networks for encoding data, pp. 235–263, Springer-Verlag, London, 1999.
- [11] C J S Webber, *Self-organisation of transformation-invariant detectors for constituents of perceptual patterns*, Network: Computation in Neural Systems **5** (1994), no. 4, 471–496.

# A Markov Chain Approach to Multiple Classifier Fusion \*

S P Luttrell  
Room EX21, QinetiQ, Malvern Technology Centre

The aim of this paper is to show how the problem of optimising the processing chain in a classifier combiner network can be recast as a Markov chain optimisation problem. A summary of the application of coding theory to Markov chains and to encoder networks is presented, and the key idea of optimising the *joint* probability density of the state of the processing chain is stated. An example of the application of these ideas to processing data from multiple correlated sources (i.e. a hierarchically correlated phase screen) is then given.

## I. INTRODUCTION

The aim of this paper is to show how the problem of optimising the processing chain in a classifier combiner network can be recast as a Markov chain optimisation problem, where the input to the chain is the original data (possibly derived from multiple sensors) that needs to be classified, and the stages of the Markov chain progressively process and combine the data as it passes along the chain.

The key problem is to choose a suitable objective function for optimising the properties of the Markov chain. In this paper the joint PDF of the states in the chain is represented in two ways corresponding to forwards and backwards passes through the chain. The objective is to maximise the log-likelihood that the backwards pass generates states that look as if they were derived from the forward pass, which is equivalent to minimising the number of bits per symbol needed to specify the state of the chain (forward pass) using a model (backward pass). This idea was originally proposed in [3]. The objective function can be interpreted as imposing a self-consistency condition on the forwards and backwards passes through the Markov chain. Note that although these two directions are related by Bayes theorem, in this approach they are treated as being independent, although they must satisfy the above self-consistency condition.

It turns out that this type of Markov chain is very useful for implementing classifier combiner networks, because if the forward pass through the chain is arranged to compress the data (e.g. by progressively reducing the size of the state space as the data passes along the chain), then the above optimisation is forced to find efficient ways of encoding the data. Typically, as the data passes along the chain its small degrees of freedom and its internal correlations are progressively removed, and by the time the data emerges from the far end of the chain only a few large degrees of freedom remain. In the classifier combiner context the data is encoded in great detail near the

start of the chain, whereas it is encoded in coarse outline near the far end of the chain, and the progressive compression of the fine detail into coarse outline has exactly the form of a familiar (and possibly hierarchical) classifier combiner network.

In Section II a brief summary of the relevant parts of coding theory is given, focusing on coding of sources and then generalising to Markov sources. In Section III these results are applied to Markov chains used as encoder networks, and the results obtained are shown to be equivalent to the self organising network introduced in [2] and used for some simple classifier combiner applications in [4]. In Section IV these results are applied to the problem of designing classifier combiner networks, and this is illustrated by a simulation in which hierarchically correlated phase screen data causes the self-organising network to develop a hierarchical data processing architecture.

## II. CODING THEORY

The purpose of this section is to firmly establish the underlying principles on which the later derivations depend. In Section II A a brief summary of source coding theory is given, and in Section II B this is generalised to Markov chain sources. A useful reference work on coding theory is [6], the original papers on information theory are [8, 9], and for a discussion on the number of bits required to encode a source see [7].

### A. Source Coding

A source of symbols (drawn from an alphabet of  $M$  distinct symbols) is modelled by a vector of probabilities  $\mathbf{P}$

$$\mathbf{P} \equiv (P_1, P_2, \dots, P_M) \quad (2.1)$$

which describes the relative frequency with which each symbol is drawn independently from the source  $\mathbf{P}$ . An ordered sequence of symbols may be partitioned into long subsequences of  $N$  symbols, and each such subsequence will be called a message. A message from  $\mathbf{P}$  will be called a  $\mathbf{P}$ -message. The number of times  $n_i$  that each symbol  $i$  occurs in a  $\mathbf{P}$ -message of length  $N$  is  $n_i = N P_i$ , where

---

\*Typeset in L<sup>A</sup>T<sub>E</sub>X on May 21, 2019.

This appeared in Proceedings of the 4th International Workshop on Multiple Classifier Systems, Guildford, 11-13 June 2003, pp 217-226. © Copyright QinetiQ Ltd 2003

$\sum_{i=1}^M P_i = 1$  guarantees that the normalisation condition  $\sum_{i=1}^M n_i = N$  is satisfied. Define the entropy  $H(\mathbf{P})$  of source  $\mathbf{P}$  as the logarithm of the number of different  $\mathbf{P}$ -messages (per message symbol)

$$\begin{aligned} H(\mathbf{P}) &\equiv -\sum_{i=1}^M P_i \log P_i \\ &\geq 0 \end{aligned} \quad (2.2)$$

where  $H(\mathbf{P})$  is the number of bits per symbol (on average) that is required to encode the source assuming a perfect encoder.

The mathematical model of  $\mathbf{P}$  may be derived from a vector of probabilities  $\mathbf{Q}$ , whose  $M$  elements model the probability of each symbol drawn from an alphabet of  $M$  distinct symbols. The log-probability  $\Pi_N(\mathbf{P}, \mathbf{Q})$  that a  $\mathbf{Q}$ -message is a  $\mathbf{P}$ -message is

$$\begin{aligned} \Pi_N(\mathbf{P}, \mathbf{Q}) &\approx -N \sum_{i=1}^M P_i \log \frac{P_i}{Q_i} \\ &\leq 0 \end{aligned} \quad (2.3)$$

which is negative because the model  $\mathbf{Q}$  generates  $\mathbf{P}$ -messages with less than unit probability. The model  $\mathbf{Q}$  must be used to generate enough  $\mathbf{Q}$ -messages to ensure that all of the  $\mathbf{P}$ -messages are reproduced. This requires the basic  $H(\mathbf{P})$  bits per symbol that would be required if  $\mathbf{Q} = \mathbf{P}$ , plus some extra bits to compensate for the less than 100% efficiency (because  $\mathbf{Q} \neq \mathbf{P}$ ) with which  $\mathbf{Q}$  generates  $\mathbf{P}$ -messages. The number of extra bits per symbol is the relative entropy  $G(\mathbf{P}, \mathbf{Q})$  which is defined as

$$\begin{aligned} G(\mathbf{P}, \mathbf{Q}) &\equiv \sum_{i=1}^M P_i \log \frac{P_i}{Q_i} \\ &\geq 0 \end{aligned} \quad (2.4)$$

Thus  $G(\mathbf{P}, \mathbf{Q})$  is  $-\frac{\Pi_N(\mathbf{P}, \mathbf{Q})}{N}$ , which is minus the log-probability (per symbol) that a  $\mathbf{Q}$ -message is a  $\mathbf{P}$ -message. Thus  $\mathbf{Q}$  is used to generate exactly the number of extra  $\mathbf{Q}$ -messages that is required to compensate for the fact that the probability that each  $\mathbf{Q}$ -message is a  $\mathbf{P}$ -message is less than unity (i.e.  $\Pi_N(\mathbf{P}, \mathbf{Q}) \leq 0$ ).  $G(\mathbf{P}, \mathbf{Q})$  (i.e. relative entropy) is the amount by which the number of bits per symbol exceeds the lower bound  $H(\mathbf{P})$  (i.e. source entropy). For convenience, define the total number of bits per symbol  $H(\mathbf{P}) + G(\mathbf{P}, \mathbf{Q})$  as  $L(\mathbf{P}, \mathbf{Q})$

$$\begin{aligned} L(\mathbf{P}, \mathbf{Q}) &\equiv H(\mathbf{P}) + G(\mathbf{P}, \mathbf{Q}) \\ &= -\sum_{i=1}^M P_i \log Q_i \\ &\geq 0 \end{aligned} \quad (2.5)$$

## A Markov Chain Approach to Multiple Classifier Fusion

The expression for  $G(\mathbf{P}, \mathbf{Q})$  provides a means of optimising the model  $\mathbf{Q}$ . Ideally the number of extra bits that is required to compensate for the inefficiency of the model should be as small as possible, which requires that the optimum model  $\mathbf{Q}_{opt}$  should minimise the objective function  $G(\mathbf{P}, \mathbf{Q})$  with respect to  $\mathbf{Q}$ , thus

$$\begin{aligned} \mathbf{Q}_{opt} &= \arg \min_{\mathbf{Q}} G(\mathbf{P}, \mathbf{Q}) \\ &= \arg \max_{\mathbf{Q}} \log(Q_1^{n_1} Q_2^{n_2} \cdots Q_M^{n_M}) \end{aligned} \quad (2.6)$$

where  $\log(Q_1^{n_1} Q_2^{n_2} \cdots Q_M^{n_M})$  is the log-probability that a message of length  $N$  generated by  $\mathbf{Q}$  is a  $\mathbf{P}$ -message.  $G(\mathbf{P}, \mathbf{Q})$  is frequently used as an objective function in density modelling. The optimum model  $\mathbf{Q}_{opt}$  is chosen as the one that maximises the log-probability of generating the observed data  $(n_1, n_2, \dots, n_M)$ . Since  $\mathbf{Q}_{opt}$  must, in some sense, be close to  $\mathbf{P}$ , this affords a practical way of ensuring that the optimum model probabilities  $\mathbf{Q}_{opt}$  are similar to the source probabilities  $\mathbf{P}$ , which is the goal of density modelling.

## B. Markov Chain Coding

The above scheme for using a model  $\mathbf{Q}$  to code symbols derived from a source  $\mathbf{P}$  may be extended to the case where the source and the model are  $L$ -stage Markov chains as discussed in [3]. Thus  $\mathbf{P}$  and  $\mathbf{Q}$  can be split into separate pieces associated with each stage

$$\begin{aligned} \mathbf{P} &= \mathbf{P}^0 \mathbf{P}^{1|0} \cdots \mathbf{P}^{L-1|L-2} \mathbf{P}^L | \mathbf{L}^{-1} \\ &= \mathbf{P}^{0|1} \mathbf{P}^{1|2} \cdots \mathbf{P}^{L-1|L} \mathbf{P}^L \\ \mathbf{Q} &= \mathbf{Q}^0 \mathbf{Q}^{1|0} \cdots \mathbf{Q}^{L-1|L-2} \mathbf{Q}^L | \mathbf{L}^{-1} \\ &= \mathbf{Q}^{0|1} \mathbf{Q}^{1|2} \cdots \mathbf{Q}^{L-1|L} \mathbf{Q}^L \end{aligned} \quad (2.7)$$

where  $\mathbf{P}^{k|l}$  ( $\mathbf{Q}^{k|l}$ ) is the matrix of transition probabilities from layer  $l$  to layer  $k$  of the Markov chain of the source (model),  $\mathbf{P}^0$  ( $\mathbf{Q}^0$ ) is the vector of marginal probabilities in layer 0, and  $\mathbf{P}^L$  ( $\mathbf{Q}^L$ ) is the vector of marginal probabilities in layer  $L$ . These two ways of decomposing  $\mathbf{P}$  (and  $\mathbf{Q}$ ) are equivalent, because a forward pass through a Markov chain may be converted into a backward pass through another Markov chain, where the transition probabilities in the two chains are related via Bayes' theorem.

The total number of bits per symbol required to code the source  $\mathbf{P}$  with the model  $\mathbf{Q}$  is  $L(\mathbf{P}, \mathbf{Q})$  (i.e.  $H(\mathbf{P}) + G(\mathbf{P}, \mathbf{Q})$ ) is given by

$$L(\mathbf{P}, \mathbf{Q}) = \sum_{l=0}^{L-1} \sum_{i_{l+1}=1}^{M_{l+1}} P_{i_{l+1}}^{l+1} L_{i_{l+1}}(\mathbf{P}^{l|l+1}, \mathbf{Q}^{l|l+1}) + L(\mathbf{P}^L, \mathbf{Q}^L) \quad (2.8)$$


---

where the suffix  $i_{l+1}$  appears on the  $L_{i_{l+1}}(\mathbf{P}^{l|l+1}, \mathbf{Q}^{l|l+1})$  because the state of layer  $l+1$  is fixed during the evaluation of  $L_{i_{l+1}}(\mathbf{P}^{l|l+1}, \mathbf{Q}^{l|l+1})$ .

This result has a very natural interpretation. Both the source  $\mathbf{P}$  and the model  $\mathbf{Q}$  are Markov chains, and corresponding parts of the model are matched up with corresponding parts of the source. First of all, the number of bits that is required to encode the  $L^{th}$  layer of the source is  $L(\mathbf{P}^L, \mathbf{Q}^L)$ . Having done that, the number of bits that is required to encode the  $L-1^{th}$  layer of the source, given that the state of the  $L^{th}$  layer is already known, is  $L_L(\mathbf{P}^{L-1|L}, \mathbf{Q}^{L-1|L})$ , which must then be averaged over the alternative possible states of the  $L^{th}$  layer to yield  $\sum_{i_L=1}^{M_L} P_{i_L}^L L_L(\mathbf{P}^{L-1|L}, \mathbf{Q}^{L-1|L})$ . This pro-

cess is then repeated to encode the  $L-2^{th}$  layer of the source, given that the state of the  $L-1^{th}$  layer is already known, and so on back to layer 0. This yields precisely the expression for  $L(\mathbf{P}, \mathbf{Q})$  given above.

Now evaluate the expression for  $L(\mathbf{P}, \mathbf{Q})$  in the case where  $\mathbf{P}$  and  $\mathbf{Q}$  run in opposite directions through the Markov chain. Define  $K_{i_l}(\mathbf{P}^{l+1|l}, \mathbf{Q}^{l|l+1})$  as

$$K_{i_l}(\mathbf{P}^{l+1|l}, \mathbf{Q}^{l|l+1}) \equiv - \sum_{i_{l+1}=1}^{M_{l+1}} P_{i_{l+1}, i_l}^{l+1|l} \log Q_{i_l, i_{l+1}}^{l|l+1} \quad (2.9)$$

which allows the  $P_{i_{l+1}}^{l+1} L_{i_{l+1}}(\mathbf{P}^{l|l+1}, \mathbf{Q}^{l|l+1})$  terms in  $L(\mathbf{P}, \mathbf{Q})$  to be rewritten as

$$\sum_{i_{l+1}=1}^{M_{l+1}} P_{i_{l+1}}^{l+1} L_{i_{l+1}}(\mathbf{P}^{l|l+1}, \mathbf{Q}^{l|l+1}) = \sum_{i_l=1}^{M_l} P_{i_l}^l K_{i_l}(\mathbf{P}^{l+1|l}, \mathbf{Q}^{l|l+1}) \quad (2.10)$$

whence  $L(\mathbf{P}, \mathbf{Q})$  may be written as

$$L(\mathbf{P}, \mathbf{Q}) = \sum_{l=0}^{L-1} \sum_{i_l=1}^{M_l} P_{i_l}^l K_{i_l}(\mathbf{P}^{l+1|l}, \mathbf{Q}^{l|l+1}) + L(\mathbf{P}^L, \mathbf{Q}^L) \quad (2.11)$$


---

### III. ENCODER NETWORKS

The purpose of this section is to translate the Markov chain theory results that were summarised in the previous section into a notation that is more appropriate for processing vectors of input data, and which also makes contact with previous results on encoder networks. In Section III A the results of Section III B are shown to be equivalent to stochastic vector quantiser (SVQ) theory (which is a particular instance of ACEnet - the adaptive cluster expansion network [1]), and in Section III B this is generalised to multi-stage stochastic vector quantiser theory.

#### A. 1-Stage Markov Chain

Use the expression for  $L(\mathbf{P}, \mathbf{Q})$  in Equation 2.11 to obtain the objective function for a 1-stage (i.e.  $L = 1$ )

Markov chain

$$L(\mathbf{P}, \mathbf{Q}) = \sum_{i_0=1}^{M_0} P_{i_0}^0 K_{i_0}(\mathbf{P}^{1|0}, \mathbf{Q}^{0|1}) + L(\mathbf{P}^1, \mathbf{Q}^1) \quad (3.1)$$

Now change notation in order to make contact with previous results on autoencoders [2]:  $i_0 \rightarrow \mathbf{x}$  (input vector),  $i_1 \rightarrow \mathbf{y}$  (output code index vector),  $\sum_{i_0=1}^{M_0} \rightarrow \int d\mathbf{x}$ ,  $\sum_{i_1=1}^{M_1} \rightarrow \sum_{\mathbf{y}}$ ,  $P_{i_0}^0 \rightarrow \Pr(\mathbf{x})$  (input PDF),  $P_{i_1, i_0}^{1|0} \rightarrow \Pr(\mathbf{y}|\mathbf{x})$  (encoder),  $Q_{i_0, i_1}^{0|1} \rightarrow V \frac{1}{(\sqrt{2\pi}\sigma)^{\dim \mathbf{x}}} \exp(-\frac{\|\mathbf{x}-\mathbf{x}'(\mathbf{y})\|^2}{2\sigma^2})$  (decoder, where  $V$  is an infinitesimal volume element). This allows  $L(\mathbf{P}, \mathbf{Q})$  to be written as

$$L(\mathbf{P}, \mathbf{Q}) = \frac{1}{4\sigma^2} D_{SVQ} - \log \left( \frac{V}{(\sqrt{2\pi}\sigma)^{\dim \mathbf{x}}} \right) + L(\mathbf{P}^1, \mathbf{Q}^1) \quad (3.2)$$


---

where  $D_{SVQ}$  is the objective function for a stochastic vector quantiser (SVQ) defined as in [2]

$$D_{SVQ} \equiv 2 \int d\mathbf{x} \Pr(\mathbf{x}) \sum_y \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \quad (3.3)$$

If the cost of coding the output (i.e.  $L(\mathbf{P}^1, \mathbf{Q}^1)$ ) is ignored, then provided that  $V$  and  $\sigma$  are fixed quantities, the 2-layer Markov source coding objective function  $L(\mathbf{P}, \mathbf{Q})$  can be minimised by minimising the SVQ objective function  $D_{SVQ}$ .

If  $\mathbf{y} = (y_1, y_2, \dots, y_n)$  where  $y_i = 1, 2, \dots, M$ , and  $\Pr(\mathbf{y}|\mathbf{x}) = \Pr(y_1|\mathbf{x}) \Pr(y_2|\mathbf{x}) \dots \Pr(y_n|\mathbf{x})$ , then  $D_{SVQ} \leq D_1 + D_2$  where  $D_1$  and  $D_2$  are defined as in [2]

$$D_1 \equiv \frac{2}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \sum_{y=1}^M \Pr(y|\mathbf{x}) \|\mathbf{x} - \mathbf{x}'(y)\|^2 \quad (3.4)$$

$$D_2 \equiv \frac{2(n-1)}{n} \int d\mathbf{x} \Pr(\mathbf{x}) \left\| \mathbf{x} - \sum_{y=1}^M \Pr(y|\mathbf{x}) \mathbf{x}'(y) \right\|^2$$

$\mathbf{x}'(y)$  is used to approximate  $\mathbf{x}'(\mathbf{y})$  thus

$$\begin{aligned} \mathbf{x}'(\mathbf{y}) &\approx \frac{1}{n} \sum_{i=1}^n \mathbf{x}'(y_i) \\ &= \sum_{y=1}^M n(y) \mathbf{x}'(y) \end{aligned} \quad (3.5)$$


---

where  $n(y)$  is the number of times that the term  $\mathbf{x}'(y)$  appears in the sum  $\sum_{i=1}^n \mathbf{x}'(y_i)$ , so  $\mathbf{n} = (n(1), n(2), \dots, n(M))$  (where  $n = \sum_{y=1}^M n(y)$ ) is the histogram derived from the vector of samples  $\mathbf{y} = (y_1, y_2, \dots, y_n)$  drawn from  $\Pr(\mathbf{y}|\mathbf{x})$ . Similarly  $\Pr(\mathbf{y}|\mathbf{x})$  may be written as

$$\begin{aligned} \Pr(\mathbf{y}|\mathbf{x}) &= \prod_{i=1}^n \Pr(y_i|\mathbf{x}) \\ &= \prod_{y=1}^M \Pr(y|\mathbf{x})^{n(y)} \end{aligned} \quad (3.6)$$

so  $\Pr(\mathbf{y}|\mathbf{x})$  is a function of only the histogram  $\mathbf{n}$  of samples independently drawn from  $\Pr(y|\mathbf{x})$  (i.e.  $\mathbf{n}$  is a sufficient set of statistics for  $\Pr(\mathbf{y}|\mathbf{x})$ ). This fact allows the histogram  $\mathbf{n}$  to be used instead of the vector of samples  $\mathbf{y}$  where convenient. Note that the normalised histogram  $\frac{\mathbf{n}}{n}$  is also a sufficient statistic, because the value of  $n$  is known so it can later be removed to recover  $\mathbf{n}$ .

## B. Multi-Stage Markov Chain

The results will now be generalised to an  $L$ -stage (i.e.  $L+1$  layers) Markov chain to yield

$$L(\mathbf{P}, \mathbf{Q}) = \sum_{l=0}^{L-1} \left( \frac{D_{SVQ}^l}{4(\sigma^l)^2} - \log \left( \frac{V^l}{(\sqrt{2\pi}\sigma^l)^{\dim \mathbf{x}^l}} \right) \right) + L(\mathbf{P}^L, \mathbf{Q}^L) \quad (3.7)$$


---

where  $\mathbf{x}^l$  is the input vector to stage  $l$ , which is chosen to be the sufficient statistic  $\frac{\mathbf{n}^{l-1}}{n^{l-1}}$  derived from the output vector  $\mathbf{y}^{l-1}$  from stage  $l-1$ . Unlike the original Markov chain, this sufficient statistic prescription causes the input to stage  $l$  to be *different from* the output from stage  $l-1$ , but this difference does not affect the statistical properties of the Markov chain, as would be expected since sufficient statistics have been used. Note that other sufficient statistics (such as  $\mathbf{n}$  itself) could be used instead of the above prescription.

$L(\mathbf{P}, \mathbf{Q})$  is a sum of 1-stage SVQ objective functions

(where each term is weighted by  $(\sigma^l)^{-2}$ ), plus an output coding cost  $L(\mathbf{P}^L, \mathbf{Q}^L)$ . If the cost of coding the final output is ignored, then the multi-stage Markov chain objective function  $L(\mathbf{P}, \mathbf{Q})$  is minimised by minimising the sum of 1-stage SVQ objective functions  $\sum_{l=0}^{L-1} \frac{D_{SVQ}^l}{(\sigma^l)^2}$ .

These main benefit of the results is that they provide a unifying framework for a wide variety of previous results (e.g. [1, 2, 4, 5]). For instance, in [4] a simple classifier combiner problem was addressed, in which the stages of the chain were linked together, not by the nor-

malised histogram  $\frac{\mathbf{n}}{n}$ , but by the vector of probabilities  $(\Pr(y = 1|\mathbf{x}), \Pr(y = 2|\mathbf{x}), \dots, \Pr(y = M|\mathbf{x}))$ . In the limit  $n \rightarrow \infty$  these are equivalent prescriptions, because the normalised histogram represented by  $\frac{\mathbf{n}}{n}$  has a limiting form that is identical to the underlying probabilities  $(\Pr(y = 1|\mathbf{x}), \Pr(y = 2|\mathbf{x}), \dots, \Pr(y = M|\mathbf{x}))$  that are used to generate it. When  $n$  is finite the histogram fluctuates in shape about the limiting form of histogram, and provided these fluctuations are not too large it turns out that they are ignored by the encoder in the next stage of the Markov chain. Thus, for Markov chains that progressively compress the data as it passes along the chain, it is valid to use the vector of probabilities  $(\Pr(y = 1|\mathbf{x}), \Pr(y = 2|\mathbf{x}), \dots, \Pr(y = M|\mathbf{x}))$  to link the stages of the chain *even though* more generally the normalised histogram (with its fluctuations)  $\frac{\mathbf{n}}{n}$  should be used. This comment assumes that each encoder in the chain ignores (i.e. is invariant to) the degrees of freedom that comprise the fluctuations about the limiting form of each normalised histogram (see [5] for detailed analysis of invariance).

A useful side effect of using the vector of probabilities to link the stages of the chain is that the effect of later stages of the chain is to soften the probabilities used by the earlier stages, which encourages the development of separate information processing channels in intermediate stages of the processing chain.

#### IV. CLASSIFIER COMBINER NETWORKS

The purpose of this section is to show how the encoder network results of the previous section are applicable to problems of combining classifiers, and to give an example to illustrate how this approach works in practice. In Section IV A the problem of combining classifiers is recast as a problem of designing Markov chains of the type discussed in Section II and Section III, and in Section IV B these results are illustrated by a simulation.

##### A. Augmented Input Space

The  $L$  stage Markov chain has the objective function given in Equation 3.7, which is (up to an additive constant) the weighted sum of  $L$  SVQ objective functions ( $D_{SVQ}^l$ ,  $l = 0, 1, \dots, L - 1$ ), corresponding to the individual stages of the chain leading from layer 0 (the original input layer) to layer  $L$  (the final output layer), plus a term  $L(\mathbf{P}^L, \mathbf{Q}^L)$  corresponding to the final output from the chain. The sum of all of these contributions (i.e. the overall objective function) is  $L(\mathbf{P}, \mathbf{Q})$  for the whole chain, whose relevance to the problem of combining classifiers will now be discussed.

In pattern processing the basic problem is to characterise the PDF of the input  $\mathbf{P}^0$ . The simplest approach is to introduce a model  $\mathbf{Q}^0$ , and to optimise it by minimising the number of bits  $L(\mathbf{P}^0, \mathbf{Q}^0)$  needed to specify

the input (as specified by  $\mathbf{P}^0$ ) using a model (as specified by  $\mathbf{Q}^0$ ). This approach is widely used but it has limited flexibility because the model is used to *directly* characterise the input, rather than *indirectly* characterise it by modelling its underlying causes (for instance). More flexibility is possible if the input is augmented to include properties derived from the original input, and then a model is used to characterise the *whole* augmented input (i.e. original input *and* derived properties). If these properties of the input are obtained by applying an  $L$ -stage Markov chain of processing to the original input, then the PDF  $\mathbf{P}^0$  is augmented to become the *joint* PDF  $\mathbf{P}^0 \mathbf{P}^{1|0} \dots \mathbf{P}^{L-1|L-2} \mathbf{P}^{L|L-1}$  and PDF  $\mathbf{Q}^0$  becomes the *joint* PDF  $\mathbf{Q}^0 \mathbf{Q}^{1|0} \dots \mathbf{Q}^{L-1|L-2} \mathbf{Q}^{L|L-1}$ , which is exactly the same as the Markov chain model in Equation 2.7.

In the approach described in this paper the objective function is applied to the *whole* Markov chain (i.e.  $L(\mathbf{P}, \mathbf{Q})$ ) rather than to *only* its input (i.e.  $L(\mathbf{P}^0, \mathbf{Q}^0)$ ) as is usually the case in PDF modelling, because this leads more naturally to many useful self-organising properties.  $L(\mathbf{P}, \mathbf{Q})$  has two types of contribution. The sum of SVQ objective functions ( $D_{SVQ}^l$ ,  $l = 0, 1, \dots, L - 1$ ) acts to encourage the augmented input to encode useful information about the original input. The term  $L(\mathbf{P}^L, \mathbf{Q}^L)$  acts to encourage the formation of a good model  $\mathbf{Q}^L$  of the final encoded output  $\mathbf{P}^L$ . Typically the properties of  $\mathbf{P}^L$  are much simpler than those of the original input  $\mathbf{P}^0$ , so the problem of minimising  $L(\mathbf{P}^L, \mathbf{Q}^L)$  is much simpler than minimising  $L(\mathbf{P}^0, \mathbf{Q}^0)$ . Also optimising the Markov chain reduces to a number of simpler coupled optimisation problems. In the simulation presented in Section IV B the  $L(\mathbf{P}^L, \mathbf{Q}^L)$  term is omitted from the objective function.

There is a lot of flexibility in controlling the type of Markov chain that is used to augment the input space. The properties of each encoder (i.e. each stage of the chain) can be controlled by selecting its code book size  $M$  (i.e. the size of the stage output layer), number of samples  $n$  (i.e. the number of code indices that are independently sampled), and standard deviation  $\sigma$  (i.e. the smaller  $\sigma$  is the larger the stage weighting is in the objective function). The interaction between the stages depends on the choice of these parameters in all of the stages of the chain.

Optimisation of the objective function for the whole Markov chain leads to the self-organising emergence of classifier combiner networks. Each stage of the chain automatically *splits* into a number of separate information processing channels, and the stages of the chain automatically connect together to progressively *merge* the information flowing along these channels. This splitting and merging of information is the origin of the classifier combiner properties of the approach described in this paper.

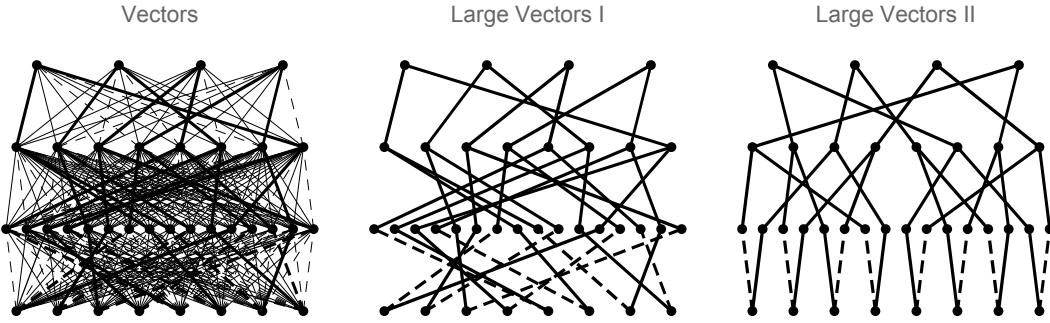


Figure 1: Reconstruction vectors in a trained 3-stage network. The left hand diagram shows all of the reconstruction vectors, using thickness to indicate absolute value and dashing to indicate negative values. The centre diagram shows only the reconstruction vector components with the largest absolute value. The right hand diagram is the same as the centre diagram except that the order of the code indices in layers 1, 2, and 3 have been permuted (together with their connections) to show more clearly the overall network connectivity.

## B. Example of Combining Classifiers

The properties of the objective function  $L(\mathbf{P}, \mathbf{Q})$  are difficult to derive analytically, so a numerical simulation will now be used to illustrate the key property (i.e. splitting and merging of information channels) of the self-organising design of classifier combiner networks.

A hierarchical model will be used to generate the training data. The specific model will be a hierarchically generated vector of phases  $(\phi_1, \phi_2, \phi_3, \phi_4)$  (i.e. "phase screen") that is generated from a single randomly chosen initial phase  $\phi_{1234}$  using the following splitting processes:  $\phi_{1234} \rightarrow (\phi_{12}, \phi_{34})$ ,  $\phi_{12} \rightarrow (\phi_1, \phi_2)$ ,  $\phi_{34} \rightarrow (\phi_3, \phi_4)$ .  $\phi_{12}$  is uniformly distributed in the interval  $[\phi_{1234} - \frac{\pi}{2}, \phi_{1234}]$ ,  $\phi_{34}$  is uniformly distributed in the interval  $[\phi_{1234}, \phi_{1234} + \frac{\pi}{2}]$ ,  $\phi_1$  is uniformly distributed in the interval  $[\phi_{12} - \frac{\pi}{2}, \phi_{12}]$ ,  $\phi_2$  is uniformly distributed in the interval  $[\phi_{12}, \phi_{12} + \frac{\pi}{2}]$ ,  $\phi_3$  is uniformly distributed in the interval  $[\phi_{34} - \frac{\pi}{2}, \phi_{34}]$ , and  $\phi_4$  is uniformly distributed in the interval  $[\phi_{34}, \phi_{34} + \frac{\pi}{2}]$ .

The phase screen  $(\phi_1, \phi_2, \phi_3, \phi_4)$  must be pre-processed in order to convert it into a form that is suitable for input to a chain of encoders. Each phase variable parameterises a circular manifold that needs to be embedded in a 2-dimensional space (i.e.  $\phi \rightarrow (\cos \phi, \sin \phi)$ ). Thus the phase screen is embedded in an 8-dimensional space  $(\phi_1, \phi_2, \phi_3, \phi_4) \rightarrow (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$  where  $(x_{2i-1}, x_{2i}) = (\cos \phi_i, \sin \phi_i)$  for  $i = 1, 2, 3, 4$ .

The phase screen  $(\phi_1, \phi_2, \phi_3, \phi_4)$  is parameterised by 4 phase variables, so it lives on a 4-torus manifold. The embedding  $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$  of the phase screen is thus an embedding of a 4-torus in an 8-dimensional space. Each possible phase screen  $(\phi_1, \phi_2, \phi_3, \phi_4)$  is represented by one point on this 4-torus manifold. The presence of correlations between the phase variables means that the probability density assigned to each point on the 4-torus manifold is non-uniform. The hierarchical structure of these correlations causes a characteristic pattern of probability density variations on the 4-torus manifold,

which then influences the way in which a chain of encoders decides how to encode such data.

Ideally, although this is not guaranteed, a 3-stage chain of encoders will attempt to invert the process whereby the phase screen data was generated. Thus it will first encode the data as 4 separate phase variables (corresponding to  $(\phi_1, \phi_2, \phi_3, \phi_4)$ ), which it will then encode as 2 separate phase variables (corresponding to  $(\phi_{12}, \phi_{34})$ ), which it will then encode as 1 phase variable (corresponding to  $\phi_{1234}$ ). Each successive encoding stage reduces the dimensionality of the encoded representation of the original data. This is achieved by progressively introducing invariances (i.e. removing degrees of freedom from the encoded representation) into the overall encoder. The stage 1 encoder removes no degrees of freedom, the stage 2 encoder removes 2 phase degrees of freedom, and finally the stage 3 encoder removes 1 more phase degree of freedom. This particular sequence of invariances is not guaranteed, but must be discovered by choosing appropriate parameter values (i.e. code book size, number of samples, and weighting in the objective function) for each encoder in the chain. In general, different choices of the encoder parameters will lead to different encoding schemes.

The chosen Markov chain network has layer sizes (4 layers) chosen so as to compress the data as it flows along the chain:  $\mathbf{M} = (8, 16, 8, 4)$ , numbers of samples (3 stages)  $\mathbf{n} = (20, 20, 20)$ , and has stage weightings (3 stages)  $\frac{1}{4\sigma^2} = (1.0, 5.0, 0.1)$ . In Equation 3.4 (for each stage of the network) the posterior probability is parameterised as  $P(y|\mathbf{x}) = \frac{Q(y|\mathbf{x})}{\sum_{y'} Q(y'|\mathbf{x})}$  where  $Q(y|\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}(y).\mathbf{x}-b(y))}$ , and all the components of the weight vectors  $\mathbf{w}(y)$ , biases  $b(y)$ , and reconstruction vectors  $\mathbf{x}'(y)$  are initialised to random numbers uniformly distributed in the interval  $[-0.1, +0.1]$ . The network is trained by a simple gradient descent algorithm using the derivatives of  $D_1 + D_2$  obtained from Equation 3.4. The detailed derivatives are given in [2].

Figure 1 shows the reconstruction vectors in the

trained 3-stage network. The right hand diagram clearly shows that the network has configured itself into a hierarchical processing architecture which processes the input in separate information channels that are progressively fused as the data passes upwards through the network.

## V. CONCLUSIONS

The problem of optimising a classifier combiner network has been recast as a Markov chain optimisation problem [4]. The input to the chain is the original data (possibly derived from multiple sensors) that needs to

be classified, and the stages of the Markov chain progressively process and combine the data as it passes along the chain, as illustrated by a simulation using hierarchically correlated phase screen data. This approach has been shown to be equivalent to multi-stage stochastic vector quantiser theory [1, 2, 5].

## VI. ACKNOWLEDGEMENT

This work was carried out as part of Technology Group 10 of the MoD Corporate Research Programme.

- [1] S P Luttrell, *A discrete firing event analysis of the adaptive cluster expansion network*, Network: Computation in Neural Systems **7** (1996), no. 2, 285–290.
- [2] ———, *Mathematics of neural networks: models, algorithms and applications*, ch. A theory of self-organising neural networks, pp. 240–244, Kluwer, Boston, 1997.
- [3] ———, *A unified theory of density models and auto-encoders*, Preprint NI97039-NNM, Isaac Newton Institute for Mathematical Sciences, Cambridge, 1997.
- [4] ———, *Multiple classifier systems*, Lecture notes in computer science, ch. A self-organising approach to multiple classifier fusion, pp. 319–328, Springer-Verlag, London, 2001.
- [5] ———, *Mathematics in signal processing*, vol. 5, ch. Using stochastic vector quantisers to characterise signal and noise subspaces, pp. 193–204, Oxford University Press, 2002.
- [6] R J McEliece, *The theory of information and coding*, Addison-Wesley, Reading, 1977.
- [7] J Rissanen, *Stochastic complexity in statistical enquiry*, World Scientific, Singapore, 1989.
- [8] C E Shannon, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 3, 379–423.
- [9] ———, *The mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 6, 623–656.



## Guided Tour of Publications \*

Stephen Luttrell

This is an informal walk-through the research context of all of the papers that I wrote between 1981 and 2007. Nearly all of the papers are single-authored and interconnected so this document is self-contained.

### 1981-1982: PhD

Papers Luttrell and Wada [1], Luttrell et al. [2], Luttrell and Wada [3] (written in collaboration with others) and PhD dissertation Luttrell [4] apply quantum chromodynamics to modelling the quark/gluon structure of hadrons seen in deep inelastic scattering experiments. Various higher order effects that modify lower order perturbative results are studied both phenomenologically and theoretically. The operator algebra techniques used in this PhD research unexpectedly turned out to be very useful for the description of Markov chain Monte Carlo methods, as described in my publications from 2005 onwards.

### 1984-1989: Bayesian Super-Resolution and Mutual Information (scattered field)

Report Luttrell and Oliver [5] describes how linear least squares error reconstruction using an appropriately weighted reconstruction space can be used to introduce prior knowledge to enhance the resolution of coherent images. The material in this report was not published because it was quickly overtaken by the full Bayesian treatment described below.

Paper Luttrell [6] describes how Bayes' theorem may be used to derive the solution to linear inverse problems under the assumption of Gaussian probability density functions (PDF). For suitable Bayesian priors this leads to super-resolution where details on a scale shorter than the Rayleigh resolution length become visible.

Book chapter Luttrell and Oliver [7] and report Luttrell [8] present an introduction to the use of prior knowledge in the analysis of SAR images. This was patented in Luttrell and Oliver [9].

Papers Luttrell [10, 11] introduce the principle of mutual information maximisation as a way of optimising the extraction of information from data.

Paper Luttrell and Oliver [12] is on clutter and targets in SAR images. My contribution showed how to use a generalisation of the super-resolution techniques originally developed in Luttrell [6] to analyse images of targets, and showed how to use mutual information (as in Luttrell [10, 11]) to interpret the information processing in terms of information channels. A brief description of this work is in Luttrell and Oliver [13].

Report Luttrell [14] describes an analysis of images of point targets in images produced by the Royal Signals and Radar Establishment (RSRE) SAR. The purpose of this was to accurately calibrate the point spread function (PSF) of the RSRE SAR so that its images could be robustly super-resolved. However, this analysis led me to the conclusion that for the RSRE SAR the PSF was itself a function of the data; in other words the system response was *non-linear*. This totally invalidated the super-resolution theory developed thus far, which assumed that the system response was *linear*, which then triggered a shift of my research away from super-resolution. A side effect of this non-linearity was that the higher moments of the image data were biased away from their naïve values, and thus should *not* have been used in their uncorrected form to deduce the statistics of the underlying clutter model.

Report Luttrell [15] discusses the intimate connection between super-resolution and the analysis of phase information in SAR data.

---

\*Typeset in LATEX on May 6, 2019.

This guided tour was originally written for my curriculum vitae.

Paper Delves et al. [16] and Conference papers Pryde et al. [17, 18] describes the results of collaborative work on a parallel processing implementation of these super-resolution techniques for analysing synthetic aperture radar (SAR) images.

Book chapter Luttrell [19] (delayed for several years in publication) reviews the ideas behind the use of information channels, drawing together ideas from mutual information, super-resolution, and Bayesian inference.

### **1985-1988: Markov Random Fields for Clutter and Texture Modelling**

Report Luttrell [20] is the first in a long series of publications on the use of Markov random fields (MRF) to build probability density function (PDF) models, which describes of how to train MRF models using a generalisation of the Boltzmann machine learning algorithm that I called the Gibbs Machine. Various generalisations were discussed such as the idea of enhancing an MRF model by using these learning techniques to attach a “brain graft” to the MRF model to patch it up wherever it did not give a good enough approximation to the data.

Paper Luttrell [21] describes how to do Monte Carlo sampling of arbitrarily complicated MRFs by bit-flipping operations that could be easily implemented in hardware.

Report Luttrell [22] uses elementary PDF methods to measure radar sensitivity in a way that is invariant w.r.t. the receiver law.

Paper Luttrell [23] presents a careful analysis of the use of MRF models for texture modelling, using arguments based on sufficient statistics to understand how the texture information is spread out amongst the various measured statistics. This is conceptually similar to the use of information channels discussed in Luttrell and Oliver [12]. Contact with image processing techniques is made via the grey level co-occurrence matrix method and the WISARD  $n$ -tuple processing network, both of which are special cases of this MRF analysis.

Conference papers Luttrell [24, 25, 26, 27] develop various aspects of the use of MRFs for texture and clutter modelling.

Paper Luttrell [28] shows how the maximum entropy method can be used to pick good statistics to measure in textured images (as introduced in Luttrell [23]), so that an efficient MRF texture model can be built. An efficient algorithm (which does *not* require the fixing of Lagrange multipliers, unlike other approaches) is given for deciding what new statistics to add to the MRF model by comparing *synthetic* textures generated using the current MRF model with the *real* texture to be modelled. As a corollary, a generalisation of the original Boltzmann machine learning algorithm is given for arbitrary MRFs.

Conference paper Luttrell [29] compares and contrasts the MRF approach (as in Luttrell [28]) and a proposed hierarchical “cluster-decomposition” approach to modelling PDFs. The generalised form of the Boltzmann machine learning algorithm is presented in detail. The advantages of the cluster decomposition approach (particularly that it does *not* use Monte Carlo simulations) are explained.

Report Luttrell [30] analyses the optimisation of hidden Markov models (HMM) using Gibbs Machine methods.

Conference paper Luttrell [31] summarises the MRF approach in the context of modelling texture in radar images.

### **1989-1991: Bayesian Super-Resolution (scatterers themselves)**

Paper Luttrell [32] extends the earlier Bayesian super-resolution method (see Luttrell [6]) by pointing out the fact that the inverse problem to solve should be the reconstruction of the object *scatterers* that produce the object field, and *not* the reconstruction of the object *field* itself. In report Luttrell [33] and paper Luttrell [34] this work is greatly extended by using the expectation-maximisation (EM) method to derive an iterative object reconstruction algorithm from first principles. This allows both super-resolution and auto-focusing to be handled within the same framework. This work was reviewed in Luttrell [35]. Report Luttrell [36] shows how the iterated EM method can be applied to reconstructing the object scatterers under the assumption that they produce a K-distributed object field.

**1988-1989: Hierarchical Self-Organising Maps**

Conference paper Luttrell [37] describes a hierarchical generalisation of the standard Kohonen self-organising map (SOM) network, and introduces the idea of the “growing grid” method of training SOMs. Conference paper Luttrell [38]) and paper Luttrell [39] train a tree of linked SOMs to encode SAR images. In paper Luttrell [40] this approach is described in more detail.

**1986-1990: Digital Receiver**

Report Luttrell and Pritchard [41] and paper Luttrell and Pritchard [42] describe the analysis, design and testing of a finite state machine for demodulating signals. My contribution was the analysis. This is essentially an early type of software radio receiver. This was patented in Luttrell and Pritchard [43].

**1989-1992: Distortion Minimising Encoders (single SOM)**

Conference papers Luttrell [44, 45] and paper Luttrell [46] present a novel derivation of SOM networks, based on a generalisation of the standard Linde-Buzo-Gray (LBG) vector quantisation algorithm to account for distortion on the communication channel, which approximates the standard Kohonen SOM algorithm as a special case (the distribution of distortions corresponds to the topographic neighbourhood function). The biggest advantage of the approach used is that it is based on minimisation of an objective function (unlike Kohonen’s SOM), which thus allows many properties to be theoretically derived. The density of code vectors in this type of SOM (for 1-dimensional data) is derived in report Luttrell [47] and paper Luttrell [48], where the density is shown to be the *same* as in a standard vector quantiser (unlike the standard Kohonen SOM which leads to a *different* density which inconveniently depends on the choice of topographic neighbourhood function). A generalisation of this result to data of arbitrary dimensionality was given in report Luttrell [49]. Report Luttrell [50] shows how to use a vector quantiser to encode data under the assumption that its noise is K-distributed.

**1988-1994: Lateral Mutual Information Maximising Hierarchical Encoders**

Reports Luttrell [51, 52] and in the arXiv papers Luttrell [53, 54] show how optimisation of the hierarchical “cluster decomposition” introduced in conference paper Luttrell [29] can be used to detect anomalies in textured images. Conference paper Luttrell [55] presents a detailed analysis of this optimisation process, where the objective function used is the relative entropy between the model PDF and the true PDF, which is equivalent to the *sum* of the (lateral) mutual informations between various nodes in each layer of the network. This approach has been patented Luttrell [56].

**1990-1992: Distortion Minimising Encoders (coarse-grain parallel SOMs)**

Report Luttrell [57] and conference paper Luttrell [58] use a communication channel model of SOMs (see Luttrell [46]) to analyse networks of connected SOMs, and this led to paper Luttrell [59]. The model used is one in which two communication channels mutually interfere thus causing distortion that is correlated between the channels. This distortion defines the topographic neighbourhood functions that are used in the SOMs (that are the channel encoders) in the first place. The optimisation of the SOMs is influenced by mutual interactions between their outputs, hence the description of this type of training as “self-supervised”.

### **1988-1994: Some Consolidation**

Report Luttrell [60] and conference paper Luttrell [61] analyse adaptive  $n$ -tuple networks (e.g. WISARD) by using a relative entropy objective function to optimise a PDF model. Both supervised and unsupervised methods of training such networks emerge naturally from this analysis.

Report Luttrell [62] and conference paper Luttrell [63] review the Bayesian approach to training and using neural networks. Various models are discussed including the “adaptive cluster expansion” model originally introduced in conference paper Luttrell [29].

Conference paper Luttrell [64] presents a rigorous Bayesian analysis of the “adaptive cluster expansion” model.

### **1992-1994: Partitioned Mixture Distributions**

Report Luttrell [65] and conference papers [66, 67] describe a generalisation (the *partitioned* mixture distribution, or PMD) of the standard mixture distribution used to model PDFs, and this led to paper Luttrell [68]. In image processing (the statistical properties of) each correlation area of an image can be modelled with a mixture distribution, and it would be convenient if all of these models could be collected together in a single translation invariant architecture. A nice solution to this problem is to build each mixture distribution using components selected from a common pool of mixture components, where each correlation area has a complete repertoire of components needed for its analysis. Mixture distributions that see overlapping areas of image have a lot of overlap in the components they use. A PMD can be trained using a generalisation of the EM method for training standard mixture distributions. Trained PMDs have many of the properties that are observed in the low-level visual cortex, such as a complete repertoire of processing machinery for each local patch of image.

### **1990-1994: Folded Markov Chains**

Report Luttrell [69] introduces the folded Markov chain (FMC) technique, and paper Luttrell [70] uses it to give a Bayesian analysis of the SOM model originally described in Luttrell [46]. In an FMC information is passed along a Markov chain, and then Bayes’ theorem is used to pass (virtually) the information back along the chain in the opposite direction until it reaches the start again. This final reconstruction of the input to the chain is compared with the original input, and the transitions in the Markov chain are adjusted to minimise the Euclidean reconstruction distortion (on average). The tight constraints that Bayes’ theorem imposes on the various probabilities make it possible to derive a training algorithm that reduces to the algorithm in Luttrell [46] when the encoder is *deterministic*. This FMC analysis unifies all my earlier work on networks of SOMs, and by allowing *probabilistic* encoders to be handled in the same framework, it sets the scene for future work.

### **1993-1995: Some Consolidation**

Reports Luttrell [71, 72, 73] describe various ways in which mixture distributions can be used in the analysis of images.

Conference paper Luttrell [74] and report Luttrell [75] shows the application of SOM techniques to the analysis of data (e.g. range profiles of radar returns from ships) that lies on a manifold with a circular topology. Prior knowledge of the topology is used to define an appropriate topographic neighbourhood function for use in the SOM.

### **1994-1997: Unification of FMC and PMD**

I celebrate “FMCPMD Day” on 17<sup>th</sup> June each year – year zero was in 1994 – in recognition of the day that I finally realised that the correct objective function for optimising a PMD should focus on minimising the reconstruction distortion rather than optimising the PDF. More generally, distortion-minimisation rather than PDF-optimisation

is the correct (and much simpler) way forward, and the reason it took me so long to realise this was because the mathematics of PDF-optimisation is so much prettier than the mathematics of distortion-minimisation.

Book chapter Luttrell [76] (delayed for several years in publication), reports Luttrell [77, 78, 79] and arXiv paper Luttrell [80] combine the FMC approach (introduced in Luttrell [70]) with the PMD approach (introduced in Luttrell [68]) to create an encoder structure that is appropriate for the analysis of images, where the form of the PMD posterior probability allows the encoder to break into local pieces that each encode a local patch of the image. This is then applied to the analysis of images derived from multiple sources. In the case of *pairs* of images the network learns a structure that closely resembles the dominance stripes and orientation maps observed in the visual cortex. The important point is that you can obtain these “biological” results as a consequence of the general properties of encoders – to emphasise this the acronym VICON (VIIsual COortex Network) was coined in Luttrell [79, 80].

Report Luttrell [81] and conference papers [82, 83] present a detailed analysis of some of the properties of PMDs (introduced in Luttrell [68]). A dynamical PMD (which has the same relationship to a static PMD that a hidden Markov model has to a mixture distribution) is analysed in detail, and is successfully applied to the problem of tracking a weak target in clutter. A first order perturbation analysis reveals the low-order properties of PMDs and allows their behaviour to be related to various neural networks.

arXiv paper Luttrell [84], report Luttrell [85], and conference paper Luttrell [86] apply the FMC approach (introduced in Luttrell [70]) to optimising encoders that output several statistically independent (given the input) codes. Various analytic optimum solutions are obtained. However, these solutions are *less* useful than those that are obtained if the optimisation is constrained in various ways, which is the focus of all subsequent papers in this area.

### **1997: “State of the Nation”**

Report Luttrell and Webber [87] summarises all publications produced during the period 1994-1997.

### **1997: Isaac Newton Neural Networks Programme**

Report Luttrell [88] and arXiv paper Luttrell [89] describes the work I did at the Neural Networks research programme at the Isaac Newton Institute. I analysed the optimisation of an objective function depending on the *joint* PDF of the state of a multi-layer network, and showed that this unified all of my work on the optimisation of hierarchical encoders.

arXiv paper [90] summarises various properties of this method of optimising a network.

### **1994-1997: Distortion Minimising Encoders (fine-grain parallel)**

Book chapters Luttrell [91, 92] and paper Luttrell [93] give the generalisation of the FMC approach of Luttrell [70] to the case where multiple codes are output by a *probabilistic* encoder for each of its input vectors. Up to this point in my work the encoders have each output a *single* code, and thus each acts as a winner-take-all network. The advantage of allowing the possibility of *multiple* codes is that the encoded information can be split across more than one channel. This is conceptually similar to the use of information channels discussed in Luttrell and Oliver [12]. Further, if these codes are chosen stochastically, then by optimising the FMC (minimum Euclidean distortion) the network can decide for itself how many information channels it needs to use. This freedom allows the network to optimise its own architecture. In Luttrell [93] a simple application of these ideas is analysed in detail, where various (supposedly) optimal choices of information channel are compared with each other, and the conditions under which each type of solution is optimal are established. An important cross-disciplinary result is that the above extension to *multiple* codes allows contact to be made with neural network models based on discrete firing events, where the recent history of which neurons have fired corresponds to the set of codes that are currently active.

Paper Luttrell [94] applies the multiple sampling ideas of Luttrell [93] to the hierarchical vector quantisation network Luttrell [40]. Although the hierarchical network of Luttrell [40] has its structure manually defined, rather than learnt via optimisation (as *could* be done using Luttrell [93]), the earlier ideas in [40] automatically fit into the framework of the later theory in Luttrell [93].

### Change in Publication Style

From here on there are currently no further journal papers, because delays in publication caused by the refereeing process had become excessive. However, the research continues to be published in conference papers and technical reports.

#### **1998: Some Consolidation**

Report Luttrell [95] describes various ways of training self-organising encoder networks by modelling their output as the posterior probability over samples (or “firing events”) drawn from a stochastic code book. The different training schemes correspond to different conditions under which the samples are drawn.

#### **1998-1999: Distortion Minimising Encoders (analytic optimisation)**

Conference paper Luttrell [96] shows how *analytic* optimisation of a stochastic encoder (an FMC with multiple output codes) leads to optimal encoders being described by *piecewise linear* posterior probabilities. This emerges because the reconstruction is a *linear* superposition of contributions, and the objective function is a *Euclidean* distortion. This piecewise property also holds in networks of linked stochastic encoders. This piecewise linearity will prove to be very useful for deriving analytic solutions to various problems.

arXiv paper Luttrell [97], book chapter Luttrell [98], and report Luttrell [99] use the piecewise linear property of optimal stochastic encoders (see Luttrell [96]) to derive optimal encoding schemes for circular and toroidal manifolds. The results derived in this paper were obtained analytically by using *Mathematica* to manipulate the various cumbersome piecewise linear expressions leading to compact results. The circle and the torus are simple curved manifolds that serve as models for the more complicated curved manifolds that arise in signal and image processing. For the toroidal manifold *two* types of optimal encoder emerge depending on the size of the encoder and the number of stochastic codes that it is allowed to output. On the one hand a *joint* encoder is optimal if there are more than a certain minimum number of codes to choose from; this corresponds to an encoder that has so many resources that it can simply partition the torus into localised code cells. On the other hand a *factorial* encoder emerges if the number of codes to choose from is severely reduced *and* the number of codes it is allowed to output is sufficiently large; this corresponds to an encoder that is starved of resources but which is allowed to have several trials at outputting a code, and which therefore chooses to use anisotropic code cells to slice the torus into localised code cells defined by the regions of intersection of *pairs* of anisotropic code cells. The automatic emergence of factorial coding is a crucial property in a network that aims to discover *for itself* what information channels to use when processing data.

#### **1999: Some Consolidation**

Reports [100, 101] motivate the use of self-organising networks for discovering and extracting information in complicated data, with an emphasis on their potential application to data fusion. Report Luttrell [102] is a user’s guide to stochastic encoders, and contains many simple examples of their use. arXiv paper Luttrell [103] is a comprehensive summary of the theory and many numerical simulations of stochastic encoders, and arXiv paper Luttrell [104] contains a small subset of these results.

#### **2000: “State of the Nation 2”**

Report [105] summarises all publications produced during the period 1997-2000.

**2000-2003: Distortion Minimising Encoders (jammer suppression)**

Conference paper Luttrell [106] applies a stochastic encoder to the problem of separating a signal from a jammer. Because the jammer is much stronger than the signal the optimisation concentrates on encoding the jammer alone. This allows the trained encoder to separate the jammer and signal subspaces (in this case these are *curved* subspaces), which allows the signal to be cleanly detected. This material is also covered in report Luttrell [107], conference papers Luttrell [108, 109], arXiv papers (extended versions of the preceding conference papers) Luttrell [110, 111], and book chapters [112, 113].

**2001: Distortion Minimising Encoders (classifier fusion)**

Conference paper Luttrell [114] and workshop paper [115] demonstrate some of the key properties of stochastic encoders. One of these is their ability to learn factorial codes in which a code book breaks into separate smaller code books, each of which encodes only one part of the input. Another is the natural way in which supervision can be introduced to steer the code books towards coding their inputs in particular ways. This is all presented in the context of multiple classifier fusion, where the processing is split across multiple information channels and then fused together again at the end.

**2001-2004: Internal Reports (part 1)**

Report Luttrell [116] describes how to use a stochastic encoder to suppress sea clutter in coherent images, which allows targets that are masked by the clutter to be revealed. The approach used is essentially the same as for separating a signal from a jammer (as in conference paper Luttrell [106]).

Report Luttrell [117] compares and contrasts the use of encoders and the use of PDFs for modelling (and suppressing) sea clutter in images.

Reports [118, 119] summarise all of my research results on self-organising networks.

Reports Luttrell [120, 121] describe various interesting and useful properties of the self-organising network known as ACEnet (Adaptive Cluster Expansion network).

Reports Luttrell [122, 123, 124] describe how to apply self-organising networks to the problem of combining the outputs of multiple classifiers.

**2003: Isaac Newton Neural Networks Programme (published, at last!)**

Conference paper Luttrell [125] introduces a new way of understanding networks of linked stochastic encoders. These ideas were originally described in report Luttrell [88]. Rather than modelling each encoder as an FMC (see Luttrell [70] and its generalisation to multiple codes in Luttrell [93]) and building up the overall network objective function as a *sum* of local contributions, the *whole* network of linked encoders is deemed to have a *single* objective function which compares the joint PDF of the whole network under two different modelling assumptions. For a chain-like network the two PDF models are built from conditional probabilities acting in opposite directions along the chain, and it is similar to Dayan and Hinton's Helmholtz machine *except that* the aim here is to optimise the *joint* network probability rather than only the *marginal* input probability. This approach is then shown to reduce to the FMC approach wherever that had been used earlier. An application of these ideas to training a network on hierarchically correlated data is presented, and the encoders (the bottom-up PDF models) automatically split into smaller encoders in precisely the way that is expected (i.e. process the most uncorrelated pieces separately, then progressively fuse the results until only the most correlated piece is left). This approach is thus capable of making *structural* changes to the network as it is trained, where it creates information channels for doing the lower-level processing, and then progressively fuses these channels to do the higher-level processing.

**2003-2004: Distortion Minimising Encoders (multiple correlation scales)**

arXiv paper Luttrell [126] describes the use of self-organising stochastic encoders for learning the structure of data manifolds. The main aim of this paper was to demonstrate the automatic separation of correlated components in the data, and their subsequent fusion whilst preserving only their dominant degree(s) of freedom, as was described in Luttrell [125]. This is a key paper that opens up a very fruitful area of research.

**2005: Discrete Network Dynamics**

arXiv paper [127] describes the use of operator algebra techniques to reformulate MCMC algorithms, so that the algorithms can be manipulated by purely algebraic methods, which would have applications in adaptive networks. This is a key paper that opens up a very fruitful area of research.

**2005-2007: Internal Reports (part 2)**

Report Luttrell [128] comprehensively reviews MRFs and SONs from the perspective of adaptive filters and recurrent networks, and introduces an operator algebra to describe the associated Markov chain Monte Carlo (MCMC) algorithms.

Reports Luttrell [129, 130] apply the operator algebra methods of [127, 128] to recurrent networks, dealing with both the small-occupancy (i.e. dominated by quantisation effects) and large-occupancy (i.e. quantisation effects negligible) cases.

Report Luttrell [131] describes a proposed application of recurrent SONs to the problem of language identification.

Report Luttrell [132] introduces symbolic algebra techniques for manipulating the operator algebra of MCMC algorithms.

Report Luttrell [133] presents a *Mathematica* implementation of recurrent SONs.

The appendix of report [134] presents some ideas on how to use operator algebra methods and self-organising networks to learn the structure of networks.

- [1] S P Luttrell and S Wada. Heavy quark production in electromagnetic deep inelastic scattering and qcd analysis of isosinglet moments. *Nuclear Physics B*, 182(3):381–396, 1981.
- [2] S P Luttrell, S Wada, and B R Webber. The Wilson coefficient functions of four quark operators and the four quark process in deep inelastic scattering. *Nuclear Physics B*, 188(2):219–232, 1981.
- [3] S P Luttrell and S Wada. The current product expansion of the two quark process at the twist four level. *Nuclear Physics B*, 197(2):290–306, 1982.
- [4] S P Luttrell. *Power law corrections to hadronic structure functions*. PhD thesis, Cambridge University, 1982.
- [5] S P Luttrell and C J Oliver. Resolution enhancement in coherent images by the introduction of prior knowledge. Memorandum 3697, Royal Signals and Radar Establishment, Malvern, 1984.
- [6] S P Luttrell. Prior knowledge and object reconstruction using the best linear estimate technique. *Optica Acta*, 32(6): 703–716, 1985.
- [7] S P Luttrell and C J Oliver. *RSRE Research Review*, chapter Prior knowledge in synthetic aperture radar (SAR) processing, pages 73–77. RSRE, 1985.
- [8] S P Luttrell. A super-resolution model for synthetic aperture radar. Memorandum 3785, Royal Signals and Radar Establishment, Malvern, 1985.
- [9] S P Luttrell and C J Oliver. Super resolution imaging system. Patent GB 2173663 B, 1985.
- [10] S P Luttrell. A new method of sample optimisation. *Optica Acta*, 32(3):255–257, 1985.
- [11] S P Luttrell. The use of transinformation in the design of data sampling schemes for inverse problems. *Inverse Problems*, 1(3):199–218, 1985.
- [12] S P Luttrell and C J Oliver. Prior knowledge in synthetic aperture radar processing. *Journal of Physics D: Applied Physics*, 19(3):333–356, 1986.

- [13] S P Luttrell and C J Oliver. The role of prior knowledge in coherent image processing. *Philosophical Transactions of the Royal Society A*, 324(1579):397–407, 1988.
- [14] S P Luttrell. The complex point spread function of the RSRE SAR. Memorandum 4079, Royal Signals and Radar Establishment, Malvern, 1987.
- [15] S P Luttrell. The relationship between super-resolution and phase imaging of SAR data. Research Note BS1/41, Royal Signals and Radar Establishment, Malvern, 1987.
- [16] L M Delves, G Pryde, and S P Luttrell. A super-resolution algorithm for SAR images. *Inverse Problems*, 4(3):681–703, 1988.
- [17] G C Pryde, L M Delves, and S P Luttrell. The performance of a parallel super-resolution algorithm for synthetic aperture radar images. In *Proceedings of IMA conference on mathematics in signal processing*, pages 739–747, Oxford, 1988. University Press.
- [18] G C Pryde, L M Delves, and S P Luttrell. A comparative study of the AMT DAP and of Transputer array architectures for the super-resolution of synthetic aperture radar images. In *Proceedings of conference on parallel processing*, pages 340–350, Cambridge, 1988. University Press.
- [19] S P Luttrell. *Mathematics in remote sensing*, chapter Inference theory, pages 205–222. The Institute of Mathematics and its Applications Conference Series. Clarendon Press, 1989.
- [20] S P Luttrell. The implications of Boltzmann-type machines for SAR data processing: a preliminary survey. Memorandum 3815, Royal Signals and Radar Establishment, Malvern, 1985.
- [21] S P Luttrell. An optimisation of the Metropolis algorithm for multibit Markov random fields. *Inverse Problems*, 2(2):L15–L17, 1986.
- [22] S P Luttrell. A proposal for a new method of calibrating radar sensitivity. Research Note BS1/37, Royal Signals and Radar Establishment, Malvern, 1987.
- [23] S P Luttrell. The use of Markov random field models to derive sampling schemes for inverse texture problems. *Inverse Problems*, 3(2):289–300, 1987.
- [24] S P Luttrell. Markov random fields: a strategy for clutter modelling. In *Proceedings of AGARD conference on scattering and propagation in random media*, pages 7.1–7.8, Rome, 1987. AGARD.
- [25] S P Luttrell. The use of Markov random field models in sampling scheme design. In *Proceedings of SPIE international symposium on inverse problems in optics*, New York, 1987. SPIE.
- [26] S P Luttrell. *Radar 87*, chapter Designing Markov random field structures for clutter modelling, pages 222–226. IEE, London, 1987.
- [27] S P Luttrell. Markov random field image models of targets and clutter. In *Proceedings of US/UK workshop on parallel processing*, pages 771–793, Shrivenham, 1987.
- [28] S P Luttrell. A maximum entropy approach to sampling function design. *Inverse Problems*, 4(3):829–841, 1988.
- [29] S P Luttrell. *Maximum entropy and Bayesian methods*, chapter The use of Bayesian and entropic methods in neural network theory, pages 363–370. Kluwer, Dordrecht, 1989.
- [30] S P Luttrell. The Gibbs Machine applied to hidden Markov model problems. Part 1: Basic theory. Research Note SP4/99, Royal Signals and Radar Establishment, Malvern, 1989.
- [31] S P Luttrell. An adaptive Bayesian network for texture modelling. In *Proceedings of the IEE seminar on texture analysis in radar and sonar*, pages 6.1–6.10, London, 1993. IEE.
- [32] S P Luttrell. The inverse cross section problem for complex data. *Inverse Problems*, 5(1):35–50, 1989.
- [33] S P Luttrell. A Bayesian derivation of an iterative autofocus/super-resolution algorithm. Memorandum 4337, Royal Signals and Radar Establishment, Malvern, 1989.
- [34] S P Luttrell. A Bayesian derivation of an iterative autofocus/super-resolution algorithm. *Inverse Problems*, 6(6):975–996, 1990.
- [35] S P Luttrell. The theory of Bayesian super-resolution of coherent images: a review. *International Journal of Remote Sensing*, 12(2):303–314, 1991.
- [36] S P Luttrell. An EM approach to iterative Bayesian super-resolution with applications to K-distributed data. Research Note SP4/102, Royal Signals and Radar Establishment, Malvern, 1989.
- [37] S P Luttrell. Self organising multilayer topographic mappings. In *Proceedings of IEEE conference on neural networks*, pages 93–100, San Diego, 1988. IEEE.
- [38] S P Luttrell. Image compression using a neural network. In *Proceedings of IGARSS conference on remote sensing*, pages 1231–1238, Edinburgh, 1988. ESA.
- [39] S P Luttrell. Image compression using a multilayer neural network. *Pattern Recognition Letters*, 10(1):1–7, 1989.
- [40] S P Luttrell. Hierarchical vector quantisation. *Proceedings of the IEE I*, 136(6):405–413, 1989.
- [41] S P Luttrell and J A S Pritchard. A compact digital communications system - Part 1: rapid carrier acquisition. Memorandum 3925, Royal Signals and Radar Establishment, Malvern, 1986.
- [42] S P Luttrell and J A S Pritchard. Rapid acquisition of low signal-to-noise carriers. *IEE Proceedings I*, 136(1):100–108, 1989.
- [43] S P Luttrell and J A Pritchard. Signal transforming device. Patent GB 2190221 B, 1987.
- [44] S P Luttrell. Hierarchical self-organising networks. In *Proceedings of IEE Conference on Artificial Neural Networks*, pages 2–6, London, 1989. IEE.
- [45] S P Luttrell. Self-organisation: a derivation from first principles of a class of learning algorithms. In *Proceedings of IJCNN international joint conference on neural networks*, pages 495–498, Washington DC, 1989. IEEE.
- [46] S P Luttrell. Derivation of a class of training algorithms. *IEEE Transactions on Neural Networks*, 1(2):229–232, 1990.

- [47] S P Luttrell. Asymptotic code vector density in topographic vector quantisers. Memorandum 4392, Royal Signals and Radar Establishment, 1990.
- [48] S P Luttrell. Code vector density in topographic mappings: scalar case. *IEEE Transactions on Neural Networks*, 2(4): 427–436, 1991.
- [49] S P Luttrell. Code vector density in topographic mappings. Technical Report 4669, Defence Research Agency, Malvern, 1992.
- [50] S P Luttrell. Vector quantisation of K-distributed data. Research Note SP4/110, Royal Signals and Radar Establishment, Malvern, 1990.
- [51] S P Luttrell. Bayesian inference on a tree. Research Note SP4/109, Royal Signals and Radar Establishment, Malvern, 1990.
- [52] S P Luttrell. A trainable texture anomaly detector using the adaptive cluster expansion (ACE) method. Memorandum 4437, Royal Signals and Radar Establishment, Malvern, 1990.
- [53] S P Luttrell. Adaptive Cluster Expansion (ACE): a hierarchical Bayesian network. 1991. URL <http://arxiv.org/abs/cs/0410020>.
- [54] S P Luttrell. Adaptive Cluster Expansion (ACE): a multilayer network for estimating probability density functions. 1991. URL <http://arxiv.org/abs/1012.3656>.
- [55] S P Luttrell. A hierarchical network for clutter and texture modelling. In *Proceedings of SPIE conference on adaptive signal processing*, pages 518–528, San Diego, 1991. SPIE.
- [56] S P Luttrell. Anomaly detector. Patent GB 2253697 B, 1992.
- [57] S P Luttrell. Self-supervision in multilayer adaptive networks. Memorandum 4467, Royal Signals and Radar Establishment, Malvern, 1991.
- [58] S P Luttrell. Self-supervised training of hierarchical vector quantisers. In *Proceedings of IEE conference on artificial neural networks*, pages 5–9, Bournemouth, 1991. IEE.
- [59] S P Luttrell. Self-supervised adaptive networks. *Proceedings of the IEE F*, 139(6):371–377, 1992.
- [60] S P Luttrell. A maximum entropy interpretation of random access memory network computations. Research Note SP4/137, Royal Signals and Radar Establishment, Malvern, 1990.
- [61] S P Luttrell. Gibbs distribution theory of adaptive n-tuple networks. In I Aleksander and J Taylor, editors, *Proceedings of IEE conference on artificial neural networks*, pages 313–316, Brighton, 1992. IEE, North-Holland.
- [62] S P Luttrell. Error measures in adaptive networks. Research Note SP4/111, Royal Signals and Radar Establishment, Malvern, 1990.
- [63] S P Luttrell. Adaptive Bayesian networks. In *Proceedings of SPIE conference on adaptive signal processing*, pages 140–151, Orlando, 1992. SPIE.
- [64] S P Luttrell. *Maximum entropy and Bayesian methods*, chapter The cluster expansion: a hierarchical density model, pages 269–278. Kluwer, Dordrecht, 1995.
- [65] S P Luttrell. Partitioned mixture distributions: an introduction. Memorandum 4671, Defence Research Agency, 1992.
- [66] S P Luttrell. An adaptive Bayesian network for low-level image processing. In *Proceedings of IEE conference on artificial neural networks*, pages 61–65, Brighton, 1993. IEE.
- [67] S P Luttrell. *Maximum entropy and Bayesian methods*, chapter The partitioned mixture distribution: multiple overlapping density models, pages 279–286. Kluwer, Dordrecht, 1995.
- [68] S P Luttrell. Partitioned mixture distribution: an adaptive Bayesian network for low-level image processing. *IEE Proceedings on Vision, Image, and Signal Processing*, 141(4):251–260, 1994.
- [69] S P Luttrell. The Markov chain theory of vector quantisers. Technical Report 4742, Defence Research Agency, Malvern, 1993.
- [70] S P Luttrell. A Bayesian analysis of self-organising maps. *Neural Computation*, 6(5):767–794, 1994.
- [71] S P Luttrell. The detection of linear anomalies in images. Research Note CSE1/232, Defence Research Agency, Malvern, 1992.
- [72] S P Luttrell. The application of mixture distributions to the detection of linear anomalies in cluttered images: a preliminary survey. Technical Report 4787, Defence Research Agency, Malvern, 1993.
- [73] S P Luttrell. The use of mixture distributions to model the density of speckled data. Technical Report DRA/CIS/CBC3/TR94002, Defence Research Agency, 1994.
- [74] S P Luttrell. Using self-organising maps to classify radar range profiles. In *Proceedings of IEE conference on artificial neural networks*, pages 335–340, Cambridge, 1995. IEE.
- [75] S P Luttrell. Range profile classification using topographic mappings. Technical Report DRA/CIS(SE1)/651/11/RP/2, Defence Research Agency, Malvern, 1995.
- [76] S P Luttrell. *Information theory and the brain*, chapter The emergence of dominance stripes and orientation maps in a network of firing neurons, pages 101–121. Cambridge University Press, 2000.
- [77] S P Luttrell. A self-organising network for processing data from multiple sensors. Technical Report DRA/CIS(SE1)/651/11/RP/1, Defence Research Agency, Malvern, 1995.
- [78] S P Luttrell. A componential self-organising neural network. Technical Report DRA/CIS(SE1)/651/11/RP/3, Defence Research Agency, Malvern, 1995.
- [79] S P Luttrell. A self-organising visual cortex network (VICON) for processing data from two imaging sensors. Technical Report DRA/CIS(SE1)/651/FUN/STIT/RP/3, Defence Research Agency, Malvern, 1996.
- [80] S P Luttrell. The development of dominance stripes and orientation maps in a self-organising visual cortex network (VICON). 1996. URL <http://arxiv.org/abs/1012.3724>.

- [81] S P Luttrell. Dynamical partitioned mixture distributions: an introduction. Technical Report DRA/CIS(SE1)/651/11/RP/4, Defence Research Agency, Malvern, 1995.
- [82] S P Luttrell. Partitioned mixture distributions: the dynamical case. In *Proceedings of IEE international conference on artificial neural networks*, pages 59–63, Cambridge, 1997. IEE.
- [83] S P Luttrell. Partitioned mixture distributions: a first order perturbation analysis. In *Proceedings of IEE international conference on artificial neural networks*, pages 280–284, Cambridge, 1997. IEE.
- [84] S P Luttrell. A self-organising neural network for processing data from multiple sensors. 1995. URL <http://arxiv.org/abs/1012.4173>.
- [85] S P Luttrell. Optimal posterior probabilities for self-organised neural networks. Technical Report DRA/CIS(SE1)/651/FUN/STIT/RP/1, Defence Research Agency, Malvern, 1996.
- [86] S P Luttrell. Optimal response functions in a network of discretely firing neurons. In *Proceedings of IEE conference on artificial neural networks*, pages 216–220, Cambridge, 1997. IEE.
- [87] S P Luttrell and C J S Webber. Spatio-temporal inference theory: final report (1 April 1994 - 31 March 1997). Technical Report DRA/CIS(SE1)/651/FUN/STIT/RP/4, Defence Research Agency, Malvern, 1997.
- [88] S P Luttrell. A unified theory of density models and auto-encoders. Technical Report DERA/CIS/CIS5/TR97303 (also Isaac Newton Institute for Mathematical Sciences Preprint NI97039-NNM), Defence Evaluation and Research Agency, Malvern, 1997.
- [89] S P Luttrell. Modelling the probability density of Markov sources. 1998. URL <http://arxiv.org/abs/cs/0607019>.
- [90] S P Luttrell. Some theoretical properties of a network of discretely firing neurons. 1998. URL <http://arxiv.org/abs/1505.00444>.
- [91] S P Luttrell. *Handbook of neural computation*, chapter Designing analysable networks, pages B5.3:1–B5.3:8. IOP and OUP, Oxford, 1995.
- [92] S P Luttrell. *Mathematics of neural networks: models, algorithms and applications*, chapter A theory of self-organising neural networks, pages 240–244. Kluwer, Boston, 1997.
- [93] S P Luttrell. Self-organisation of multiple winner-take-all neural networks. *Connection Science*, 9(1):11–30, 1997.
- [94] S P Luttrell. A discrete firing event analysis of the adaptive cluster expansion network. *Network: Computation in Neural Systems*, 7(2):285–290, 1996.
- [95] S P Luttrell. A Bayesian analysis of complementary approaches to training self-organising stochastic networks. Technical Report DERA/CIS/CIS5/TR97165, Defence Evaluation and Research Agency, Malvern, 1998.
- [96] S P Luttrell. An adaptive network for encoding data using piecewise linear functions. In *Proceedings of international conference on artificial neural networks*, pages 198–203, Edinburgh, 1999. IEE.
- [97] S P Luttrell. Self-organised factorial encoding of a toroidal manifold. 1998. URL <http://arxiv.org/abs/cs/0410036>.
- [98] S P Luttrell. *Combining artificial neural nets: Ensemble and modular multi-net systems*, chapter Self-organised modular neural networks for encoding data, pages 235–263. Perspectives in Neural Computing. Springer-Verlag, London, 1999.
- [99] S P Luttrell. Encoding data from a curved manifold. Technical Report DERA/S&P/SPI/TR98116, Defence Evaluation and Research Agency, Malvern, 1999.
- [100] S P Luttrell and C J S Webber. Using self-organising neural networks to decompose complex tasks into simpler sub-tasks. Technical Report DERA/S&P/SPI/TR980086, Defence Evaluation and Research Agency, Malvern, 1998.
- [101] S P Luttrell and C J S Webber. Using self-organising neural networks to discover structure in data. Technical Report DERA/S&P/SPI/TR990564, Defence Evaluation and Research Agency, Malvern, 1999.
- [102] S P Luttrell. A user's guide to stochastic encoder/decoders. Technical Report DERA/S&P/SPI/TR990290, Defence Evaluation and Research Agency, Malvern, 1999.
- [103] S P Luttrell. Self-organising stochastic encoders. 1999. URL <http://arxiv.org/abs/1012.4126>.
- [104] S P Luttrell. Stochastic vector quantisers. 2000. URL <http://arxiv.org/abs/1012.3705>.
- [105] S P Luttrell. Spatio-temporal inference theory: final report (1 April 1997 - 31 March 2000). Technical Report DERA/CIS(SE1)/651/FUN/STIT/RP/5 [also DERA/S&P/SPI/CR000118], DERA, Malvern, 2000.
- [106] S P Luttrell. Self-organised discovery of structure in signal manifolds. In *Proceedings of DERA workshop on sensor array signal processing*, number DERA/S&E/SPI/TR000349, pages 88–93, Malvern, 2000. DERA.
- [107] S P Luttrell. Invariance discovery by vector quantisation of noisy data. Technical Report DERA/S&P/SPI/TR000070, Defence Evaluation and Research Agency, Malvern, 2000.
- [108] S P Luttrell. Using stochastic encoders to discover structure in data. In *Digest of the IMA international conference on mathematics in signal processing*, Warwick, 2000. IMA.
- [109] S P Luttrell. Invariant stochastic encoders. In *Digest of the IMA international conference on mathematics in signal processing*, Warwick, 2000. IMA.
- [110] S P Luttrell. Using stochastic encoders to discover structure in data. 2000. URL <http://arxiv.org/abs/cs/0408049>.
- [111] S P Luttrell. Invariant stochastic encoders. 2000. URL <http://arxiv.org/abs/cs/0408050>.
- [112] S P Luttrell. *Advances in self-organising maps*, chapter Adaptive subspace encoders using stochastic vector quantisers, pages 102–109. Springer, 2001.
- [113] S P Luttrell. *Mathematics in Signal Processing*, volume 5, chapter Using stochastic vector quantisers to characterise signal and noise subspaces, pages 193–204. Oxford University Press, 2002.
- [114] S P Luttrell. *Multiple classifier systems*, chapter A self-organising approach to multiple classifier fusion, pages 319–328. Lecture notes in computer science. Springer-Verlag, London, 2001.
- [115] S P Luttrell. Adaptive sensor fusion using stochastic vector quantisers. In *Proceedings of the DERA/IEE workshop on intelligent sensor processing*, pages 2/1–2/6, Birmingham, 2001. DERA and IEE.

- [116] S P Luttrell. A self-organising network for sea clutter suppression. Technical Report DERA/S&E/SPI/TR000513, Defence Evaluation and Research Agency, Malvern, 2001.
- [117] S P Luttrell. A comparison of self-organising network and probability modelling approaches to sea clutter suppression. Technical Report DERA/S&E/SPI/TR010477, Defence Evaluation and Research Agency, Malvern, 2001.
- [118] S P Luttrell. Self-organising information processing: research position. Technical Report QinetiQ/S&E/SIP/CR011926, QinetiQ, Malvern, 2001.
- [119] S P Luttrell. Self-organising information processing: research position update. Technical Report QinetiQ/S&E/APC/CR041282, QinetiQ, Malvern, 2004.
- [120] S P Luttrell. Multi-layer self-organising networks using ACEnet. Technical Report QinetiQ/S&E/APC/CR021071, QinetiQ, Malvern, 2002.
- [121] S P Luttrell. Multi-layer self-organising networks using ACEnet: Part 2. Technical Report QinetiQ/S&E/APC/CR021835, QinetiQ, Malvern, 2002.
- [122] S P Luttrell. A self-organising approach to optimally combining classifiers. Technical Report QinetiQ/S&E/SIP/CR011805, QinetiQ, Malvern, 2002.
- [123] S P Luttrell. A self-organising approach to optimally combining classifiers: Part 2. Technical Report QinetiQ/S&E/SIP/CR022003, QinetiQ, Malvern, 2003.
- [124] S P Luttrell. A self-organising approach to optimally combining classifiers: Part 3. Technical Report QinetiQ/S&E/SIP/CR040126, QinetiQ, Malvern, 2004.
- [125] S P Luttrell. A Markov chain approach to multiple classifier fusion. In T Windeatt and F Roli, editors, *Proceedings of international workshop on multiple classifier fusion*, pages 217–226, London, 2003. Springer-Verlag.
- [126] S P Luttrell. Using self-organising mappings to learn the structure of data manifolds. 2003. URL <http://arxiv.org/abs/cs/0406017>.
- [127] S P Luttrell. Discrete network dynamics. Part 1: operator theory. 2005. URL <http://arxiv.org/abs/cs/0511027>.
- [128] S P Luttrell. Review of recurrent networks. Technical Report QinetiQ/SPS/CSIP/TR050877, QinetiQ, Malvern, 2005.
- [129] S P Luttrell. Recurrent self-organising networks: large occupancy case. Technical Report QinetiQ/SPS/CSIP/TR057273, QinetiQ, Malvern, 2006.
- [130] S P Luttrell. Recurrent self-organising networks: small occupancy case. Technical Report QinetiQ/SPS/CSIP/TR057352, QinetiQ, Malvern, 2006.
- [131] S P Luttrell. Proposed use of recurrent self-organising networks for language identification. Technical Report QinetiQ/D&TS/SS/TWP057427, QinetiQ, Malvern, 2006.
- [132] S P Luttrell. Application of symbolic algebra techniques to Bayesian information processing. Technical Report QinetiQ/D&TS/SS/CR0602193, QinetiQ, Malvern, 2006.
- [133] S P Luttrell. Recurrent self-organising networks: a software implementation. Technical Report QinetiQ/SPS/CSIP/TR0703548, QinetiQ, Malvern, 2007.
- [134] S P Luttrell. Proposed use of self-organising networks for identifying communities of interest. Technical Report QinetiQ/D&TS/SS/TR0612427 (draft appendix, omitted in final version), QinetiQ, Malvern, 2007.