Submitted to a Special Issue of IEE Proceedings on Vision, Image and Signal Processing
Created 29 November 1993
An earlier version of this paper appeared in the Proceedings of the International Conference on Artificial Neural Networks, Brighton, 1993, pp. 313-316

The Partitioned Mixture Distribution: An Adaptive Bayesian Network for Low-Level Image Processing

Stephen P Luttrell
Adaptive Systems Theory Section
Defence Research Agency
St Andrews Rd, Malvern,
Worcestershire, WR14 3PS,
United Kingdom
luttrell@signal.dra.hmg.gb

Abstract

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 well-matched to high-dimensional problems such as image processing. An extension, called a partitioned mixture distribution (PMD) is presented, which is essentially a set of overlapping mixture distributions. 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 in the PMD receives the necessary complement of machinery for it to compute its mixture distribution.

1. Introduction

This paper is an expanded version of Luttrell (1), 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, in order to ensure that the operation of 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 (see e.g. Cox (2)).

One of the main problems in Bayesian information processing is to find a flexible way of building its basic computational objects, i.e. the PDFs which it operates upon. There are many techniques in the literature, usually called density estimators, for modelling simple

© Crown Copyright 1993/DRA

PDFs. However, the goal in this paper is to model the rather complicated PDFs that arise in image processing. The main problem in image processing is the extremely large dimensionality of the data; a single image, when represented as a vector, can have as many as 10^6 components (or dimensions). The joint statistical properties of the values recorded in the image pixels can be extremely complicated.

The best existing PDF modelling technique is the Markov random field (MRF) approach (see, for instance, Kindermann and Snell (3)), which imposes a probability measure on the allowed joint pixel values in an image by introducing a fictitious "energy of interaction" between them, and allows low energy images to be more probable than high energy images. The main disadvantage of the MRF approach is that there is no easy way to prevent the interaction energy between adjacent pixels in an image from giving rise to correlations between pixels that are widely separated in the image. In general, it is undesirable that an MRF model that deals adequately with short range correlations should also produce spurious long-range correlations which have to be somehow cancelled by introducing further terms into the interaction energy. The presence of these long-range correlations can also make adaptive training of an MRF model extremely costly, because any changes to the parameters of the interaction energy give rise to corresponding changes in the long-range correlations which can be computed only by lengthy computer simulations.

The problem of spurious long-range correlations can be avoided if the image is broken up into a number of subimages each of which has its own PDF model (not necessarily an MRF model). However, this approach is very crude, because although it allows the statistical properties *within* each subimage to be modelled, it prohibits correlations *between* the subimages from being modelled. This problem can be cured by introducing a top-down hierarchy of PDF models, in which the top-level PDF model deals with correlations *between* the subimages, and the PDF models at the next level down deal with correlations *within* each subimage. This hierarchical approach can be extended to an arbitrary number of levels. However, because it is a top-down approach, if the statistical properties of a patch of image pixels are required, then the statistical properties of all of its ancestors in the hierarchy must also be taken into account. In effect, the long-range correlation problem of standard MRF models has been replaced by the problem of correlations between generations of a hierarchical MRF model. This is a much tamer problem, so progress has been made.

In order to make further progress in simplifying the PDF model it is necessary to appreciate the distinction between top-down and bottom-up approaches. A top-down approach is really a model of how the image data is generated; it can be used to generate synthetic images. This type of approach must be run "in reverse" before it can be used to process the data, and Bayes' theorem provides the necessary machinery to perform this inversion operation. Is it possible to construct a PDF model that is bottom-up from the outset, and which is therefore directly applicable to the problem of image processing? The answer to this question is yes, and the approach is called the adaptive cluster expansion (ACE), which is described in various different ways in Luttrell (4,5,6). The bottom layer of an ACE network has a separate PDF model for each subimage, as discussed in the previous paragraph. An ACE network then introduces additional layers of PDF model, in a

bottom-up fashion, to account for the correlations between the subimages. Note that a bottom-up ACE network is *not* simply the Bayes inverse of a top-down hierarchical MRF network. The problem of designing a whole multilayer ACE network is very interesting, because there are many ways in which the individual PDF models can be constructed and linked together. In this paper the focus will be on using mixture distribution (MD) models, and attention will be limited to a single stage ACE network. Some early discussions on the implementation of ACE using MD appeared in Luttrell (7).

In this paper an extension of the standard MD model, called a partitioned mixture distribution (PMD), is introduced (see Luttrell (8)). 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 neighbouring nodes in the output layer. This limited connectivity allows PMD networks to scale sensibly to high-dimensional problems, such as image processing. It also leads to interesting properties such as lateral inhibition.

In section 2 the Bayesian approach to training a PDF model will be reviewed in considerable detail. In section 3 mixture distributions and their extension to PMDs will be discussed, and the theory of PMDs will be presented in section 4. In section 5 the results of several numerical experiments will be presented in order to demonstrate some of the more interesting properties of PMDs.

2. 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 is characterised by several standard steps that will be described below. Note that the $Q(\cdots)$ notation is used generically to denote "the PDF of ...", and should not be taken to imply that $Q(\cdots)$ is a fixed function of its arguments. The notation will be made more explicit only where its meaning is not clear from the context.

2.1. Step 1: Select a State Space

Choose a state space to describe the system under study, and denote a vector in this space as $\boldsymbol{x}$. For instance, in the case of image processing, the state space might be the space of all images, in which case the components of $\boldsymbol{x}$ would be the pixel values of an image.

2.2. Step 2: Select a PDF model

Define a PDF $Q(\boldsymbol{x})$ to model the joint probability density of the components of the vector $\boldsymbol{x}$. $Q(\boldsymbol{x})$ will have one or more internal variables (or parameters), which may be represented as a vector $\boldsymbol{s}$ which is unknown at the outset, so $Q(\boldsymbol{x})$ may be expanded out as follows

$$Q(\boldsymbol{x}) = \int d\boldsymbol{s}\; Q(\boldsymbol{x}|\boldsymbol{s})\, Q(\boldsymbol{s}) \tag{1}$$

where $Q(\boldsymbol{x}|\boldsymbol{s})$ is the PDF when $\boldsymbol{s}$ is known, and $Q(\boldsymbol{s})$ is the prior PDF of $\boldsymbol{s}$. Thus the PDF $Q(\boldsymbol{x})$ is a weighted average over the various alternative contributing PDFs $Q(\boldsymbol{x}|\boldsymbol{s})$.

2.3. Step 3: Select a Training Set

Define a training set $\boldsymbol{X}$ of vectors drawn from the state space of the system. If there are M such vectors then $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$. If the training samples are drawn independently from the underlying distribution, then the likelihood of the training set $\boldsymbol{X}$ given a particular choice of the parameter vector $\boldsymbol{s}$ is given by

$$Q(\boldsymbol{X}|\boldsymbol{s}) = \prod_{i=1}^{M} Q(\boldsymbol{x}_i|\boldsymbol{s}) \tag{2}$$

and the logarithmic likelihood is given by

$$\begin{aligned} L(\boldsymbol{X}|\boldsymbol{s}) &\equiv \log Q(\boldsymbol{X}|\boldsymbol{s}) \\ &= \sum_{i=1}^{M} \log Q(\boldsymbol{x}_i|\boldsymbol{s}) \end{aligned} \tag{3}$$

2.4. Step 4: Train the PDF Model

Present the training data $\boldsymbol{X}$ to the model $Q(\boldsymbol{x})$ in order that the model can learn something about the true state of its internal parameter vector $\boldsymbol{s}$. The overall effect of this is to convert the prior PDF $Q(\boldsymbol{s})$ into a posterior PDF $Q(\boldsymbol{s}|\boldsymbol{X})$ that embodies the prior information contained in $Q(\boldsymbol{s})$ plus the additional information contained in $\boldsymbol{X}$. This conversion trick is implemented using Bayes' theorem thus

$$Q(\boldsymbol{s}|\boldsymbol{X}) = \frac{Q(\boldsymbol{X}|\boldsymbol{s})\, Q(\boldsymbol{s})}{\int d\boldsymbol{s}'\, Q(\boldsymbol{X}|\boldsymbol{s}')\, Q(\boldsymbol{s}')} = Q(\boldsymbol{s}) \prod_{i=1}^{M} \left(\frac{Q(\boldsymbol{x}_i|\boldsymbol{s})}{\int d\boldsymbol{s}'\, Q(\boldsymbol{x}_i|\boldsymbol{s}')\, Q(\boldsymbol{s}')} \right) \tag{4}$$

where the product ranges over the (assumed) statistically independent training vectors in $\boldsymbol{X}$. The training process converts $Q(\boldsymbol{x})$ into $Q(\boldsymbol{x}|\boldsymbol{X})$ thus

$$Q(\boldsymbol{x}|\boldsymbol{X}) = \int d\boldsymbol{s}\; Q(\boldsymbol{x}|\boldsymbol{s})\, Q(\boldsymbol{s}|\boldsymbol{X}) \tag{5}$$

which is a weighted average over the possible contributing PDFs $Q(\boldsymbol{x}|\boldsymbol{s})$, with the untrained prior PDF $Q(\boldsymbol{s})$ replaced by the trained PDF $Q(\boldsymbol{s}|\boldsymbol{X})$. This result should be compared with Equation 1.

2.5. Step 5: Approximate the PDF Model

$Q(\boldsymbol{x}|\boldsymbol{X})$ is usually difficult to calculate exactly because it involves an integration over alternative states of the parameter vector $\boldsymbol{s}$. There are many ways in which approximations can be made, depending on the peculiarities of the $\boldsymbol{s}$ integral and on what $Q(\boldsymbol{x}|\boldsymbol{X})$ will be used for later on. However, if the integral is dominated by contributions from the

integrand in the vicinity of a single value $\boldsymbol{s}(\boldsymbol{X})$ of $\boldsymbol{s}$, then $Q(\boldsymbol{s}|\boldsymbol{X})$ can be approximated by a point PDF $\delta(\boldsymbol{s}-\boldsymbol{s}(\boldsymbol{X}))$ located at $\boldsymbol{s}(\boldsymbol{X})$. This simplifies $Q(\boldsymbol{x}|\boldsymbol{X})$ as follows

$$\begin{aligned} Q(\boldsymbol{x}|\boldsymbol{X}) &\approx \int d\boldsymbol{s}\; Q(\boldsymbol{x}|\boldsymbol{s})\,\delta(\boldsymbol{s}-\boldsymbol{s}(\boldsymbol{X})) \\ &= Q(\boldsymbol{x}|\boldsymbol{s}(\boldsymbol{X})) \end{aligned} \tag{6}$$

The optimum location $\boldsymbol{s}(\boldsymbol{X})$ of this point PDF $\delta(\boldsymbol{s}-\boldsymbol{s}(\boldsymbol{X}))$ is debatable. Two popular schemes are

$$\boldsymbol{s}(\boldsymbol{X}) = \begin{cases} \arg\max\limits_{\boldsymbol{s}} Q(\boldsymbol{X}|\boldsymbol{s}) & \text{maximum likelihood} \\ \arg\max\limits_{\boldsymbol{s}} Q(\boldsymbol{s}|\boldsymbol{X}) & \text{maximum posterior probability} \end{cases} \tag{7}$$

Note that the maximum likelihood prescription can be written out explicitly as

$$\boldsymbol{s}(\boldsymbol{X}) = \arg\max_{\boldsymbol{s}} \sum_{i=1}^{M} \log Q(\boldsymbol{x}_i|\boldsymbol{s}) \tag{8}$$

If the point PDF approximation is replaced by a weighted sum of point PDFs then Equation 6 becomes

$$\begin{aligned} Q(\boldsymbol{x}|\boldsymbol{X}) &\approx \int d\boldsymbol{s}\; Q(\boldsymbol{x}|\boldsymbol{s}) \sum_{c=1}^{n} Q(c|\boldsymbol{X})\,\delta(\boldsymbol{s}-\boldsymbol{s}_c(\boldsymbol{X})) \\ &= \sum_{c=1}^{n} Q(\boldsymbol{x}|\boldsymbol{s}_c(\boldsymbol{X}))\, Q(c|\boldsymbol{X}) \end{aligned} \tag{9}$$

where the $Q(c|\boldsymbol{X})$ are normalised as $\sum_{c=1}^{n} Q(c|\boldsymbol{X}) = 1$. This type of approximation to $Q(\boldsymbol{x}|\boldsymbol{X})$ is called a mixture distribution because $Q(\boldsymbol{x}|\boldsymbol{X})$ is approximated by a PDF that is a weighted mixture of n separate PDFs $Q(\boldsymbol{x}|\boldsymbol{s}_c(\boldsymbol{X}))$. The $\boldsymbol{s}_c(\boldsymbol{X})$ and $Q(c|\boldsymbol{X})$ may be optimised using an approximate prescription such as the maximum likelihood method.

To make contact with the more usual terminology of density estimation theory, c is usually referred to as a class label, in which case $Q(c|\boldsymbol{X})$ is the prior probability of class c, and $Q(\boldsymbol{x}|\boldsymbol{s}_c(\boldsymbol{X}))$ is the likelihood that data vector $\boldsymbol{x}$ could be produced by class c.

3. Adaptive Mixture Distribution Networks

In this section mixture distribution networks will be discussed in much greater detail, and the generalisation to partitioned mixture distributions (PMD) will be presented.

3.1. The Neural Network Interpretation of a Mixture Distribution

The optimum choice of parameter vectors $\boldsymbol{s}_c(\boldsymbol{X})$ and weights $Q(c|\boldsymbol{X})$ to use in a mixture distribution is determined by the training data $\boldsymbol{X}$ and the chosen optimisation criterion. It

will be assumed henceforth that the maximum likelihood prescription is the chosen optimisation scheme.

A mixture distribution can be interpreted in "neural network" language as follows. The parameters $\boldsymbol{s}_c(\boldsymbol{X})$ and $Q(c|\boldsymbol{X})$ correspond to the neural network weights (or whatever) that emerge from the training process. Subsequent computation of the mixture distribution approximation to $Q(\boldsymbol{x}|\boldsymbol{X})$ 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.

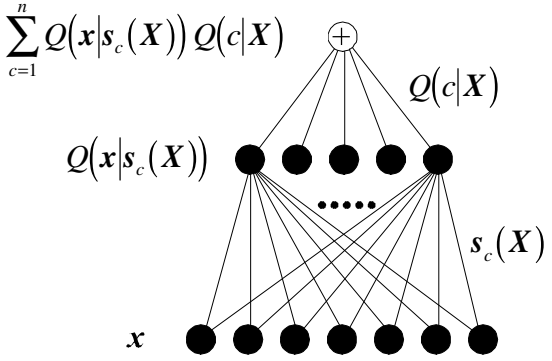


Figure 1: A mixture distribution represented as a neural network. The bottom layer represents the components of the input vector $\boldsymbol{x}$. The middle layer represents the output $Q(\boldsymbol{x}|\boldsymbol{s}_c(\boldsymbol{X}))$ for each possible value of the class label *c*; these are the class likelihoods. The top layer is the network output, which is a linear combination (or mixture) of the various $Q(\boldsymbol{x}|\boldsymbol{s}_c(\boldsymbol{X}))$ weighted by the class prior probabilities $Q(c|\boldsymbol{X})$.

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 $Q(\boldsymbol{x}|\boldsymbol{s}_c(\boldsymbol{X}))$ indexed by the label *c*, which are then transformed via a linear summation operation weighted by the prior probabilities $Q(c|\boldsymbol{X})$ to produce the mixture distribution $\sum_{c=1}^{n} Q(\boldsymbol{x}|\boldsymbol{s}_c(\boldsymbol{X}))\, Q(c|\boldsymbol{X})$. The goal is to maximise the likelihood of the training data with respect to $\boldsymbol{s}_c(\boldsymbol{X})$ and $Q(c|\boldsymbol{X})$ thus

$$\{\boldsymbol{s}_c(\boldsymbol{X}), Q(c|\boldsymbol{X}): c = 1,2,\cdots,n\} = \underset{\{\boldsymbol{s}_c, Q(c): c = 1,2,\cdots,n\}}{\arg\max} \sum_{i=1}^{M} \log\left(\sum_{c'=1}^{n} Q(\boldsymbol{x}_i|\boldsymbol{s}_{c'})\, Q(c') \right) \quad (10)$$

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 neural network training procedure.

3.2. Multiple Independent Mixture Distributions

In situations where the dimensionality $\dim \boldsymbol{x}$ of the input vector $\boldsymbol{x}$ 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 $\boldsymbol{x}$ is partitioned into *N* subspaces

as $\boldsymbol{x} = \left(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^N\right)$, and likewise the training data $\boldsymbol{X} = \left(\boldsymbol{X}^1, \boldsymbol{X}^2, \cdots, \boldsymbol{X}^N\right)$, and a mixture distribution $Q^k\left(\boldsymbol{x}^k \middle| \boldsymbol{X}^k\right) \quad k = 1,2,\cdots,N$ is constructed independently for each subspace. The overall PDF $Q\left(\boldsymbol{x} \middle| \boldsymbol{X}\right)$ is then the product of the individual $Q^k\left(\boldsymbol{x}^k \middle| \boldsymbol{X}^k\right)$.

$$Q\left(\boldsymbol{x} \middle| \boldsymbol{X}\right) = \prod_{k=1}^{N} Q^k\left(\boldsymbol{x}^k \middle| \boldsymbol{X}^k\right) \tag{11}$$

This expression does not take account of any correlations that might exist between the various $\boldsymbol{x}^k$ for different values of *k*.

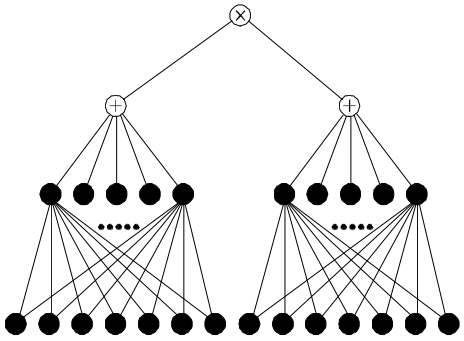


Figure 2: A pair of independent mixture distribution networks. The final output is the product of the output of the pair of mixture distribution networks. The left hand half of the network has input vector $\boldsymbol{x}^1$, parameter vectors $\boldsymbol{s}^1_{c^1}\left(\boldsymbol{X}^1\right)$, prior probabilities $Q\left(c^1 \middle| \boldsymbol{X}^1\right)$, and output $\sum_{c^1=1}^{n^1} Q\left(\boldsymbol{x}^1 \middle| \boldsymbol{s}^1_{c^1}\left(\boldsymbol{X}^1\right)\right) Q\left(c^1 \middle| \boldsymbol{X}^1\right)$. A similar comment applies to 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 $\left(\boldsymbol{x}^1, \boldsymbol{x}^2\right)$, 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.

Each $Q^k\left(\boldsymbol{x}^k \middle| \boldsymbol{X}^k\right)$ 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 $Q\left(\boldsymbol{x} \middle| \boldsymbol{X}\right)$

$$Q\left(\boldsymbol{x} \middle| \boldsymbol{X}\right) \approx \prod_{k=1}^{N} \left(\sum_{c^k=1}^{n^k} Q\left(\boldsymbol{x}^k \middle| \boldsymbol{s}^k_c\left(\boldsymbol{X}^k\right)\right) Q\left(c^k \middle| \boldsymbol{X}^k\right) \right) \tag{12}$$

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 10.

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*.

3.3. 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. What is required is a network structure that treats each patch of data on a equal footing, unlike Figure 2 which partitions the input vector $\boldsymbol{x}$ into left and right halves as $\left(\boldsymbol{x}^1, \boldsymbol{x}^2\right)$.

In the following subsections a two stage derivation of such a network will be presented, starting with a standard mixture distribution network, and finishing with what will be called a partitioned mixture distribution (PMD) network, Luttrell (8). For concreteness, this derivation will refer to the network in Figure 2, but the technique may be applied to arbitrary *N*.

3.3.1. 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.

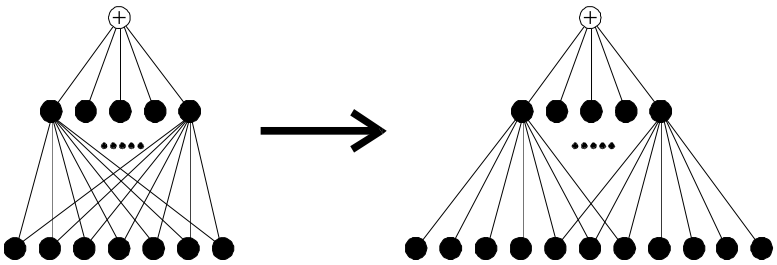


Figure 3: Modification of the basic mixture distribution network. In the standard mixture distribution each mixture component (or class) sees the whole of the input space, whereas in the modified version each class sees only part of the input space, as indicated by the connections drawn in the right 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.

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 $Q\left(\boldsymbol{x}\middle|\boldsymbol{s}_c\left(\boldsymbol{X}\right)\right)$ 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 $\boldsymbol{x}$ 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 each seen by only one component of the mixture distribution, whereas the central three components of $\boldsymbol{x}$ are seen by all five components of the mixture distribution. Therefore, the scope for tailoring

the shape of the mixture distribution $Q(\boldsymbol{x}|\boldsymbol{X})$ 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 mixture distribution

With this modification, the network in Figure 2 would become as shown in Figure 4.

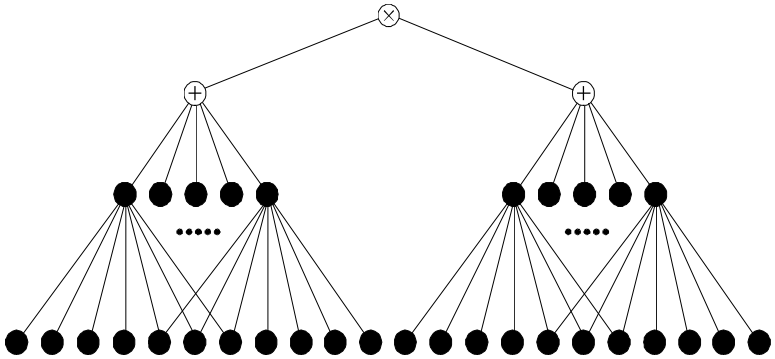


Figure 4: A pair of independent mixture distribution networks, as in Figure 2, but using the modified mixture distribution network shown in Figure 3.

3.3.2. 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".

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 $\boldsymbol{x}$ (e.g. the correlation between pixels on opposite sides of an image) because no mixture distribution simultaneously sees information from widely separated sources. A multilayer PMD network based on the "Adaptive Cluster Expansion" approach, Luttrell (4,5,6), is needed to remove this limitation.

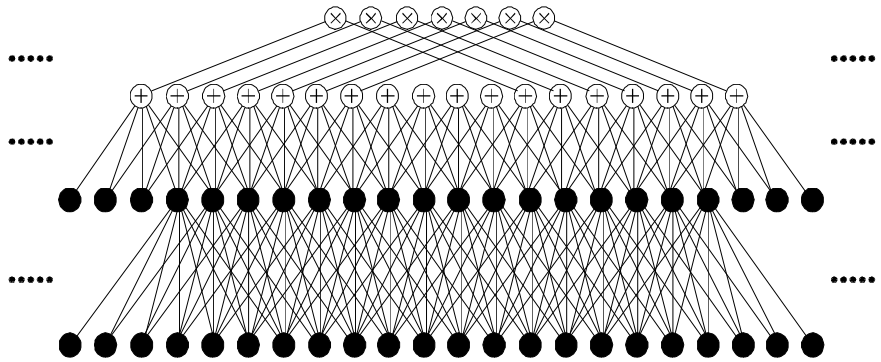


Figure 5: A superposition of mixture distribution networks of the type shown in Figure 4. Because the input vector is assumed to be high-dimensional, the dots on either side of the diagram indicate that the basic structure is to be replicated across the input vector. This network will be referred to as a partitioned mixture distribution (PMD) network

4. The Theory of PMDs

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.

4.1. The Basic Theory of PMDs

For a PMD the basic definition of a mixture distribution (see Equation 9) becomes

$$Q_c(\boldsymbol{x}|\boldsymbol{X}) = \sum_{c'=c-n/2}^{c+n/2} Q(\boldsymbol{x}, c'|\boldsymbol{X}) = \frac{\sum_{c'=c-n/2}^{c+n/2} Q(\boldsymbol{x}|\boldsymbol{s}_{c'}(\boldsymbol{X}))Q(c'|\boldsymbol{X})}{\sum_{c''=c-n/2}^{c+n/2} Q(c''|\boldsymbol{X})} \qquad c = 1,2,\cdots,N \tag{13}$$

where edge effects are ignored (this could be achieve by using circular boundary conditions, for instance). The index c selects which of the N mixture distributions is required, the half-width parameter $n/2$ is used to determine the number of components in each mixture distribution (if n is odd then $n/2$ is replaced by the largest integer less than $n/2$), and the normalisation factor $\sum_{c'=c-n/2}^{c+n/2} Q(c'|\boldsymbol{X})$ ensures that each mixture distribution is normalised so that its integral over $\boldsymbol{x}$ is unity. 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, an augmented set $\boldsymbol{S}$ of parameters will be defined as

$$\boldsymbol{S} \equiv \{\boldsymbol{s}_c, Q(c): c = 1,2,\cdots N\} \tag{14}$$

and the notation $\boldsymbol{S}(\boldsymbol{X})$ will be used to denote a single choice of $\boldsymbol{S}$ (such as the maximum likelihood choice) for the training set $\boldsymbol{X}$. The maximum likelihood procedure for optimising $\boldsymbol{S}$ for a set of N independent mixture distributions 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

$$\boldsymbol{S}(\boldsymbol{X}) = \underset{\boldsymbol{S}}{\arg\max} \sum_{c=1}^{N} \sum_{i=1}^{M} \log Q_c(\boldsymbol{x}_i|\boldsymbol{S}) \tag{15}$$

where

$$Q_c(\boldsymbol{x}_i|\boldsymbol{S}) = \frac{\sum_{c'=c-n/2}^{c+n/2} Q(\boldsymbol{x}_i|\boldsymbol{s}_{c'})Q(c')}{\sum_{c''=c-n/2}^{c+n/2} Q(c'')} \tag{16}$$

This result should be compared with Equation 8. The various parameter vectors now cooperate amongst themselves in order to maximise the average logarithmic likelihood. They must cooperate in such a way that each optimal mixture distribution $Q_c(\boldsymbol{x}|\boldsymbol{S}(\boldsymbol{X}))$ in the PMD can do a good job of modelling the PDF in the subspace of $\boldsymbol{x}$ that it sees.

In the limit of a large training set size Equation 15 is equivalent to maximising the average logarithmic likelihood.

$$\boldsymbol{S}(\boldsymbol{X}) = \underset{\boldsymbol{S}}{\arg\max} \int d\boldsymbol{x}\ P(\boldsymbol{x}) \sum_{c=1}^{N} \log Q_c(\boldsymbol{x}|\boldsymbol{S}) \tag{17}$$

where the average over training data has been replaced by a shorthand notation in which the PDF $P(\boldsymbol{x})$ is used to represent the density of training vectors, as follows

$$\frac{1}{M}\sum_{i=1}^{M}(\cdots) \rightarrow \int d\boldsymbol{x}\ P(\boldsymbol{x})(\cdots) \tag{18}$$

In PMD calculations it is useful to write the joint probability for the mixture distribution centred on class c as

$$Q_c(\boldsymbol{x}, c'|\boldsymbol{S}) = \begin{cases} \dfrac{Q(\boldsymbol{x}|\boldsymbol{s}_{c'})Q(c')}{\sum_{c''=c-n/2}^{c+n/2} Q(c'')} & |c'-c| \le n/2 \\ 0 & |c'-c| > n/2 \end{cases} \tag{19}$$

and the corresponding posterior probability as

$$Q_c(c'|\boldsymbol{x}, \boldsymbol{S}) = \begin{cases} \dfrac{Q(\boldsymbol{x}|\boldsymbol{s}_{c'})Q(c')}{\sum_{c''=c-n/2}^{c+n/2} Q(\boldsymbol{x}|\boldsymbol{s}_{c''})Q(c'')} & |c'-c| \le n/2 \\ 0 & |c'-c| > n/2 \end{cases} \tag{20}$$

which satisfies the normalisation condition $\sum_{c'=c-n/2}^{c+n/2} Q_c(c'|\boldsymbol{x}, \boldsymbol{S}) = 1$. In a PMD there are N separate mixture distributions, each with its own set of posterior probabilities, which occupy class labels $c-n/2, c-n/2+1, \cdots, c+n/2$ for the cth 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

$$\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S}) \equiv \sum_{c'=c-n/2}^{c+n/2} Q_{c'}(c|\boldsymbol{x},\boldsymbol{S}) = Q(\boldsymbol{x}_i|\boldsymbol{s}_c)Q(c) \sum_{c'=c-n/2}^{c+n/2} \frac{1}{\sum_{c''=c'-n/2}^{c'+n/2} Q(\boldsymbol{x}_i|\boldsymbol{s}_{c''})Q(c'')} \tag{21}$$

This definition implies the normalisation condition $\sum_{c=1}^{N} \tilde{Q}(c|\boldsymbol{x},\boldsymbol{S}) = N$, 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 $\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S})$ 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

$$\tilde{Q}(c|\boldsymbol{S}) \equiv Q(c) \sum_{c'=c-n/2}^{c+n/2} \frac{1}{\sum_{c''=c'-n/2}^{c'+n/2} Q(c'')} \tag{22}$$

which satisfies $\sum_{c=1}^{N} \tilde{Q}(c) = N$.

4.2. The Expectation-Maximisation (EM) Method for Training PMDs

The basic mixture distribution optimisation prescription in Equation 8, and the generalised PMD prescription in Equation 15, need to be converted into an explicit training algorithm. The are two basic types of algorithm: the expectation-maximisation (EM) method (see Dempster, Laird and Rubin (9)), 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 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. 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 $\Delta L(\boldsymbol{S},\boldsymbol{S}_0;\boldsymbol{X})$ between the logarithmic likelihood evaluated for two different choices, $\boldsymbol{S} \equiv \{\boldsymbol{s}_c, Q(c): c = 1,2,\cdots N\}$ and $\boldsymbol{S}_0 \equiv \{\boldsymbol{s}_{0,c}, Q_0(c): c = 1,2,\cdots N\}$, of the parameters

$$\begin{aligned} \Delta L(\boldsymbol{S},\boldsymbol{S}_0;\boldsymbol{X}) &\equiv L(\boldsymbol{X}|\boldsymbol{S}) - L(\boldsymbol{X}|\boldsymbol{S}_0) \\ &= \int d\boldsymbol{x}\ P(\boldsymbol{x}) \sum_{c=1}^{N} \log\left(\frac{Q_c(\boldsymbol{x}|\boldsymbol{S})}{Q_c(\boldsymbol{x}|\boldsymbol{S}_0)}\right) \end{aligned} \tag{23}$$

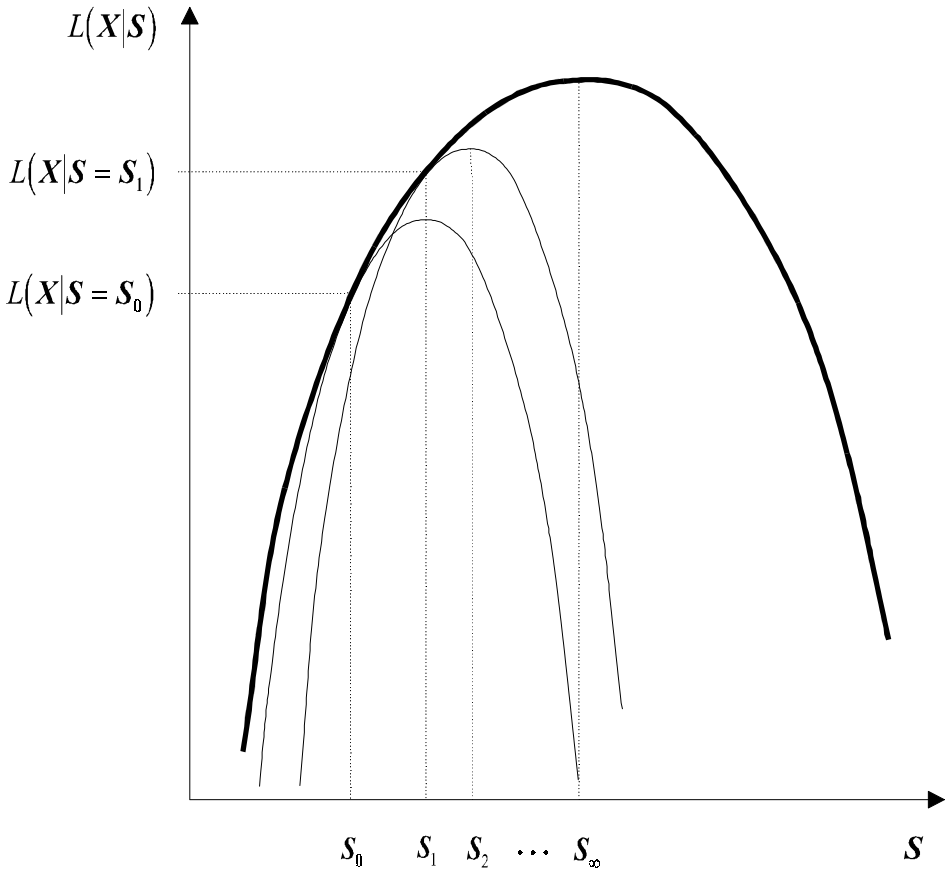


Figure 6: An example of a logarithmic likelihood function $L(X|S)$ is shown in bold. Assume that the starting value of the parameter vector is $S = S_0$, and the goal is to locate the position S_∞ of the peak of the $L(X|S)$ curve. The EM method proceeds by fitting a lower bound function beneath the $L(X|S)$ curve, which touches it at the point $S = S_0$, as shown in the diagram, and then locating the peak S_1 of this lower bound. Some progress has been made towards the maximum of $L(X|S)$, as can be seen by inspecting the diagram. This procedure is then repeated with the new starting value $S = S_1$. After several iterations of this type the value of S approaches the required location S_∞ of the peak of the logarithmic likelihood function $L(X|S)$, as can readily be seen in the diagram. The key trick to 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 steps of the derivation are as follows.

$$\begin{aligned}
\Delta L(\boldsymbol{S},\boldsymbol{S}_0;\boldsymbol{X}) &= \int d\boldsymbol{x}\ P(\boldsymbol{x}) \sum_{c=1}^{N} \log\left(\frac{\sum_{c'=c-n/2}^{c+n/2} Q(\boldsymbol{x}|\boldsymbol{s}_{c'}) Q(c')}{Q_c(\boldsymbol{x}|\boldsymbol{S}_0) \sum_{c''=c-n/2}^{c+n/2} Q(c'')} \right) \\
&= \int d\boldsymbol{x}\ P(\boldsymbol{x}) \sum_{c=1}^{N} \log\left(\sum_{c'=c-n/2}^{c+n/2} Q_c(c'|\boldsymbol{x},\boldsymbol{S}_0) \frac{Q(\boldsymbol{x}|\boldsymbol{s}_{c'}) Q(c')}{Q_c(\boldsymbol{x},c'|\boldsymbol{S}_0) \sum_{c''=c-n/2}^{c+n/2} Q(c'')} \right) \\
&\geq \int d\boldsymbol{x}\ P(\boldsymbol{x}) \sum_{c=1}^{N} \sum_{c'=c-n/2}^{c+n/2} Q_c(c'|\boldsymbol{x},\boldsymbol{S}_0) \log\left(\frac{Q(\boldsymbol{x}|\boldsymbol{s}_{c'}) Q(c') \sum_{c'''=c-n/2}^{c+n/2} Q_0(c''')}{Q(\boldsymbol{x}|\boldsymbol{s}_{0,c'}) Q_0(c') \sum_{c''=c-n/2}^{c+n/2} Q(c'')} \right) \\
&= \Delta L_0(\boldsymbol{S},\boldsymbol{S}_0;\boldsymbol{X})
\end{aligned} \tag{24}$$

where $\Delta L_0(\boldsymbol{S},\boldsymbol{S}_0;\boldsymbol{X})$ is defined as

$$\begin{aligned}
\Delta L_0(\boldsymbol{S},\boldsymbol{S}_0;\boldsymbol{X}) \equiv \int d\boldsymbol{x}\ P(\boldsymbol{x}) \sum_{c=1}^{N} \left(\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S}_0) \log\left(Q(\boldsymbol{x}|\boldsymbol{s}_c) Q(c) \right) - \log\left(\sum_{c'=c-n/2}^{c+n/2} Q(c') \right) \right) \\
+\text{a term that is independent of } \boldsymbol{S}
\end{aligned} \tag{25}$$

The penultimate step of the derivation in Equation 24 was obtained by using Jensen's inequality for convex functions, and the last step was obtained by using the result $\sum_{c'=c-n/2}^{c+n/2} Q_c(c'|\boldsymbol{x},\boldsymbol{S}) = 1$ and the result

$$\begin{aligned}
\sum_{c=1}^{N} \sum_{c'=c-n/2}^{c-n/2} Q_c(c'|\boldsymbol{x},\boldsymbol{S}) (\cdots) &= \sum_{c'=1}^{N} \sum_{c=c'-n/2}^{c'-n/2} Q_c(c'|\boldsymbol{x},\boldsymbol{S}) (\cdots) \\
&= \sum_{c'=1}^{N} \tilde{Q}(c'|\boldsymbol{x},\boldsymbol{S}) (\cdots)
\end{aligned} \tag{26}$$

where the term $(\cdots)$ is symmetric under interchange of c and c'. Combining the above results yields

$$L(\boldsymbol{X}|\boldsymbol{S}) \geq \Delta L_0(\boldsymbol{S},\boldsymbol{S}_0;\boldsymbol{X}) + L(\boldsymbol{X}|\boldsymbol{S}_0) \qquad \text{where } \Delta L_0(\boldsymbol{S},\boldsymbol{S};\boldsymbol{X}) = 0 \tag{27}$$

$\Delta L_0(\boldsymbol{S},\boldsymbol{S}_0;\boldsymbol{X}) + L(\boldsymbol{X}|\boldsymbol{S}_0)$ 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

$$\boldsymbol{S}_1 = \underset{\boldsymbol{S}}{\arg\max} \int d\boldsymbol{x}\ P(\boldsymbol{x}) \sum_{c=1}^{N} \left(\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S}_0) \log Q(\boldsymbol{x},c|\boldsymbol{S}) - \log \sum_{c'=c-n/2}^{c+n/2} Q(c'|\boldsymbol{S}) \right) \tag{28}$$

where $\boldsymbol{S}$ and $\boldsymbol{S}_0$ have been carefully used on the right hand side to indicate exactly which terms are varied in the maximisation process (i.e. those containing $\boldsymbol{S}$). Subsequent iterations of the EM algorithm are analogously defined. Note that each update step is not

obliged to update all of the parameters in $\boldsymbol{S}$. For instance, the $Q(c)$ might be held constant, and only the $\boldsymbol{s}_c$ 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 $\boldsymbol{S}_0, \boldsymbol{S}_1, \cdots, \boldsymbol{S}_\infty$ of successively improved estimates of the required $\boldsymbol{S}(\boldsymbol{X})$. 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 local maxima without becoming trapped.

4.3. The Gaussian Partitioned Mixture Distribution

A popular model to use for $Q(\boldsymbol{x}|\boldsymbol{s}_c)$ is a Gaussian PDF with mean $\boldsymbol{m}_c$ and covariance $\boldsymbol{A}_c$.

$$Q(\boldsymbol{x}|\boldsymbol{s}_c) \to Q(\boldsymbol{x}|\boldsymbol{m}_c, \boldsymbol{A}_c) \equiv \frac{1}{\sqrt{\det 2\pi \boldsymbol{A}_c}} \exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_c)^T \boldsymbol{A}_c^{-1}(\boldsymbol{x}-\boldsymbol{m}_c)\right) \tag{29}$$

The likelihoods $Q_c(\boldsymbol{x}|\boldsymbol{S})$, joint probabilities $Q_c(\boldsymbol{x}, c'|\boldsymbol{S})$, posterior probabilities $Q_c(c'|\boldsymbol{x}, \boldsymbol{S})$, and averaged posterior probabilities $\tilde{Q}(c|\boldsymbol{x}, \boldsymbol{S})$ may readily be obtained from $Q(\boldsymbol{x}|\boldsymbol{s}_c)$ and $Q(c)$ using the theory outlined earlier.

The EM method may be used to solve for the maximum likelihood choice of $\boldsymbol{S}$ as follows. Differentiate the right hand side of the EM update equation with respect to $Q(c|\boldsymbol{S})$, set the result to zero, interchange the order of summations, and use the definition in Equation 22 to obtain an implicit equation for the updated $Q(c|\boldsymbol{S})$

$$\tilde{Q}(c|\boldsymbol{S}_1) = \int d\boldsymbol{x}\ P(\boldsymbol{x})\, \tilde{Q}(c|\boldsymbol{x}, \boldsymbol{S}_0) \tag{30}$$

Similarly, differentiate with respect to $\boldsymbol{m}_c$ (assuming that $\boldsymbol{A}_c^{-1}$ is full rank) to obtain the updated $\boldsymbol{m}_c$

$$\boldsymbol{m}_c(\boldsymbol{S}_1) = \frac{\int d\boldsymbol{x}\ P(\boldsymbol{x})\, \tilde{Q}(c|\boldsymbol{x}, \boldsymbol{S}_0)\, \boldsymbol{x}}{\int d\boldsymbol{x}\ P(\boldsymbol{x})\, \tilde{Q}(c|\boldsymbol{x}, \boldsymbol{S}_0)} \tag{31}$$

Similarly, differentiate with respect to $\boldsymbol{A}_c^{-1}$ (using $\partial \log\det \boldsymbol{A}_c^{-1} / \partial \boldsymbol{A}_c^{-1} = \boldsymbol{A}_c^T$) to obtain the updated $\boldsymbol{A}_c$

$$\boldsymbol{A}_c(\boldsymbol{S}_1) = \frac{\int d\boldsymbol{x}\ P(\boldsymbol{x})\, \tilde{Q}(c|\boldsymbol{x}, \boldsymbol{S}_0)\, (\boldsymbol{x}-\boldsymbol{m}_c(\boldsymbol{S}_1))(\boldsymbol{x}-\boldsymbol{m}_c(\boldsymbol{S}_1))^T}{\int d\boldsymbol{x}\ P(\boldsymbol{x})\, \tilde{Q}(c|\boldsymbol{x}, \boldsymbol{S}_0)} \tag{32}$$

where the right hand side depends on the updated $\boldsymbol{m}_c$. In order to simplify notation define the first three moments of the posterior probability weighted training data as

$$\begin{aligned} M_c^0(\boldsymbol{S}_0) &\equiv \int d\boldsymbol{x}\ P(\boldsymbol{x})\, \tilde{Q}(c|\boldsymbol{x}, \boldsymbol{S}_0) \\ \boldsymbol{M}_c^1(\boldsymbol{S}_0) &\equiv \int d\boldsymbol{x}\ P(\boldsymbol{x})\, \tilde{Q}(c|\boldsymbol{x}, \boldsymbol{S}_0)\, \boldsymbol{x} \\ \boldsymbol{M}_c^2(\boldsymbol{S}_0) &\equiv \int d\boldsymbol{x}\ P(\boldsymbol{x})\, \tilde{Q}(c|\boldsymbol{x}, \boldsymbol{S}_0)\, \boldsymbol{x}\boldsymbol{x}^T \end{aligned} \tag{33}$$

in which case the solutions to the EM update equation may be expressed as

$$\begin{aligned}\tilde{Q}(c|\boldsymbol{S}_1) &= M_c^0(\boldsymbol{S}_0)\\ \boldsymbol{m}_c(\boldsymbol{S}_1) &= \frac{\boldsymbol{M}_c^1(\boldsymbol{S}_0)}{M_c^0(\boldsymbol{S}_0)}\\ \boldsymbol{A}_c(\boldsymbol{S}_1) &= \frac{\boldsymbol{M}_c^2(\boldsymbol{S}_0)}{M_c^0(\boldsymbol{S}_0)} - \boldsymbol{m}_c(\boldsymbol{S}_1)\boldsymbol{m}_c(\boldsymbol{S}_1)^T\end{aligned} \tag{34}$$

The updated prior probabilities $Q(c|\boldsymbol{S}_1)$ may be found by solving the first of these equations iteratively, but the means $\boldsymbol{m}_c(\boldsymbol{S}_1)$ and covariances $\boldsymbol{A}_c(\boldsymbol{S}_1)$ may be obtained directly in closed form using the second and third equations respectively. In the numerical simulations presented in the next section the prior probabilities are fixed to be constant and equal (i.e. they are not updated), and the covariances are assumed to be diagonal.

These results for a Gaussian PMD reduce to the standard results for the corresponding Gaussian mixture distribution when n=N.

5. 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 1-dimensional PMD with 21 nodes, an input window width of 7, a mixture window width (i.e. n) of 7, holding the prior probabilities constant and equal, and using diagonal (and sometimes isotropic) covariance matrices.

An interesting modification to the training algorithm (not derived above) was also sometimes used. Thus the averaged posterior probabilities $\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S})$ were modified as follows

$$\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S}) \rightarrow (1-2\varepsilon)\,\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S}) + \varepsilon\left(\tilde{Q}(c-1|\boldsymbol{x},\boldsymbol{S}) + \tilde{Q}(c+1|\boldsymbol{x},\boldsymbol{S})\right) \tag{35}$$

where $0 \le \varepsilon \le 0.5$. In effect node c leaks some of its activity onto its neighbours. This type of modification arises naturally in the extension of PMD theory which include time history (not discussed in this paper). This can readily be implemented by replacing the prior probabilities $Q(c)$ by the averaged posterior probabilities $\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S})$ derived from the *previous* time instant and then leaked as above. 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.

5.1. 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 static input vector. Thus the initial presentation of data $\boldsymbol{x}$ causes the PMD to output a set of posterior probabilities $\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S};t=0)$ at each node, which is then fed back, perhaps after node

activity leakage, to become the set of prior probabilities $Q(c|\boldsymbol{S};t=1)$ at the next time instant. The input vector remains unchanged, and a new $\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S};t=1)$ is then output, and fed back, an so on. The lateral inhibition effect becomes obvious when the pattern of numbers $\{\tilde{Q}(c|\boldsymbol{x},\boldsymbol{S};t): c=1,2,\cdots,N\}$ is plotted out for successive times *t*.

For the purpose of this simulation it is not necessary to introduce explicit likelihood functions $Q(\boldsymbol{x}|\boldsymbol{s}_c)$; it is sufficient to use input data that is itself the pattern of numbers $\{Q(\boldsymbol{x}|\boldsymbol{s}_c): c=1,2,\cdots,N\}$.

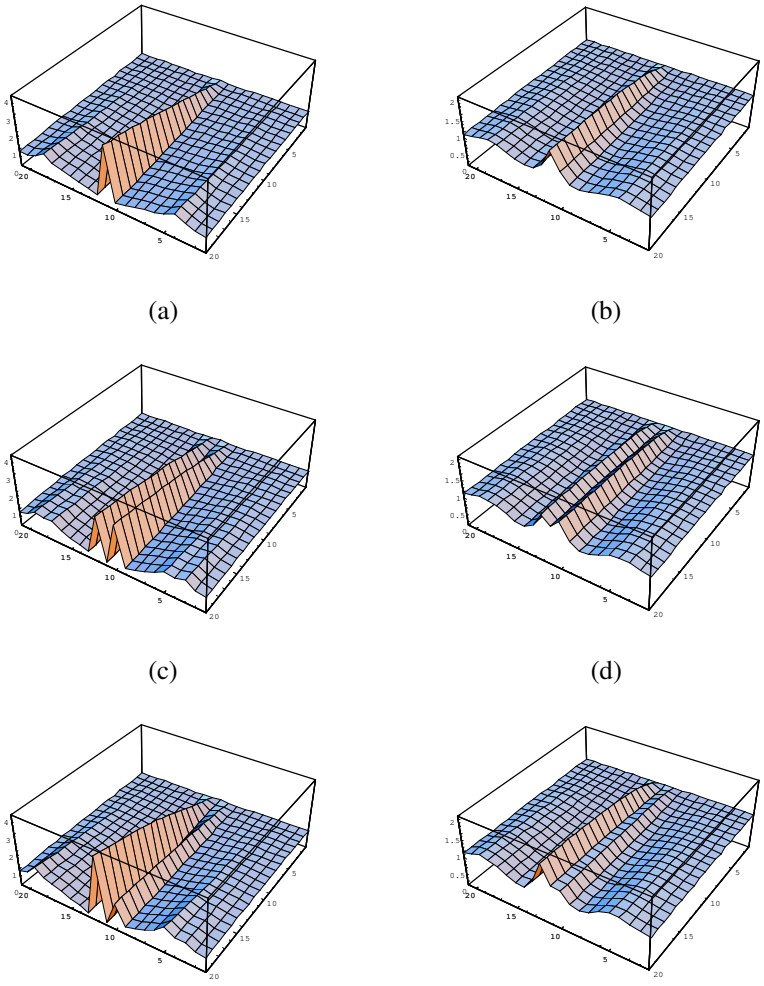
In Figure 7a and 7b the point stimulus leads to a strong response from the stimulated node, and a suppressed reponse from the surrounding nodes. This lateral inhibition effect arises from the form of the expression used to define the posterior probability in a PMD. Because the node responses are actually 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 7d the two equal point stimuli lead to equal responses because neither leads to a single node gaining supremacy. However in Figure 7e and 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 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.

5.2. 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 $\boldsymbol{s}_c(\boldsymbol{X})$ 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 4 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. This will be referred to as the "completeness" property.



(a) (b) (c) (d) (e) (f)

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 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 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. 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 and to the left of the page).

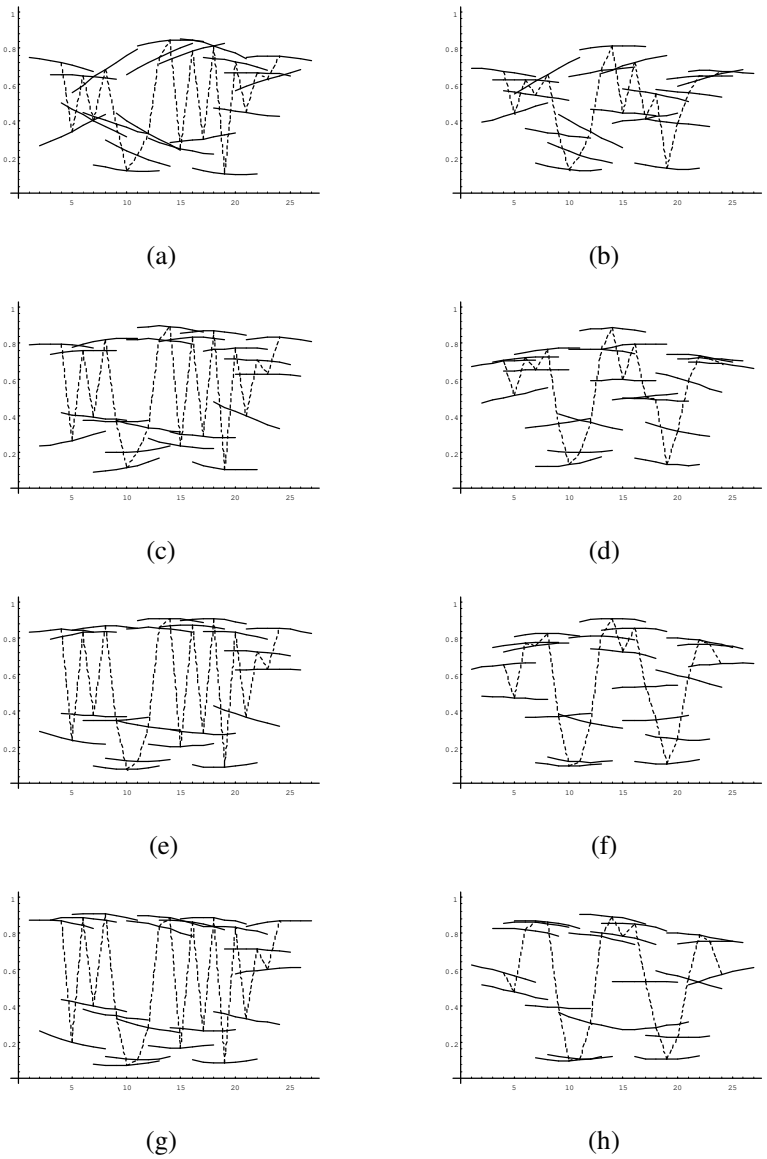


Figure 8: In this simulation the PMD is a 1-dimensional array of 21 nodes with a mixture window width of 7 and an input window width of 7. The likelihood functions are Gaussian, each with a constant variance of 0.1, and a trainable mean whose components are initialised to be random numbers uniformly 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; 4 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.

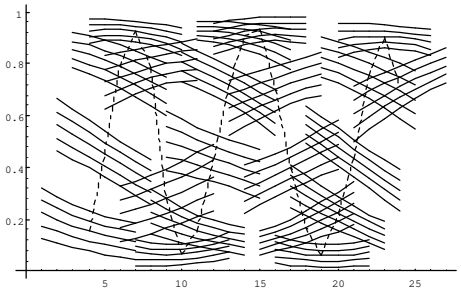
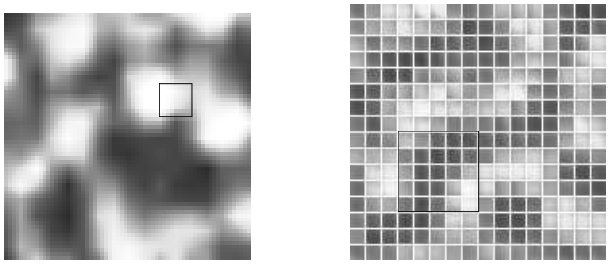


Figure 9: This is the result of 10 iterations of the EM algorithm. The parameters are the same as for the simulation with leakage in Figure 8. In this simulation both the variances and the means are updated. Each mean and variance is shown as 5 curves: mean, $\text{mean} \pm 0.25\sqrt{\text{variance}}$, $\text{mean} \pm 0.5\sqrt{\text{variance}}$.

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 1/4 and 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 occurs because the training data density is greater where the randomly phased sinusoids in the training set have turning points.



(a) (b)

Figure 10: The result of a 2-dimensional PMD simulation. This PMD is a 16×16 array of nodes, each input window is 9×9, and each mixture window is 5×5. (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.

In Figure 10 the result of a 2-dimensional simulation is shown for comparison with Figure 9. The fluctuations appear as they did 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. it exhibits completeness). This is true wherever the square is placed on the montage.

6. Conclusions

Mixture distributions are a natural candidate for approximating PDFs, 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 using mixture distributions to model the PDF of high-dimensional data, and their theoretical properties are a natural extension of those of standard mixture distributions.

Two interesting behaviours of PMDs have been demonstrated in this paper. 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 the level of individual nodes, 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.

7. References

1 Luttrell S P, "An adaptive Bayesian network for low-level image processing", *Proc. 3rd IEE Conf. on Artificial Neural Networks*, Brighton, 1993, 61-65

2. Cox R T, "Probability, frequency and reasonable expectation", *Am. J. Phys.*, 1946, **14**, 1-13

3 Kindermann R and Snell J L, "Markov random fields and their applications", *Contemporary Mathematics*, American Mathematical Society, 1980, **1**

4. Luttrell S P, "The use of Bayesian and entropic methods in neural network theory", *Proc. 8th Int. MAXENT Workshop*, Kluwer Academic Publishers, Cambridge, Skilling J (ed.), 1988, 363-370

5. Luttrell S P, "A trainable texture anomaly detector using the Adaptive Cluster Expansion (ACE) method", *RSRE Memorandum*, 1990, **4437**

6 Luttrell S P, "A hierarchical network for clutter and texture modelling", *Proc. SPIE Conf. on Adaptive Signal Processing*, 1991, **1565**, 518-528

7. Luttrell S P, "Adaptive Bayesian networks", *Proc. SPIE Conf. on Adaptive and Learning Systems*, 1992, **1706**, 140-151

8. Luttrell S P, "Partitioned mixture distributions: an introduction", *DRA Malvern Memorandum*, 1992, **4671**

9 Dempster A P, Laird N M and Rubin D B, "Maximum likelihood from incomplete data via the EM algorithm", *J. Roy. Statist. Soc. Ser. B*, 1977, **39**, 1-37