Dynamic Trees: A Hierarchical Probabilistic Approach to Image Modelling

Nicholas J. Adams

Doctor of Philosophy

Institute for Adaptive and Neural Computation

Division of Informatics

University of Edinburgh

2001

Abstract

This work introduces a new class of image model which we call *dynamic trees* or DTs. A *dynamic tree* model specifies a prior over structures of trees, each of which is a forest of one or more tree-structured belief networks (TSBN). In the literature standard tree-structured belief network models were found to produce "blocky" segmentations when naturally occurring boundaries within an image did not coincide with those of the subtrees in the rigid fixed structure of the network. *Dynamic trees* have a flexible architecture which allows the structure to vary to accommodate configurations where the subtree and image boundaries align, and experimentation with the model showed significant improvements. They are also hierarchical in nature allowing a multi-scale representation and are constructed within a well founded Bayesian framework.

For large models the number of tree configurations quickly becomes intractable to enumerate over, presenting a problem for exact inference. Techniques such as Gibbs sampling over trees are considered and search using simulated annealing finds high posterior probability trees on synthetic 2-d images generated from the model. However simulated annealing and sampling techniques are rather slow. Variational methods are applied to the model in an attempt to approximate the posterior by a simpler tractable distribution, and the simplest of these techniques, mean field, found comparable solutions to simulated annealing in the order of 100 times faster. This increase in speed goes a long way towards making real-time inference in the *dynamic tree* viable. Variational methods have the further advantage that by attempting to model the full posterior distribution it is possible to gain an indication as to the quality of the solutions found.

An EM-style update based upon mean field inference is derived and the learned conditional probability tables (describing state transitions between a node and its parent) are compared with exact EM on small tractable fixed architecture models. The mean field approximation by virtue of its form is biased towards fully factorised solutions which tends to create degenerate CPTs, but despite this mean field learning still produces solutions whose log likelihood rivals exact EM.

Development of algorithms for learning the probabilities of the prior over tree structures completes the *dynamic tree* picture. After discussion of the relative merits of certain representations for the disconnection probabilities and initial investigation on small model structures the full *dynamic tree* model is applied to a database of images of outdoor scenes where all of its parameters are learned. DTs are seen to offer significant improvement in performance over the fixed architecture TSBN and in a coding comparison the DT achieves 0.294 bits per pixel (bpp) compression compared to 0.378 bpp for lossless JPEG on images of 7 colours.

Acknowledgements

I would like to thank firstly my supervisor Chris Williams for his limitless enthusiasm which did so much to keep me motivated, for his excellent guidance, the many good ideas he has contributed, and his conscientious help in that most arduous of tasks of proof reading this thesis. He really has been a magnificent supervisor.

I am grateful to Amos Storkey for his collaborative support on the mean field dynamic tree (Chapter 4), and contributions to the work on the disconnecting tree (Section 5.4).

Thanks are due also to EPSRC for providing the financial support for my studies, BAE SYSTEMS Ltd - Sowerby, for allowing use of their database of images of outdoor scenes, and to my lab colleagues and friends who have helped in so many ways and done so much to keep the level of sanity in the work environment acceptably low.

Lastly but certainly not the least I would like to thank my Father God in heaven who's love and care have sustained me throughout, and for His miraculous provision of solutions when on a number of occasions I felt circumstances had conspired to thwart the continuation of my studies.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Nicholas J. Adams*)

To My Lord Jesus Christ

Table of Contents

Chapter 1

Introduction

Probabilistic modelling of images provides a useful and well grounded framework to conduct inference from images. There has been much interest in this area over recent years, and this has given rise to the development of a rich and varied suite of models and techniques. Numbered amongst these are wavelet models, elastic template matching, and one which has been particularly prominent is that of Markov Random Field (MRF) approaches. They each have their own merits, but also limitations, and a critique of their relative merits and weaknesses along with those of other important work in the field is given in the next Chapter. Of these models Markov Random Field (MRF) models have been very popular. However, one of their main limitations is that inference in a general MRF is NP-hard. They also lack an hierarchical structure and do not provide a multi-scale interpretation of the image – which is highly desirable.

We concentrate on a newly emerging and promising image model called the Tree-Structured Belief Network (TSBN). Tree-structured belief networks provide a natural way of modelling images within a probabilistic framework. By this method a balanced tree-structured belief network is constructed with a single root node and the image is presented at the leaves. Inference can then be conducted by an efficient linear-time algorithm (Pearl, 1988a). Fixed-structure TSBNs have been used by a number of authors as models of images such as Bouman and Shapiro (1994); Luettgen and Willsky (1995); Irving *et al.* (1997); Fieguth *et al.* (1994). They have an attractive

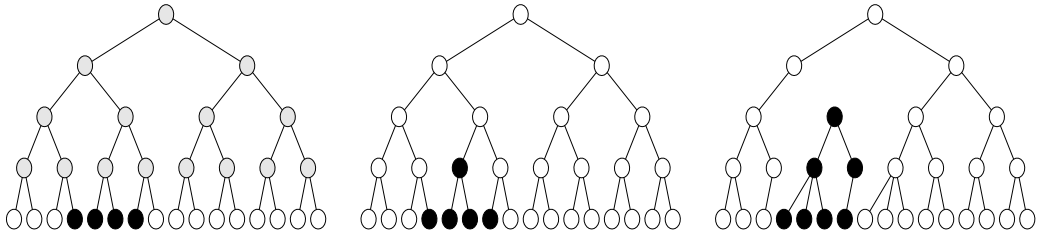


Figure 1.1: (a) “Balanced” tree with image applied (b) resulting segmentation and (c) an example *dynamic tree*.

multi-scale architecture, but suffer from problems due to the fixed tree structure, which can lead to very *blocky* segmentations.

Consider for instance the four level binary tree of Figure 1.1(a). The image applied in the example is a 1-d image of a black bar on a white background, and initially all the other nodes in the tree are uninstantiated. The structure and size of the TSBN directly determines the images it can deal with. The binary tree shown handles 1-d images, but more commonly quad-trees are used which better model the real 2-d images that are of interest.

For the segmentation all the nodes in the network are updated to their most probable state given the model parameters and the image. Since the network is probabilistic in nature we can also ascertain measures of certainty about the state of each node and so have a measure of performance. Our example produces the tree of Figure 1.1(b). A common approach to doing this is by Pearl-style message passing (Pearl, 1988a), which provides an attractive linear-time algorithm as long as the underlying undirected graph does not contain any loops[1].

The hierarchical structure of TSBNs naturally leads to coarser-scale representations of the image at successive levels. As well as providing a natural mechanism for all regions in the image to have some influence over each other and thus exert global consistency, there is also potential for using the segmentations given at higher levels in image coding applications.

[1] In general loopy graphs cannot be solved by linear-time inference, however examples do exist which are collapsible down to a structure that can be solved in linear-time.

However some problems arise when the natural boundaries in the image do not coincide with those of sub-trees in the TSBN. This effect is illustrated by Figure 1.1(b), where the black bar spans two sub-trees with roots at the third level. The resulting segmented images exhibit an undesirable *blockiness* as a consequence. The aim of this work is to attempt to find models which produce good representations of natural images and to use these models to improve on current image segmentation techniques. One strategy to overcome the fixed structure of TSBNs is to break away from the tree structure, and use belief networks with cross connections; see e.g. Dayan *et al.* (1995). However, this means losing the linear-time belief-propagation algorithms that can be used in trees (Pearl, 1988a) and using approximate inference algorithms.

TSBNs have many attractive properties and we believe that models based upon them, but having a dynamically adjustable tree structure to enable their boundaries to better reflect those of the image, provide a very promising starting point. Such models we have named *dynamic trees* (DTs) and one such tree produced for the toy image data is shown in the Figure 1.1(c).

Dynamic trees are a generalisation of the fixed-architecture tree structured belief networks. Belief networks can be used in image analysis by grouping its nodes into visible units $\mathbf{X}_v$, to which the data is applied, and having hidden units $\mathbf{X}_h$, connected by some suitable configuration, to conduct a consistent inference. DTs set a prior $P(\mathbf{Z})$ over tree structures $\mathbf{Z}$ which allows each node to choose its parent so as to find one which gives a higher posterior probability $P(\mathbf{Z}, \mathbf{X}_h | \mathbf{X}_v)$ for a given image $\mathbf{X}_v$. This effectively produces forests of TSBNs.

Allowing this flexibility to chose their structure is not without cost. For useful *dynamic trees* the number of configurations $\mathbf{Z}$ is very large and typically we cannot tractably enumerate over all of them. This thesis will explore the *dynamic tree* model from conception to a real world application of the model.

After a survey of existing models in Chapter 2, the *dynamic tree* model will be introduced in Chapter 3. Sampling and search techniques operating on the true posterior distribution of the DT for conducting inference are considered.

Typically such techniques tend to be computationally intensive and slow to converge. An alternative is to approximate the true probability distribution with a simpler tractable one, and variational methods are proving very popular. Chapter 4 derives the mean field algorithm for the *dynamic tree* which is the simplest of the variational approximations. Learning algorithms for the DT are then developed and assessed in Chapter 5, and in Chapter 6 the *dynamic tree* is used to learn a database of outdoor scenes, where DTs are seen to offer significant improvement in performance over the fixed architecture TSBNs. Chapter 7 concludes this work by summarising the key results obtained and some promising avenues for subsequent research are also exposited.

Chapter 2

Literature Review

Image modelling has naturally been of considerable interest for many years and the extensive research has generated a rich array of modelling strategies. Though some image models work directly on the images, frequently the real-valued colour or grey-scale pixels in the image are mapped to one of a set of discrete label classes. There can be a one to one mapping of pixels to components in the labelled image, but more usually adjacent pixels are grouped together as cliques resulting in a labelled image of lower resolution. A number of techniques exist to do this, for example in Williams and Feng (1998) a neural network is used. Image models are then applied to the labelled image. Whilst the former is important, much work has already been done in this area and our focus is the more interesting task of working on the labelled data.

A comprehensive discussion of all of the modelling strategies would be impractical, so we concentrate on current state-of-the-art technologies. Within this we can group the image models into one of two types, that of fixed and dynamic architectures.

2.1 Fixed Architecture Image Models

Of the fixed architecture models a popular method in image modelling is that of Markov Random Fields (MRF). This class of model has seen widespread use in many areas,

and in image segmentation these models have been used by authors such as Chellappa and Chatterie (1985), Chellappa and Jain (1993) and Jain and Nadabar (1993). In MRF models a neighbourhood relation is firstly defined between pixels or groups of pixels in the image. A random variable designating the state is associated with each element in the image, and at a particular site the state value of this variable is not only given by the class of the pixel(s) to which it relates, but is also conditional on the states of its neighbours. Thus the model considers the local correlations within an image, and produces a Gibbs distribution which can be explored by sampling methods (Geman and Geman, 1984).

Their popularity stems from their principled framework which seeks to develop algorithms based upon sound reasoning rather than ad-hoc heuristics. MRFs also allow inclusion of prior information and can be constructed in a manner so as to ensure the stationarity of the process. However, the nature of MRF algorithms is such that they do not examine global effects directly, but only as a conspiracy of local effects, by virtue of the fact that the state of a node in the model is only influenced directly by those of its neighbours. This enables efficient parallel implementation of the algorithm but leads to a failure to capture explicitly less-localised effects such as a region of sky or a table in an image. MRFs are undirected graphs and non-hierarchical in structure which means that unlocalised or even global information cannot directly be considered. This is clearly disadvantageous in image segmentation.

The Hierarchical Image Probability (HIP) model (Spence and Parra, 2000) uses Gaussian filters arranged hierarchically in a tree structure to repeatedly blur and filter the image from a fine to coarse representation leading to an identification of the object at the root. Though hierarchical, the repeated filtering discards information required to track long range dependencies in the image and extra hidden units are required to model this.

So we see with hierarchical models we can consider the concept of objects in images. Pentland (1989) describes an interesting approach to identifying objects in images, which is claimed to be motivated by biologically plausible mechanisms. Initially filters are applied to the image to create a bank of potential object parts, then these basic

building blocks are combined by a network to produce the simplest, and it is argued by the author, the most likely description of the objects in the image. Whilst novel, the objects produced by the combination of parts can only ever approximate a region in an image and this model is non-hierarchical in nature.

Recently tree-structured belief networks (TSBNs) have been applied to image segmentation. Bouman and Shapiro (1994) and Luettgen and Willsky (1995) provide two such examples. These TSBNs use fixed balanced tree-structures which have the image (or an encoded representation of the image) instantiated at the leaf level of the network and an algorithm which propagates messages of belief about the status of the network to all the other nodes in the tree up to the root. The resulting equilibrium state, which occurs when all nodes have had their beliefs updated, provides the image segmentation. This method is favourable over the MRF approach because TSBNs are directed graphs and hierarchical by design. This hierarchical structure enables the image to be broken down into regions which can in turn be further subdivided and so on down to individual pixels at the leaf level. Thus nodes at different layers in the hierarchy explain variable sized regions of pixels in the image, and enables global effects to be directly considered unlike in MRFs.

Attractive linear time inference algorithms also exist to perform the belief propagation (Pearl, 1988b) required for conducting inference in TSBNs, giving TSBNs a significant advantage over MRFs whose inference is NP-hard. Other computationally attractive algorithms do exist for structures of this form such as the Laplacian Pyramid of Burt and Adelson (1987) used in image compression, but they are non-probabilistic in nature.

Belief networks are not new to the field of image modelling. Utans (1994) proposes a method of growing tree structures upwards from the data. He uses a predetermined configuration of nodes which are each assigned to a given layer to provide the model framework. Then an incremental algorithm working up the layers is employed to find correlations between child nodes and connect related children to the same parent, with an EM algorithm being used to learn the parameters for the nodes. The result is essentially a hierarchical clustering model, and the toy example of dot clusters considered

was too trivial to properly assess the model's viability. It appears to be a somewhat computationally intensive method, and the example could have been efficiently solved by a simple clustering algorithm.

The fixed structure of TSBNs can often lead to *blocky* segmentations which occur when the segmentation boundaries do not correspond with those of the quad-structure subtree components of the network. Dayan *et al* (1995) and Bouman and Shapiro (1994) both attempt to overcome this by proposing a cross-connected network architecture. Bouman and Shapiro overlap the subtrees in their model so that nodes on the boundaries also connect across subtrees – in addition to their normal children in the quad-tree arrangement – to produce an augmented pyramidal configuration. This means that some nodes have multiple parents and in order to maintain computability of their algorithm they make the simplifying assumption that at a particular scale (layer) all of the nodes below adopt a quad-tree structure. The effect of the cross-connections is to enforce smoothing across the boundaries of the subtrees, and so diminish the "blockiness". However such a scheme sacrifices the linear-time belief propagation algorithm which makes the TSBN so attractive.

Another approach to this problem by Irving *et al* (1997) uses an overlapping tree structure in which pixels which fall on the boundaries dictated by the tree structure are duplicated in the adjacent subtrees. By this method the duplicate pixels are weighted such that their total effect sums to unity and so the boundaries between adjacent subtrees are smoothed. As in the cross-connected networks this is at the expense of the fast inference algorithms.

Bouman and Shapiro (1994) and Irving *et al.* (1997) above hold firmly to fixed network architectures, which are essentially balanced tree-structured belief networks with slight modifications that attempt to reduce the "blocky" effect. Though "blockiness" is reduced by these strategies they are still susceptible to image translation.

It is interesting to note that TSBNs bear some similarity to another class of hierarchical model based on wavelets, of which De Bonet *et al.* (1998), De Bonet and Viola (1998) and Istas (1995) are examples. Inference is straightforward in these models as they use a complete basis of wavelets (c.f. over-complete bases of wavelets where inference is

not straightforward); however though the models are hierarchical, each level is a scaled version of the source image, without any representation of "objects", and there is no robustness to image translation.

We now consider dynamic architecture models and discuss their pros and cons over the fixed architecture models described above.

2.2 Dynamic Architecture Image Models

Models whose architecture can be dynamically altered provide a powerful means of addressing the image translation issue. The distinction between these and the fixed architecture models is that dynamic models don't merely instantiate hidden variables in a fixed structure conditionally on the supplied data, but they seek also to find relationships between substructures of these variables and are able to modify these relationships on the fly should subsequent data demand it.

Such issues in images have been considered by von der Malsburg (1988) who proposed a model of deformable templates called the Dynamic Link Architecture (DLA). In this model the input image triggers feature detector cells which are then elastically matched to templates residing in cells in the layer above, by exploring dynamic links between the two layers. The templates are labelled graphs of objects expected to be found in the image, and the task is essentially one of labelled graph matching. Though dynamic in architecture this model is non-hierarchical with all object templates residing in a single layer. Implementations based around this framework includes Lades *et al.* (1993) who use it for the task of face recognition.

Bienenstock and Doursat (Bienenstock and Doursat, 1994, 1990) also proposed a shape recognition model inspired by von der Malsburg's dynamic link theory. Their model assesses the quality of a match between two images by assessing the cost of elastically deforming an image to the prototype image of a particular class. They consider the task of handwritten digit recognition and show good generalisation results for training set sizes of up to 1100 examples. The results quoted by these authors were all positive;

however the DLA model has a complex structure of labelled graphs, is computationally expensive, and foreknowledge of the objects likely to appear in the image is required. This makes the model poor at generalising to new images where a previously unseen object might arise.

Hummel and Biedermann (1992) alternatively use a 7-layer ANN model inspired by biology to try and solve the dynamic binding paradigm. Their model takes line drawings as an input and attempts to output a classification of the object it is depicting, but as for the DLA model it cannot generalise to novel, previously unseen objects.

For images with objects that take on specific forms and with a discrete set of classes techniques such as these based on deformable templates or dynamic binding may be viable. For image segmentation where like regions or objects could take on very different forms in similar images, and in others may not even be present, a template strategy would appear to be too inflexible.

Geman and Geman (1984) introduce line processes as an extension of the basic MRF approach. These line processes (which also form an MRF) occupy the boundaries between pixels, and the connection of a number of these segments together produced the regions. Line processes are dynamically combined as edge elements which describe the boundaries between regions in the image. They perform this within a Bayesian framework and apply the model to an image restoration problem. This is an interesting model, but it still suffers from the disadvantages of MRF approaches in that inference is NP-hard.

A promising alternative to the above is to allow a certain flexibility to the network architecture such that the subtree elements can be constructed in a way that their boundaries correspond directly to the natural boundaries in the image, producing unbalanced TSBNs. It is anticipated that such models would have the flexibility required to alleviate the “blockiness” experienced by balanced TSBNs and be invariant to image translation. Unbalanced tree structures have already been used by other areas of image modelling such as Montanvert *et al.* (1991) and Meer and Connelly (1989) with an algorithmic approach. Modelling such structures within a Bayesian framework seems very attractive. Maintaining a tree structure without cross-connections would further allow the use of the attractive linear-time inference algorithms of Pearl.

Some work applying a hierarchical Bayesian model to images has already been undertaken by Utans and Gindi (1993). Their motivation was object recognition and, like von der Malsburg, sought a means of identifying particular objects in an image independent of position, scale and rotation. These may appear anywhere in the image and the task was to identify and match them against a previously constructed model of a single instance of the object. They simplified the problem to the simultaneous labelling of the parts of a single object (a gingerbread man) alongside the determination of its parameters. The hierarchy consists of three layers, with a single object match neuron at the top level connected to the second level object subparts whose labelling is based on the stored object structure. The lowest level neurons identify the individual components of the object in the given image, and the algorithm then tries to automatically label and connect these neurons to parents in the layer above. As in von der Malsburg (1988), prior knowledge of the object is required and generalising the approach to scenes containing multiple objects, including unknown and occluded types is a very difficult task.

More recently Bayesian networks have been used by Bienenstock *et al.* (1997). They use object primitives to model complex scenes, but these primitives are also handcrafted and it is not clear how the model would generalise to novel scenes.

For segmentation we are interested in identifying and labelling the regions within an image which can be viewed as multiple objects of indeterminate size and shape, and the above approach (just as with von der Malsburg's templates in the DLA) would seem inappropriate due to the large variability of image regions.

There is an analogy between the structural form of TSBNs and Context Free Grammars (CFGs). Sampson (1996) conducts a search over parse trees of natural language sentences in an attempt to learn their grammar. Parse trees of these grammars are very similar in form to what we might expect by allowing dynamic architectures in tree-structured belief networks. An extension to standard CFGs is to add a stochastic element, and Chou (1989) and Geman and Manbeck (1994) use such models.

To pre-empt the work of Section 3.4.1 we note that as well as having dynamic architectures between a fixed number of units it might also be desirable to vary the number of units. Hinton *et al* (1998) adopt such a strategy with their "hierarchical community

of experts", where-in the participation or non-participation of a unit in the model is determined by a gating function at its output. They use a directed graph model which is fully connected between successive layers, which means the Pearl inference algorithm can not readily be adopted as the graphs contain multiple paths. So too Hinton *et al*. (2000) also use general DAGs for their credibility networks. In Ghahramani and Hinton (1998) the switchable units are fully fledged models, and Ghahramani *et al* use a Markovian process to select which model to use at each interval of time.

Although not an image model the approach of Geiger and Heckerman (1996) is also of importance. As for the *dynamic tree* model (to be described in the next Chapter) Geiger and Heckerman use the Bayesian formalism to set a prior over their network's architecture. In a class of model they call similarity networks they represent their data as a collection of Bayesian sub-networks. While cleverly using known conditional independencies to reduce parameterisation the model is fully knowledge based requiring an expert to assign each node in the graph both function and class labels.

Chapter 3

The Dynamic Tree

3.1 Introduction

This chapter introduces the *dynamic tree* model which underpins the core work of this thesis. *Dynamic trees* are probabilistic where-by solutions are automatically assigned a probability of certainty by the model. This makes them more robust than non-probabilistic models as in addition to providing solutions it also gives an indication of certainty about them. *Dynamic trees* are also hierarchical models allowing a multi-scale representation of images which can assist in finding better solutions and also suggests possibilities for compression and image coding. They are essentially mixtures of tree-structured belief networks implemented within a Bayesian formalism which allows us to change their architecture.

As well as introducing the model this chapter explores its capabilities investigating firstly its ability to handle translation of the image at which fixed architecture TSBNs ordinarily perform badly; secondly its generative capabilities are assessed to see whether it can generate similar features to those one would ordinarily expect to see in real images. The assumption here is that if the model naturally generates such features then we would hope it to perform well in inference on real life images. Finally a more comprehensive study of the types of distributions the *dynamic tree* is naturally suited to is undertaken, before finally some treatment is given to possible enhancements to the basic model formalism.

Section 3.2.1 describes the model theory and explains how inference can be conducted in *dynamic trees*. Then follows experimentation on small toy 1-d 16 pixel example images which show the *dynamic tree* to have improved performance over Tree Structured Belief Networks against "blockiness" in Section 3.2.2.1. Section 3.2.2.2 then explores the generative capabilities of *dynamic trees* and assess qualitatively whether the *dynamic tree* ordinarily generates the sort of images one might see in real life. This work is done on larger 16×16 pixel 2-d images. Sampling and simulated annealing is used to find maximum a posteriori (MAP) solutions of the generated images and this serves to give good insights into the inferential characteristics of the model[1].

An exploration of the *dynamic tree* parameter space is made in Section 3.3 to quantitatively explore the circumstances over which *dynamic trees* are preferable to the best fixed structure choice and attention is given to potential improvements to the basic model in Section 3.4[2].

3.2 The Dynamic Tree Model

3.2.1 Dynamic Tree Theory

3.2.1.1 Bayesian Overview

It was noted earlier that probabilistic modelling of images provides a useful and well grounded framework from which we can conduct inference from images. The MRF approach was seen to suffer the drawbacks that it is non-hierarchical in nature and so considers only local effects in images and also that inference is NP-hard. A more recent approach is the use of tree-structured belief networks (TSBN) whose hierarchical structure and linear time inference algorithms (Pearl, 1988b) provide an attractive alternative, but these were seen to have disadvantages in that they produce "blocky" segmentations.

[1] The material of these Sections has been published in Williams and Adams (1999).
[2] This work has been published in Adams and Williams (1999).

The aim of the work being undertaken is to develop an image model which will improve upon the segmentation performance of existing techniques. TSBNs have many desirable qualities, outlined in Chapter 2 earlier, and a promising direction would be to utilise these strengths and seek to overcome their limitations. The key strategy to adopt here is to develop mechanisms for generating tree-like belief network structures that are more malleable and able to adapt to real images. *Dynamic trees* seek to adopt these strategies and it will be seen that they can be viewed as as a collection or "forest" of TSBNs. To do this the task is reformulated within a Bayesian framework.

The structure of the DT can be described by indicator variables z_i which give the index of the parent node for child node i. Encapsulating the z_i's for each node of the DT in a single vector $\mathbf{Z}$ thus gives a compact description of the DT configuration, and by this method orphan children can be readily accommodated by setting their z_i to a value that does not reference any nodes in the network.

To fully describe the DT model two other parameters are required. These are the prior P which sets the prior probability of a root node being in a particular state, and the Conditional Probability Tables (CPTs) θ defining the state transition probabilities between connected nodes. These are both discussed in Section 3.2.1.4. They remain fixed during the following analysis and are omitted for clarity of description. The DT model is referred to as $\mathbf{Z}$ and it is understood that there will be a number of fixed Ps and θs associated with it.

For such a model $\mathbf{Z}$ with an observed image vector $\mathbf{X}_v$ Bayes' rule gives the following

$$P(\mathbf{Z}|\mathbf{X}_v) = \frac{P(\mathbf{Z})P(\mathbf{X}_v|\mathbf{Z})}{P(\mathbf{X}_v)} \tag{3.1}$$

A prior $P(\mathbf{Z})$ is defined to describe the probability of a particular set of links between an arrangement of nodes. The details behind this are more fully explained in Section 3.2.1.2.

The likelihood $P(\mathbf{X}_v|\mathbf{Z})$ of observing the image given the model can be readily calculated for tree structured belief networks. Pearl (1988b) derives a scheme for instantiating the nodes of belief networks with probabilistic beliefs of being in a particular state

by a mechanism of message passing, and Neapolitan (1990); Castillo *et al.* (1997) give a computationally efficient implementation of the theory. This is described in Section 3.2.1.4 and Appendix A shows how this is adapted to calculate the likelihood.

$P(\mathbf{X}_v)$ can be viewed as a normalising constant such that the sum of the posterior probabilities is unity. That is $P(\mathbf{X}_v) = \sum_i P(\mathbf{Z} = \mathbf{Z}_i)P(\mathbf{X}_v|\mathbf{Z} = \mathbf{Z}_i)$.

Equation (3.1) gives the posterior probability which is a measure of how likely the observation $\mathbf{X}_v$ belongs to the class of model defined by $\mathbf{Z}$. Strictly speaking the posterior above is the posterior marginal as the model contains hidden units which take on the values $\mathbf{X}_h$. We concentrate on the posterior marginal as opposed to the full posterior $P(\mathbf{Z}, \mathbf{X}_h|\mathbf{X}_v)$ as knowledge of the distribution $\mathbf{X}_h$ is not currently required explicitly, and $P(\mathbf{X}_h|\mathbf{X}_v, \mathbf{Z})$ is readily obtainable from the posterior marginal by standard techniques if desired.

Thus there are two essential components that make up a *dynamic tree* network (i) the tree architecture – governed primarily by the prior – and (ii) the nodes and conditional probability tables (CPTs) in the given tree – which come into play in the likelihood calculation. In Sections 3.2.1.2 and 3.2.1.4, the prior and likelihood calculations are fully described and this is tied together with a discussion of inference in tree-structured belief networks in Section 3.2.1.5.

3.2.1.2 Specifying the Prior

The role of the prior is to encode our knowledge and intuitions as to what might constitute a good or bad model such that structures that are considered to be better will have a higher prior probability and be seen more frequently.

For image segmentation there are a number of constraints which we need to impose on belief networks. These are firstly that there will be a fixed number of units that will be instantiated in such a way that they capture the (real valued) pixel data of the image, and secondly in order to use the linear time inference algorithms of Pearl (1988b) there should be no loops[3]. TSBNs have been used previously for image segmentation

[3] In general loopy graphs cannot be solved by linear-time inference, however examples do exist which are collapsible down to a structure that can be solved in linear-time.

as the leaf nodes provide a natural means of introducing an image and the tree structure contains no loops. They also have the advantage that successive layers of the tree each provide a coarser abstraction of the image which enables regions or "objects" in the image to be detected by single nodes that will become the roots of subtrees spanning the object. The level in the tree that such nodes occur will depend on the size of the object in the image (that is the number of pixels it spans).

Clearly in specifying the prior we wish to encourage the formation of tree structures. However the "balanced" architecture used in Bouman and Shapiro (1994) and Luettgen and Willsky (1995) is not flexible enough.

It is therefore necessary to devise a mechanism for automatically adjusting the tree architecture to facilitate the best possible image segmentation (ones with the smoothest possible boundaries) and models so produced we call *dynamic trees* (DTs).

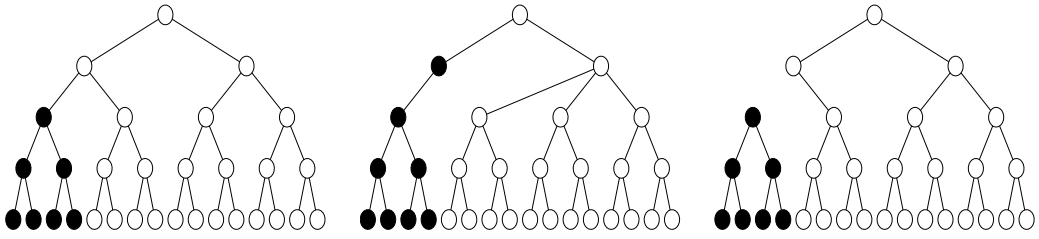


Figure 3.1: (a) "Balanced" tree (b) "unbalanced" tree and (c) "disconnection".

Variations to the balanced tree architecture will give rise to two basic tree forms illustrated in Figure 3.1 above. The DT will be composed of varying numbers of these two types of operation.

Unbalancing of the tree occurs by moving a link from the "natural" parent to a different parent such as in Figure 3.1(b). This allows subtrees to grow or contract to more accurately fit the object in the image segment they are covering.

Allowing some nodes to totally disconnect from all parents and become a root (Figure 3.1(c)) enables groups of pixels in an image to be modelled by a single tree. Thus the single tree network will become a "forest" of trees. This is advantageous as the network will be able to group and identify regions of the image as distinct "objects".

Introducing flexibility over structure however is not without cost. It gives rise to an exponentially increasing (with tree depth) number of configurations (Section 3.2.1.5), and in order to exactly calculate the posterior of Equation (3.1) all have to be enumerated. An obvious simplification to the model might be to consider one layer at a time. Firstly we can marginalize out $\mathbf{Z}$ to create a fully connected belief network. Then by enumerating over all of the states of all of the nodes on a layer we can reduce the layer to a single node, and thus the *dynamic tree* model becomes a Markov chain. However if there are C states for a node and n_d nodes on a layer at depth d this computation is of $O(C^{n_d})$, which is exponential in the number of nodes on the layer and still intractable.

It is possible that the summing over trees may well be an NP-complete problem, although to show this formally it is necessary to find a transformation to a problem that is already known to be NP-complete. (Garey and Johnson (1979) and Gibbons (1988) provide a good introduction to NP-completeness theory.)

The next Section describes one formalism which allows nodes to chose different parents or disconnect, while still enforcing the desirable tree structure. Section 3.2.2.1 then discusses the results obtained by this method assessing its merits and weaknesses.

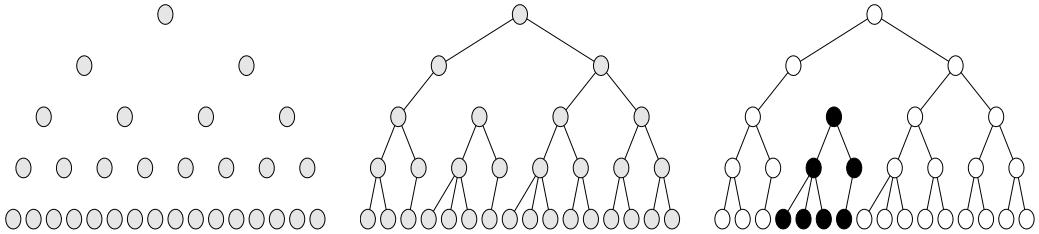
3.2.1.3 DTs: The *Full-Time-Node-Employment* Prior

Consider a number of nodes arranged into layers, as in Figure 3.2(a). Define $\mathbf{z}_i$ to be the indicator vector showing to which parent node i belongs, then the matrix $\mathbf{Z}$ whose columns are the individual $\mathbf{z}_i$ vectors will specify the tree structure.

The simplest scheme for the prior is then to assume that each link is independent, so that the prior for the tree, $P(\mathbf{Z})$, will simply be a product of the probabilities for each link

$$P(\mathbf{Z}) = \prod_i P(\mathbf{z}_i) \tag{3.2}$$

We wish to generate tree-like structures and create hierarchical models such that nodes on a given level will act as detectors for objects of a given size. For this reason a child



(a) (b) (c)

Figure 3.2: (a) "Naked" nodes, (b) a sample from the prior over $\mathbf{Z}$, (c) data generated from the tree in (b).

node is restricted to only being allowed to connect to nodes in the layer above. Each layer also contains a null parent such that any child connected to it becomes a root of its own tree.

Currently $P(\mathbf{z}_i)$ has been specified as follows. A child node in a given layer is assigned an "affinity" for connecting to each of the nodes, including the null parent, in the parent layer above. Child nodes are considered to have a "natural" parent which is the parent it would have had if it was in a balanced tree structure such as Figure 3.1(a). In assigning the affinities the present approach has been to give the highest value to the "natural" parent and decrease it steadily with increasing distance of the potential parent from the "natural" parent. Null parents are assigned an affinity in accordance with the desirability of there being roots at a particular level of the network.

Denoting this affinity for node k in the parent layer as a_k then the following provides a convenient probability measure

$$P(\mathbf{z}_i = \mathbf{e}_k) \quad = \quad \frac{\exp(\beta a_k)}{\sum_{j \in Pa_i} \exp(\beta a_j)} \tag{3.3}$$

with $\beta > 0$, and $\mathbf{e}_k$ the unit vector with a 1 in position k. This is essentially the *softmax* function (Bridle, 1990) which has found frequent use in neural networks. Figure 3.2(b) shows a DT generated by this method and Figure 3.2(c) data generated from this tree. We have named this prior the "*full-time-node-employment* prior" as all the nodes participate in the creation of the tree structure to some degree. (In Section 3.4 a procedure where all nodes need not participate will be discussed.)

3.2.1.4 Calculating the Likelihood

Given that we have specified a prior strategy over network architectures and generated a "forest" structure attention can now be given to calculating the likelihood of the network.

The nodes in a TSBN take on discrete values and can be considered as C-class multinomial random variables. On each of the links there is a conditional probability table (CPT) θ, which defines the probability of changing state during a transition between nodes, and for the root node there is a prior P.

DTs use this same methodology, however they can contain more than one tree and potentially have a root at any level. Thus priors need to be set for each node in the network.

At Appendix A the likelihood of observing the data $\mathbf{X}_v$ from a TSBN with a single root node U is derived. For the $N > 0$ roots in a DT the total likelihood is simply a product of all the likelihoods of its component trees, i. Some of these probability values may well be small, so for practical implementations the log-likelihood is used. The *dynamic tree* log-likelihood is thus given by

$$\begin{aligned} \log(P(\mathbf{X}_v|\mathbf{Z})) &= \log\Big\{\prod_i \sum_j P(\mathbf{U}_i^{(j)})\lambda(\mathbf{U}_i^{(j)})\Big\} \\ &= \sum_i \log\Big\{\sum_j P(\mathbf{U}_i^{(j)})\lambda(\mathbf{U}_i^{(j)})\Big\} \end{aligned} \tag{3.4}$$

with $j = 1 \ldots C$, $P(\mathbf{U}_i^{(j)})$ is the prior of root node $\mathbf{U}_i$ being in state j, and where $\lambda(\mathbf{U}_i)$ is the Pearl-style message vector that the root $\mathbf{U}_i$ receives. In this vector is a summary of the states of all the child nodes in the sub-network of which $\mathbf{U}_i$ is the root.

3.2.1.5 Inference on Images

Inference in DTs on images is a matter of obtaining the posterior $P(\mathbf{Z}, \mathbf{X}_h|\mathbf{X}_v)$ of the tree with structure $\mathbf{Z}$. Recall from the discussion of Section 3.2.1.1 that we prefer

to find the posterior marginal $P(\mathbf{Z}|\mathbf{X}_v)$, and an application of Bayes' formula gave Equation (3.1).

The prior and likelihood have been dealt with above. The other element in the formula, $P(\mathbf{X}_v)$, is a normalising term which is independent of $\mathbf{Z}$. Having no knowledge about the prior over images $P(\mathbf{X}_v)$, it is obtainable only by summing $P(\mathbf{Z})P(\mathbf{X}_v|\mathbf{Z})$ over all models.

For a set of nodes created from balanced tree with branching factor b and depth D it is easy to show that the total number of forest structures T, is given by

$$T = \prod_{d=2}^{D} (b^{(d-2)} + 1)^{b^{(d-1)}} \tag{3.5}$$

This is exponential in tree depth and the total number of structures rapidly becomes large and a naive evaluation over all configurations is clearly intractable. We have also observed in Section 3.2.1.2 that the reduction of the model to a belief network that has full layer to layer connections though offering a tremendous reduction in complexity still results in something which is intractable, so we cannot normalise the posterior.

So we seek to obtain samples from the posterior marginal $P(\mathbf{Z}|\mathbf{X}_v) \propto P(\mathbf{Z})P(\mathbf{X}_v|\mathbf{Z})$, and for large dimensional spaces, such as this, Markov Chain Monte Carlo methods (MCMC) provide a means of accomplishing this. This is possible because the two components $P(\mathbf{Z})$ and $P(\mathbf{X}_v|\mathbf{Z})$ are readily evaluated. $P(\mathbf{Z})$ is the prior defined by Equation (3.2) and $P(\mathbf{X}_v|\mathbf{Z})$ can be computed from Equation (3.4).

MCMC is used to construct a Markov chain whose equilibrium distribution is the desired $P(\mathbf{Z}|\mathbf{X}_v)$. Two popular ways of doing this are Gibbs sampling and the Metropolis algorithm, and there exists a wealth of information about these methods for the interested reader in sources such as Geman and Geman (1984); Neal (1993); Gelman *et al.* (1995). In the initial work described in Section 3.2.2.1 the Gibbs sampler was used, so this is discussed briefly.

Gibbs sampling considers each variable in the joint distribution separately and sam-

ples them all[4] in turn conditioned on the other variables in the distribution. For a DT structure of N nodes the algorithm is as follows

For n = 1 **To** Number of iterations
  For i = 2 **To** Number of nodes in structure
    Sample from $P(\mathbf{z}_i|\mathbf{Z}_{-i}, \mathbf{X}_v)$
  Next i
Next n

where $P(\mathbf{z}_i|\mathbf{Z}_{-i})$ denotes the state of all the links in the DT *except* that from the ith node to its parent.

In common with the other MCMC algorithms Gibbs sampling has a period of "burn-in" where the distribution does not reflect that of the one we are sampling from. It is difficult to establish at which point during the sampling that the equilibrium distribution is reached and typically the first half of the chain is discarded. There is also no firm method for determining when the chain has converged. For a discussion of these issues see Cowles and Carlin (1996). Gibbs sampling is also computationally expensive, and even in long runs there is no guarantee that we will find high posterior trees, which are of interest.

In the 2-d image model experiments discussed in Section 3.2.2.2 we shall be solely interested in finding the maximum a posteriori (MAP) state from the posterior, so simulated annealing will be adopted in preference to Gibbs sampling as it is geared towards finding such solutions. It also is less computationally intensive than Gibbs sampling as we only need to propose new configurations and decide whether or not to accept them, rather than evaluate all of the alternatives for a particular $\mathbf{z}_i$ given the other parameters and probabilistically choose one. This is advantageous for the much larger search space of the 2-d models.

With simulated annealing there is the issue of what is the best annealing schedule to adopt. Much work in the literature has already explored this (see for instance Stander

[4]The first node in the tree is not considered in Gibbs sampling as it is the root node and will always be connected to the Null parent.

and Silverman (1994) and Rees and Ball (1987)). In practice the traditional exponential schedule was found to be adequate for our requirements. It may well be that more optimal schedules exist for the DT, but this is an open research issue.

3.2.2 Dynamic Tree Experiments

In this Section we describe two experiments conducted on the *dynamic tree* models. The first has been designed to compare the translation performance of DTs with that of the balanced TSBN structure and is described in Section 3.2.2.1. In Section 3.2.2.2 we generate 2-d images from the DT model, find the MAP *dynamic tree* for these images, and contrast their performance relative to the balanced TSBN.

3.2.2.1 Comparing DTs with the Balanced TSBN

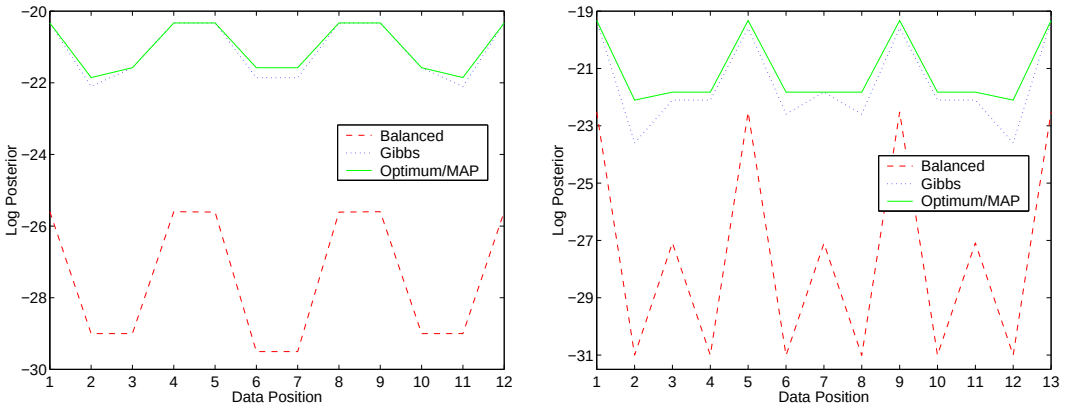
We consider a 5-layer binary tree with 16 leaf nodes, as shown in Figure 3.2. Each node in the tree is a binary variable, taking on the values of white/black.

The priors P_l, CPTs θ_l, and affinities were set to be the same at each level, and a global value of β covered the whole network. A brief exploration of the parameter space by hand gave rise to the following values, $P = (0.75, 0.25)$ with 0.75 referring to white, and θ had values of 0.99 on the diagonal and 0.01 off diagonal. We shall be dealing with binary images of black "objects" on a white background and the prior was set to reflect the fact that the images are largely white with smaller black regions in them. The strongly diagonal CPTs favour parent nodes being in the same state as their children. The affinities[5] were set as 1 for the natural parent, 0 for the nearest neighbour(s) of the natural parent, $-\infty$ for non-nearest neighbours and $a_{null} = 0$, with $\beta = 1.25$.

To illustrate the effects of translation, we have taken a stimulus made up of a bar of five black pixels, and moved it across the image. At each position the unnormalised log-posterior $\log P(\mathbf{Z}) + \log P(\mathbf{X}_v|\mathbf{Z})$ was calculated for the balanced TSBN[6] architecture

[5] The affinities are defined up to the addition of an arbitrary constant.

[6] Strictly speaking the notion of a prior over a fixed architecture is meaningless, but this was necessary in order to make a fair comparison with the DT whose posterior cannot be normalised. The prior was set up so that the balanced TSBN configuration had the highest probability so that it was not penalised by this action.



(a) 5 black nodes (b) 4 black nodes

Figure 3.3: Plots of the unnormalised log-posterior vs position of the input pattern for (a) the 5 black-nodes pattern and, (b) 4 black-nodes pattern.

and compared to the highest value found for the DT after conducting a search over $\mathbf{Z}$. To gauge the effectiveness of the Gibbs sampling "optimal" DT structures were found by hand for each image using our intuitions as to what should be the best tree. Note that due to symmetries there are in reality fewer than 12 distinct configurations. Figure 3.3(a) shows clearly that the balanced TSBN is a poor model for this stimulus, and that much better interpretations can be found using DTs, even though the "natural parent" idea ensures that the $\log P(\mathbf{Z})$ is always larger for the balanced tree.

The x-axis in the plots denotes the position of the black bar in the image, and the y-axis the log-posterior probability. Due to the large number of different tree structures possible it is intractable to normalise the *dynamic tree* posterior so we cannot average over all $\mathbf{Z}$s to ascertain the constant $P(\mathbf{X}_v)$ (Equation (3.1)). So we compare the MAP configuration $\mathbf{Z}_{MAP}$ against the balanced-tree treating the latter within the same framework. The definition of $P(\mathbf{Z})$ is such that the balanced-tree model scores highest so for any other configuration to beat this it must have a higher likelihood. Clearly it can be seen that the balanced TSBN is a poor model of translated images compared to DTs, even despite the fact that the "natural" parent notion ensures that $P(\mathbf{Z})$ is *always* higher for the balanced TSBN.

Notice also that the balanced TSBN displays greater sensitivity of the log-posterior with respect to position than the DT model. This is readily explained by the fact that during translation of the bar the black pixels span differing height subtrees in

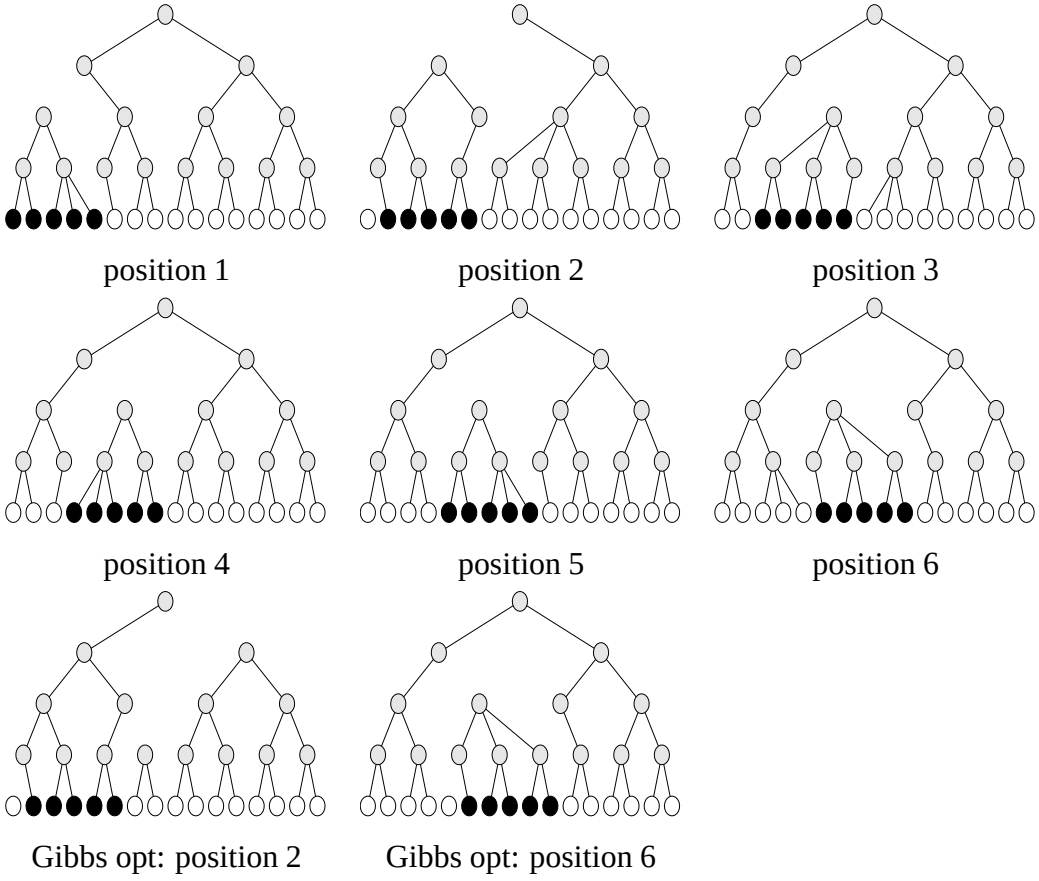


Figure 3.4: The optimal configurations for positions 1 to 6 for 5-black-nodes stimuli, and the two sub-optimal configurations found by Gibbs sampling.

the balanced TSBN leading to varying levels of uncertainty about the states of the hidden units in the affected subtrees. The DT model overcomes this by reorganising its structure so that the black image forms its own tree.

For the Gibbs runs four chains each constructed from random starting positions were run over 25,000 sweeps, and the DT found with the largest posterior is the one quoted in the results. It can be seen from Figure 3.3 that Gibbs sampling found the optimal or a sub-optimal solution near the optimum in each case. (Typically the log-posterior values varied between the -20s and -70s). The optimal DT architectures for positions 1 to 6 are shown in Figure 3.4, along with the slightly sub-optimal architectures discovered by Gibbs sampling for positions 2 and 6. Notice how the high posterior structures do indeed pick out the black "objects" in the image as separate trees.

In Figure 3.3(b) we have shown the log-posterior for a stimulus made up of *four* black

nodes[7]. In this case the balanced TSBN is even more sensitive to the stimulus location, as the four black nodes fit exactly under one sub-tree when they are in positions 1, 5, 9 or 13. By contrast, the *dynamic tree* is less sensitive to the alignment, although it does retain a preference for the configuration most favoured by the balanced TSBN. This is due to the concept of a "natural" parent built into the (current) architecture (see Section 3.5 for further discussion).

Clearly these results are somewhat sensitive to settings of the parameters. One of the most important parameters is the diagonal entry in the CPT. This controls the relative desirability of having a disconnection against a transition in the tree that involves a colour change. For example, if the diagonal entry in the CPT is reduced to 0.95, the gap between the optimal and balanced trees in Figure 3.3(b) is decreased. We have experimented with CPT entries of 0.90, 0.95 and 0.99, but otherwise have not needed to further explore the parameter space to obtain the results shown.

3.2.2.2 Generating from the Prior and Finding Maximum a Posteriori Trees

We now wish to investigate the generative capabilities of the DT. The 16 pixel 1-d images used above are insufficient for this task so we turn our attention to 2-d images. Considering a 5 layer quad-tree node arrangement gives a total of 256 leaf nodes or a 16x16 pixel image. A structural plot of such a tree generated from the prior is shown in Figure 3.5.

Each sub-plot is a slice through the tree showing the nodes on successive levels. The boxes represent a single node on the current level and their shading indicates the tree to which they belong. Nodes in the parent layer above are superimposed as circles and the lines emanating from them shows their connectivity. Black circles with a smaller white circle inside are used to indicate root nodes. Thus in the example above we see that the forest consists of four trees, three of whose roots lie at level 3 (which between them account for most of the black in the image, Figure 3.5(f)), while the root node at level 1 is responsible for the background.

[7]The parameters are the same as above, except that a_{null} in level 3 was set to 1.0 to encourage disconnections at this level, where the nodes cover four pixel sections of the image.

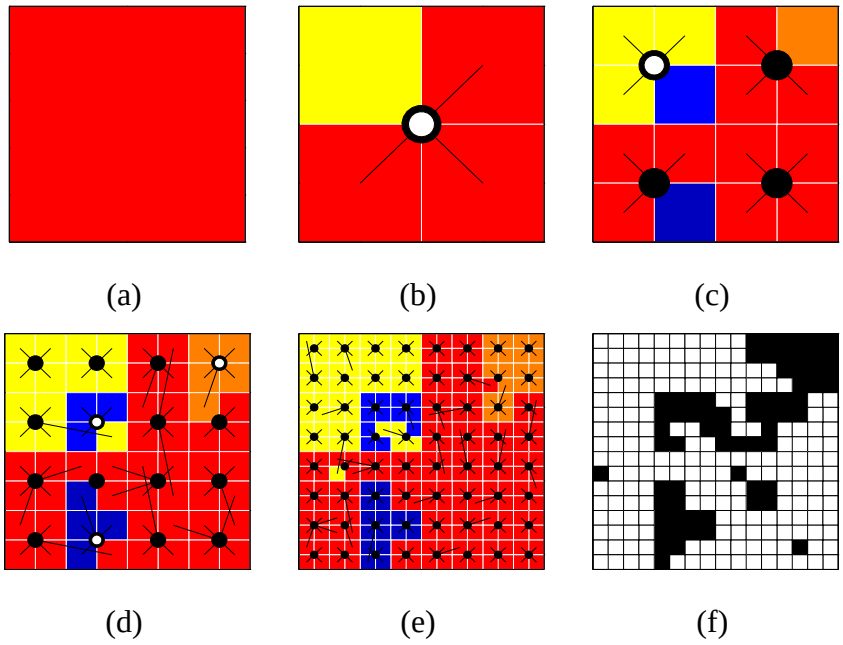


Figure 3.5: Plot of the generative *dynamic tree* and accompanying image (f). Each colour in the plots (a)–(e) represents a single tree, and the image (f) has black and white pixels.

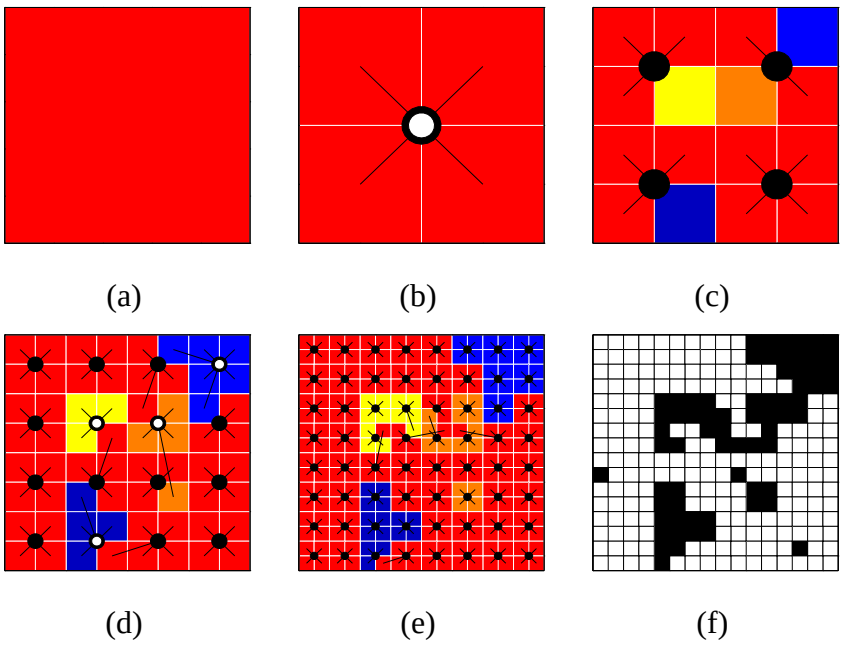


Figure 3.6: Plot of the MAP *dynamic tree* of the accompanying image (f). Each colour in the plots (a)–(e) represents a single tree, and the image (f) has black and white pixels.

Level	Affinities					Transition probabilities	
	a_{nat}	a_{nn}	a_{other}	a_{null}	β	P	CPT diagonal
1	1.0	0.0	-1 -2 . . .	-0.5	1.5	0.9	0.99
2	1.0	0.0	-1 -2 . . .	0.25	1.5	0.9	0.99
3	1.0	0.0	-1 -2 . . .	0.5	2.5	0.25	0.99
4	1.0	0.0	-1 -2 . . .	-0.5	2.5	0.9	0.99
5	1.0	0.0	-1 -2 . . .	-9999.0	3.5	0.9	0.99

Table 3.1: 2-d *dynamic tree* model parameters. The ordering is from the root of the tree at level 1 down to the leaves at level 5.

Broadly speaking the parameters for the 2-d DTs were set to be similar to the 1-d trees of the previous section, except that the disconnection affinities were set to favour disconnections higher up the tree, and to values for the leaf level such that leaf disconnection probabilities tend to zero. In practice this resulted in all leaves being connected to parent nodes (which is desirable as we believe that single-pixel objects are unlikely). The β values increase with tree depth so that lower level nodes choose parents from a tighter neighbourhood. The P_l and θ_l values were unchanged, and again we consider binary valued nodes. The full set of parameters are summarised in Table 3.1.

A suite of 600 images were created by sampling DTs from the above prior and then generating 5 images from each. Figure 3.5(f) shows an example of an image generated by the DT and it can be seen that the "blockiness" exhibited by balanced TSBNs is not present.

A search for the MAP *dynamic tree* of each of these images was undertaken using simulated annealing with the same exponential strategy described earlier, and their log-posteriors are compared with those of the balanced TSBN in the Plot 3.7. The line denotes the boundary of equal log-posterior and the location of all the points above this clearly shows that in every case the MAP tree found has a higher posterior probability. This means that the MAP trees found all have a higher posterior probability than the balanced quad-tree. A plot of the MAP structure for the example image discussed earlier is given in Figure 3.6. Note how tree structures have formed to cover the black pixels and a single larger tree models all of the white background absorbing the isolated

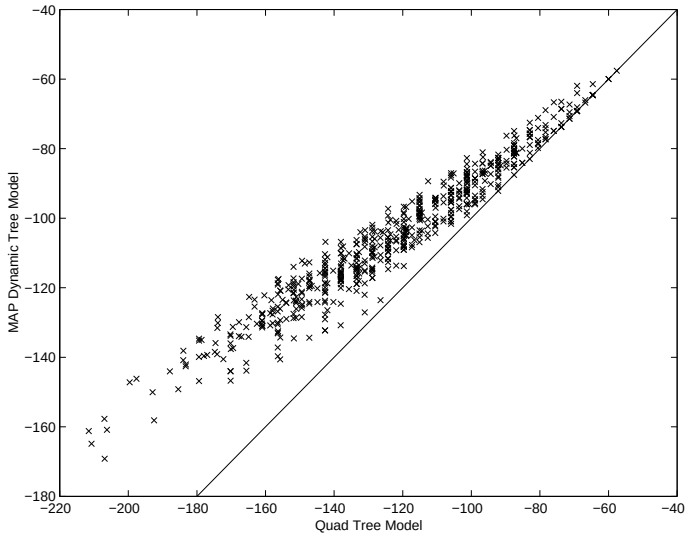


Figure 3.7: Comparison of the MAP DT log-posterior against the quad-tree for 600 images.

black pixels as noise. Ideally these small "black" trees should have joined together at level 2, but the four nodes present on this layer are clearly fully occupied with the white background. This suggests having more nodes at the higher levels would be advantageous. Work investigating this will be described in Section 3.4.

3.3 What Distributions can Dynamic Trees Model?

The preceding Sections have focused primarily on constructing the *dynamic tree* model and exploring its generative and inferential capabilities. They have given good insight into the operation of the model, and a qualitative assessment of its generative capabilities has been made, but it is instructive at this juncture to ask

> What distributions can the *dynamic tree* model? Are there any that it favours? Are they of interest to us? And does it offer any advantages over existing (simpler) models?

This Section is an attempt to address these questions. Firstly the procedure for making a comparison between the *dynamic tree* and the fixed tree structured belief network

is outlined in Section 3.3.1. Then in Section 3.3.2 the results of the comparison are shown, and in Section 3.3.3 the implications of the findings are considered.

3.3.1 Making a Comparison

In order to ascertain a measure of the usefulness of the *dynamic tree* compared to other models the natural candidate for comparison would be the fixed balanced tree-structured belief network. This is an interesting question to ask as the *dynamic tree* was motivated by certain limitations in the fixed structured belief network, most particularly its "blockiness," and furthermore the *dynamic tree* model is itself a generalisation of the fixed TSBN over structure.

We are interested in discovering whether there are any probability distributions over data which the *dynamic tree* can model more effectively than a fixed tree-structured network, and the best chance of finding such a distribution would be to use the *dynamic tree* itself as the generative model. To do this we make a comparison between $\langle \log P(\mathbf{X}_v) \rangle$ for the DT against $\langle \log P(\mathbf{X}_v) \rangle$ for the best quad-tree – where the average is over all of the patterns in our artificially generated dataset – and observe the gap between them.

The process requires averaging over all of the tree structures so we are limited to very small networks. For the comparison a 3-level, binary node configuration was chosen $(1-2-4)$, which gives rise to a total of 324 structures when allowing disconnections. Images handled by the model will thus be 1-d and 4 pixels in size. Pixel values were set to be binary, so a total of up to 16 images can be seen. The comparison is made between the *dynamic tree* and whichever of the 324 different fixed structures performs best on the given dataset.

Though very restricted in size, tractability prevents bigger models from being considered. Despite their size, the following Sections will demonstrate that some interesting insights into the *dynamic tree* can still be gained even with these simple models.

The comparison is straightforward. A parameter set is chosen and a generative *dy-*

namic tree model is created from them. This provides the weightings[8] $P(\mathbf{X}_v^n|\boldsymbol{\theta},\mathbf{a}) = \sum_{\mathbf{Z}^n} P(\mathbf{X}_v^n|\mathbf{Z}^n,\boldsymbol{\theta})P(\mathbf{Z}^n|\mathbf{a})$ for each of the $n = 1\ldots16$ data patterns effectively simulating infinite training data and representing the distribution exactly. For the *dynamic tree* the average entropy (negative log-likelihood) over these is given by

$$H(P(\mathbf{X}_v|\boldsymbol{\theta},\mathbf{a})) = -\sum_{n=1}^{16} P(\mathbf{X}_v^n|\boldsymbol{\theta},\mathbf{a}) \log P(\mathbf{X}_v^n|\boldsymbol{\theta},\mathbf{a}) \tag{3.6}$$

For the fixed structure model each of the 324 structures are considered in turn and the one whose average log-likelihood over the dataset is the highest was used in the comparison. To be completely fair to the fixed TSBN model its CPT and state prior parameters are optimised using the EM algorithm described in Section 5.3 for each of the 324 structures separately before the best is chosen.

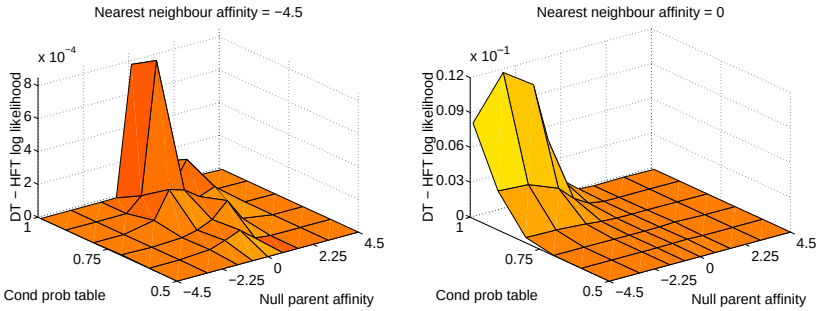
3.3.2 Investigating the Dynamic Tree Capabilities

The above framework provides the basis from which to investigate the capabilities of the *dynamic tree*. In order to examine as clearly as possible the role played by each type of parameter it is wise not to have too many of each, so they are kept to a minimum.

Consequently the CPT parameters were tied at all levels, as also were the priors. The natural parent affinity was fixed at 0 to provide a baseline and different values for the nearest-neighbour and null parent affinities were considered both above and below this. In a network of this size there are no other type of neighbours. β is fixed at 1.0 so the affinities correspond directly to the exponents used in the softmax function defining the connection probabilities (Equation (3.3)).

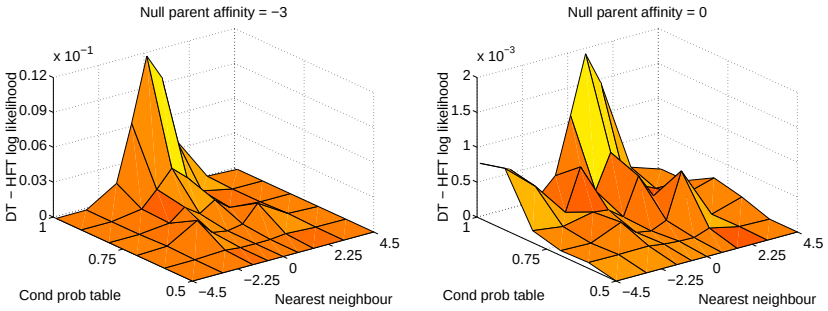
A systematic grid search was conducted over this domain to gain a general insight into the nature of the parameter space. An optimiser using the Nelder-Mead simplex method (Matlab function `fmins`) was then used at starting points near to the optima discovered in order to locate them more precisely. Visualizations of the results over key regions are plotted in Figures 3.8–3.10.

[8] $P(\mathbf{X}_v^n|\mathbf{Z}^n,\boldsymbol{\theta})$ is obtainable by Pearl message passing and the summations can be fully enumerated in the small tractable models.



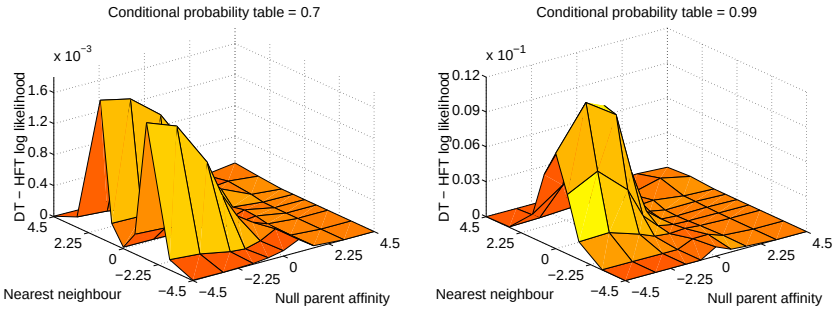
(a) (b)

Figure 3.8: Comparing the variation of the difference between DT and highest fixed tree (HFT) log-likelihood for null parent affinity, against the CPT at differing nearest neighbour affinity values of (a) -4.5, and (b) 0.



(a) (b)

Figure 3.9: Comparing the variation of the difference between DT and highest fixed tree (HFT) log-likelihood for the CPT against the nearest neighbour affinity at null parent affinity values of (a) -3, and (b) 0.



(a) (b)

Figure 3.10: Comparing the variation of the difference between DT and highest fixed tree (HFT) log-likelihood for null parent affinity, against the nearest neighbour affinity at differing CPT values of (a) 0.7, and (b) 0.9.

The affinity ranges are shown from -4.5 up to 4.5 which represent most of the full range of connection probabilities[9] between zero and one for that parameter. Ranges outside these parameters largely correspond to one link connection probability tending towards 1.0 essentially fixing the structure of the model to a single form and consequently the corresponding fixed structure does as well as the *dynamic tree*.

The CPTs are shown only for the range 0.5 to 0.99 as for binary units the lower half is a reflection about 0.5 corresponding to switching their class labels.

Figure 3.8 compares the effect of the null parent parameter against the CPT parameter on DT performance for nearest neighbour connection probabilities that are (a) very small, and (b) the same as the natural parent. There is one prominent peak on the plots and it can be seen that for low nearest neighbour connection probabilities it occurs when the null affinity is the same as the natural parent affinity. If the nearest neighbour affinity is increased to that of the natural parent (Figure 3.8(b)) this peak moves to null parent affinity of -3. Interestingly the height of the peak also goes up by an order of magnitude. In each case the CPT is hard which penalises transitions between states along the links – a useful feature that allows the *dynamic tree* to represent different regions as distinct "objects."

This suggests that the nearest neighbour-natural parent pair of affinities are more important to the *dynamic tree* than the null parent-natural parent combination. This is a little surprising as intuitively one might expect the latter to be more significant. However fixed structure trees with one or more of the higher nodes disconnected can often approximate the training data well by optimising their state prior at the roots, but it is harder when the choice is primarily between two parents in the tree. In the extreme case the fully disconnected tree has a lot of freedom, and was frequently picked as the best single fixed structure.

The nearest neighbour-CPT comparison plots of Figure 3.9 are of similar form and further illustrate the dominant importance of the nearest neighbour probability. In Figure 3.9(b) it is seen that if the null parent affinity is increased to that of the natural parent, then the best performing *dynamic trees* set their nearest neighbour affinities higher still.

[9] dependent on the values of the other affinity parameters.

The null parent against nearest neighbour affinity comparison plots of Figure 3.10 complete the picture. They demonstrate clearly that none of the *dynamic tree* parameters can be considered in isolation, and that they all have their part to play. Weakening the diagonal of the CPTs to 0.7 for example as in Figure 3.10(a) alters the optimal balance between the affinities requiring the nearest neighbour affinities to be different to the natural parent for highest likelihood gap between the *dynamic tree* and the best fixed structure. Nearest neighbour affinities higher than zero give the nearest neighbour a higher connection probability than the natural parent, and at the two peaks seen in the Figure the nearest neighbour and natural parent connection probabilities have nearly the same ratios, but are reversed (0.81 and 0.18).

At the extreme affinity ranges on the periphery of the plots and further out there is no difference between the average log-likelihood of the *dynamic tree* and the best fixed structure. This is due to one of the affinities dominating reducing the *dynamic tree* to a single form. This behaviour is summarised in the table. The undecided entry depends on the CPT value as to which structural form is dominant.

Nearest Neighbour	**Null Parent**	**Dominant Connection**	**Tree Structure**
Low	Low	Natural Parent	Balanced
Low	High	Null Parent	Fully Disconnected
High	Low	Nearest Neighbour	Nearest Neighbour
High	High	Nearest Neighbour/Null	Undecided

Table 3.2: Summary of *dynamic tree* behaviour at high affinities.

The plots shown in this Section and the discussion in the ensuing analysis all use a uniform state prior at the roots of $[0.5\ 0.5]$. The full range of priors from 0.5 to 0.99 were examined, but in practice the graphs were all seen to be of similar form. Thus the state prior parameter appears less important.

The grid search technique gave the maximum difference D of 0.0119 at nearest neighbour affinity of 0, null parent affinity -3, with a CPT diagonal of 0.99, and the Simplex search method corroborated this.

3.3.3 Analysis

This work was motivated by the desire to examine whether or not the *dynamic tree* model had any advantages over the simpler tree structured belief network which is prone to "blockiness" in its segmentations. The previous Section showed clearly that there were parameter ranges where the *dynamic tree* is superior to the best fixed structure that can be made from the same set of nodes. The highest gap in average log-likelihoods between the two was of the order of 10^{-2}, and although it is a little disappointing that it was not higher it should be noted these comparisons were made from structures containing only 7 nodes with there being a maximum of 16 different images that the model could produce.

For such a simple configuration it is not surprising that fixed structures perform well so the fact that there were DT parameter settings which allowed it to do better is very encouraging. It would have been nice to have considered larger models where the gap between *dynamic tree* and fixed model would be expected to widen considerably, but the addition of even a single further layer to the model increases the number of different configurations to in excess of 10^8 taking us into the realms of intractability.

Hard[10] CPT diagonals were seen to be preferable as this encouraged the formation of distinct trees for each region in an image and allows the concept of "objects" to be adopted. With such CPTs the plots showed that the nearest neighbour affinity plays a more significant role than the null parent affinity and that the optimal parameter settings to produce the widest gap in log-likelihoods occurred at nearest neighbour affinities identical to the natural parent and null affinities significantly smaller, typically around -3.

Intuitively this may seem surprising as one would expect disconnections to be more palatable under such a CPT scheme, but it appears that fixed structure trees with disconnections can optimise their root priors and produce good approximations. With the choice being out of two parents in the tree (nearest neighbour or natural) then no single fixed structure will be optimal and the *dynamic tree*'s configurability comes into its own.

[10] that is elements tending towards a probability of 1.0.

So in answer to the question posed at the beginning of this Section there are distributions that the *dynamic tree* favours, and it does offer advantages over the fixed tree structured belief network in many situations. As to the usefulness of the distributions in question it is hard to ascertain on such small models. However it shows there to be much potential for the *dynamic tree* and that the effort of developing the model such that it can be applied to and learn from real data would be interesting and worthwhile.

3.4 Enhancing the Dynamic Tree - Sparse Dynamic Trees

Experimentation with DT prior produced promising results as described in Sections 3.2.2.1 and 3.2.2.2. However limitations in the number of nodes due to starting with the balanced TSBN node arrangement created bottlenecks at higher levels of the network where more than one tree in the DT "forest" competed for the same node and led to some slightly awkward solutions. An example of such a case is shown in Figure 3.11 where two whites pixels are cut off from the rest of the white background.

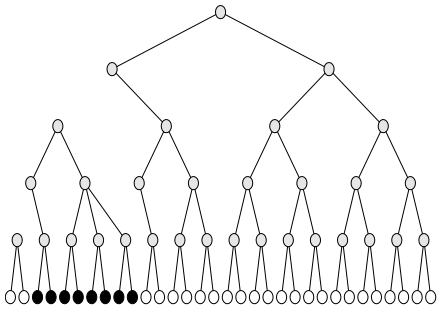


Figure 3.11: An example of too few nodes in the *full-time-node-employment* prior.

It could be argued that in such instances we may wish the two white pixels to remain isolated, but suppose we know that white is the background then it might be desirable to allow the *dynamic tree* to be able to produce interpretations where the background is fully joined. In the 1-d translation experiments of the 5 pixel stimulus, position 2 in Figure 3.4 is one instance where this could have been useful. The MAP example tree discussed in the 2-d experiments in Section 3.2.2.2 presents an even more compelling case for extra nodes at higher levels.

One solution to this dilemma may be to avoid modelling the background and concentrate on constructing trees only for the foreground objects. This may not be suitable generally as some images may not have any region which could be satisfactorily designated as background, however in a great deal of tasks such as digit or handwriting recognition for example, this method could be very desirable. This is another open research issue. Here we choose instead to relax the number of nodes at higher levels and allow the model to decide which nodes to use, and which nodes to leave unused and inactive. This gives rise to the *sparse dynamic tree*. This will now be discussed.

3.4.1 SDTs: The *Part-Time-Node-Employment* Prior

Increasing the number of nodes in each layer and requiring many of them to remain inactive means that we cannot continue with the prior described above where each unit chooses its parent independently, as this would tend to lead to chains or isolated links rather than tree structures. We have solved this problem by adopting a top-down approach to tree generation, with the probability of links occurring at a given layer being conditional on the activations of nodes in the layer above; we can thus grow forests of trees from the roots to the leaves. We note that Montanvert *et al.* (1991), and references therein describe a number of approaches to tree generation using both top-down and bottom-up approaches.

We start with an arrangement of nodes such as that in Figure 3.12(a). Initially all the nodes except those at the leaf (image) level are inactive, denoted by a broken line in the figure. A prior for activation is set for the nodes of the top layer and this is used to probabilistically activate some of the nodes and turn them on, as in Figure 3.12(b). Typically we wish only a single node to be activated at this level to form a master root for the image, so the prior probability needs to be set so as to encourage this.

After activating some (or none) of the top level nodes we consider the second layer. As before each child in turn probabilistically chooses a parent out of its set of potential parents, and if the parent chosen is active it connects to it and is turned on (Figure 3.12(c)). Choosing an inactive parent results in the node remaining inactive

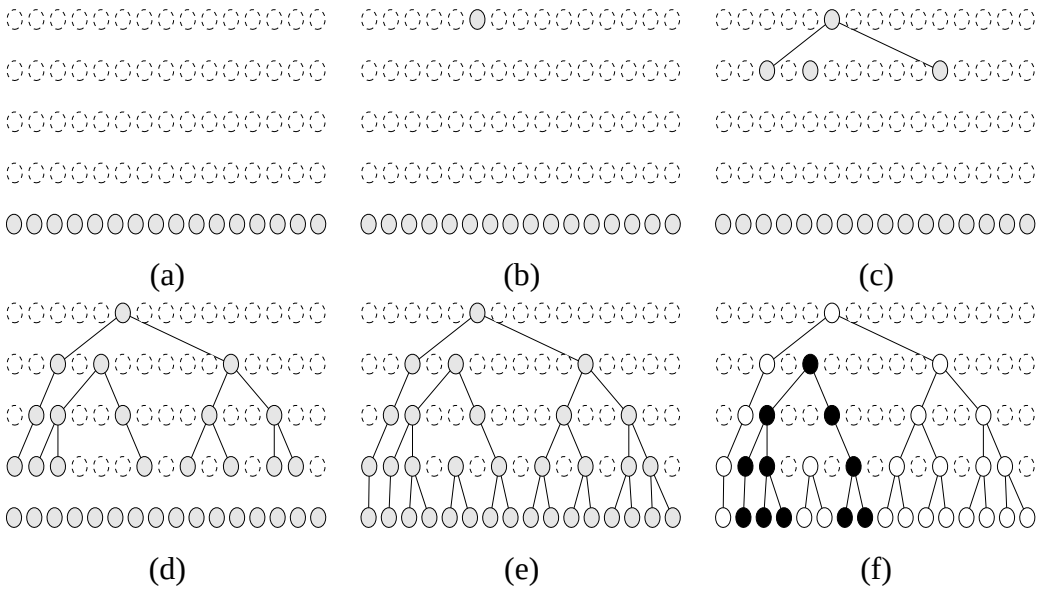


Figure 3.12: (a) Inactive nodes, (b) prior turns on a top level node, (c) connections are made to next level, (d) process is repeated until penultimate layer (e) bottom-up approach ensures all leaves get connected and (f) data generated from the tree in (e).

and disconnected. The "null" parent in each level is considered to be always active so that connecting to it results in the child becoming an active root.

For the sparse node arrangement we no longer have the notion of a "natural" parent as used by the *full-time-node-employment* prior, so the affinity profiles must be designed to encourage formation of tree structures. To get coherent trees such as in Figure 3.13, nodes at higher levels must connect to parents further away than those in the lower layers, so each layer is treated independently. We adopt the notation of affinity profiles and replace the individual affinities a_k that a node has for a particular parent with templates encompassing all nodes in the parent layer. These templates are set so that a node favours parents at or near the desired distance indicated by the peaks in the profiles of Figure 3.13. We use a reference affinity of 1 for the most favoured node(s), decreasing by 1 for nodes further from these points and affinities of $-\infty$ for nodes we do not wish to connect to. Within this framework we similarly set the affinities for the "null" parents for each of the lower layers. Thus the affinity profile for each child at higher levels in the tree will typically be bimodal, indicating that we expect some spreading out of the tree to occur, as shown in Figure 3.12(c).

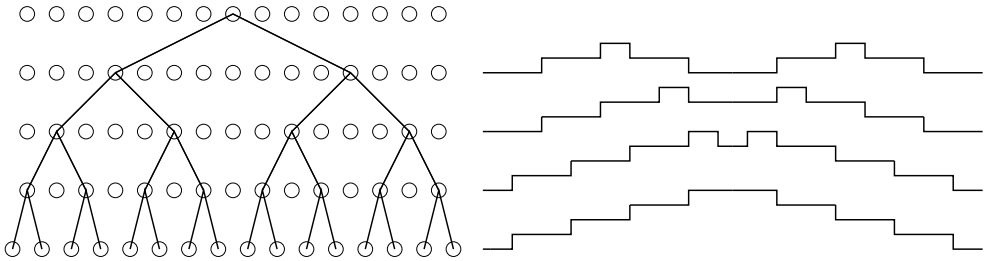


Figure 3.13: Affinity profile to encourage formation of tree structures.

To produce the connection probabilities the appropriate template for the particular layer is centred on the closest node above each child. The affinity values are then assigned to all the parent nodes from the template accordingly, and Equation (3.3) can then be used to produce the probabilities in the same way as for the standard *dynamic tree*.

The top-down procedure is repeated on successive layers down to the penultimate level (see Figure 3.12(d)). In this top-down approach the probability of node arrangement at level i is conditional only on the state of the nodes at level $i-1$. Denoting by $\mathbf{Z}_i$ the connection matrix of the ith layer and letting the leaf layer have index L, then we have the Markovian scheme $P(\mathbf{Z}_0, \mathbf{Z}_1, \ldots, \mathbf{Z}_{L-1}) = P(\mathbf{Z}_0) \prod_{i=1}^{L-1} P(\mathbf{Z}_i | \mathbf{Z}_{i-1})$.

The top-down method presents a problem at the leaf level of the network as it does not guarantee that all the leaves will be active. Clearly we cannot allow inactive leaves as they correspond to pixels in the image, so we modify the top-down strategy with a bottom-up step. To do this we simply allow each leaf node to choose its parent in the penultimate layer independently, and if any parents were not previously activated, they now become new roots. The fifth node from the left on the penultimate level of the tree in Figure 3.12(c) is an example of such a new root. Using this combined top-down and bottom-up scheme we can specify fully a prior over trees.

We have termed this the "*part-time-node-employment* prior" as the nodes can decide whether or not they wish to be employed in the tree structure or remain redundant and inactive.

When calculating $P(\mathbf{Z})$, we need to take into account the fact that if a node in layers

2 to $L-1$ is inactive, it could have become so by connecting to *any* inactive parent in the layer above. Hence the probability of a node becoming inactive is computed by summing the probabilities of connecting to all of the off parents.

Given that we have specified a prior over network architectures and generated a "forest" structure Z we must now translate this into a TSBN. The nodes in a TSBN take on discrete values and can be considered as C-class multinomial random variables. On each of the links there is a conditional probability table (CPT) θ_k, which defines the probability of changing state during a transition between nodes, and for the root node there is a prior P. DTs and SDTs use this same methodology, however we need to associate a prior P with each unit as all have the potential to become a root. Currently these parameters are defined on a level by level basis. Given a particular $\mathbf{Z}$ matrix the probability of a particular instantiation of all of the random variables is simply the product of the probabilities of all of the trees, where the appropriate root probabilities and CPTs are picked up from the P_ls and θ_ls. Figure 3.12(e) shows a sample generated from our tree structure.

3.4.2 Inference in SDTs

With the more complex *part-time-node-employment* prior of Section 3.4.1 the links of the SDT are no longer independent of each other and thus some architectural changes in a given tree will not be legal.

Sampson (1996) used simulated annealing to search over parse trees for natural language sentences in an attempt to learn their grammar. In order to achieve this he defined neighbourhood relationships which indicated the points in the space that could be reached from any given point by one of the discrete set of permissible moves.

By similarly constructing relationships between particular configurations we can use sampling to explore the posterior distribution of this prior and simulated annealing to find MAP structures just as was obtained for the *dynamic tree* model. To do this we define a mechanism for proposing changes to the network structure which the annealer can decide to reject or accept in the normal way. As the connections of the network

are conditioned on the states of the nodes in the adjacent layer and not independent as in the case of the *full-time-node-employment* prior, we must also consider the effects on neighbouring units as part of any proposed change to a node i. Nodes can now be active or inactive so we also need to provide a means to allow dormant nodes to be activated and redundant ones to be put to sleep. We use the move set summarised below to explore $\mathbf{Z}$ space:

Randomly choose a node $i \in \{\text{all nodes}\}$
If Node is currently *on*
  If Node is **not** *a leaf*
    De-activate Node i;
    Break link with parent;
    Probabilistically choose new parent for all daughters connected to i;
  Else
    Choose another parent for the leaf;

Else
  Activate Node i;
  Probabilistically choose a parent for i;
  Ask all active daughters if they prefer connecting to i. Move link if *yes*;

The move set described above is clearly only one of a number of possible strategies, for example it might be argued that toggling the activation state of non-leaf nodes may cause too big a change in the tree structure and allowing an intermediate move where an active node simply picks different parents could prove beneficial. Another interesting alternative includes moving or combining nodes within a layer while preserving their links (as far as possible). This would allow better placed nodes to be found and employed while retaining the network connectivity and enable gentler changes to the network than with the current strategy. It would still need to be combined with mechanisms to enable nodes to try different parents and for new nodes to be switched in or redundant ones de-activated as in the above.

Consideration of the move strategy is an important issue which directly influences the

coverage of the search space made during annealing and thus the quality of solutions found. The presence of units which may be active or inactive also adds further complications in MCMC (see Godsill (1997)). This is an interesting area for further research.

3.4.3 Existing Sparse Models

The notion of *sparseness* in SDTs connects to the second source for this work, sparse coding. There has been considerable interest in the idea of sparse coding, e.g. Olshausen and Field (1996). In this work an image is generated as a linear combination of basis functions $\mathbf{I} = \sum_i c_i \boldsymbol{\phi}_i$, where the c_i's are the coefficients on the basis functions $\{\boldsymbol{\phi}_i\}$. There is a prior on the coefficients which encourages relatively few of them to be on, but this prior is independent for each coefficient. The SDT extends this by using a layered prior which means that correlations are induced in units in a layer from patterns of activation in the layer above.

The general sparse framework can be fully described using three sets of parameters, existence variables for each unit S, variables specifying the connectivity between units Z, and state variables X. The Credibility networks of Hinton *et al.* (2000) focus primarily on S and Z, where-as the SDT considers them all. Note in contrast that the *dynamic tree* model which constitutes the core emphasis of this thesis considers only Z and X.

3.4.4 Comparing SDTs to other Image Models

The SDT was motivated by a need to allow the DT model to have extra nodes at higher levels. We repeat the experiments of Section 3.2.2.1 on the SDT to compare the performance of the *sparse dynamic trees* with that of the *full-time-employment* prior of the DT, and the balanced TSBN.

We consider the same stimulus of 4 and 5 pixel bars in a 1d image of 16 pixels as used for the 1-d *full-time-employment* prior (DT model) experiments. The corresponding balanced TSBN for such an image is thus a 5-layer binary tree and we choose the

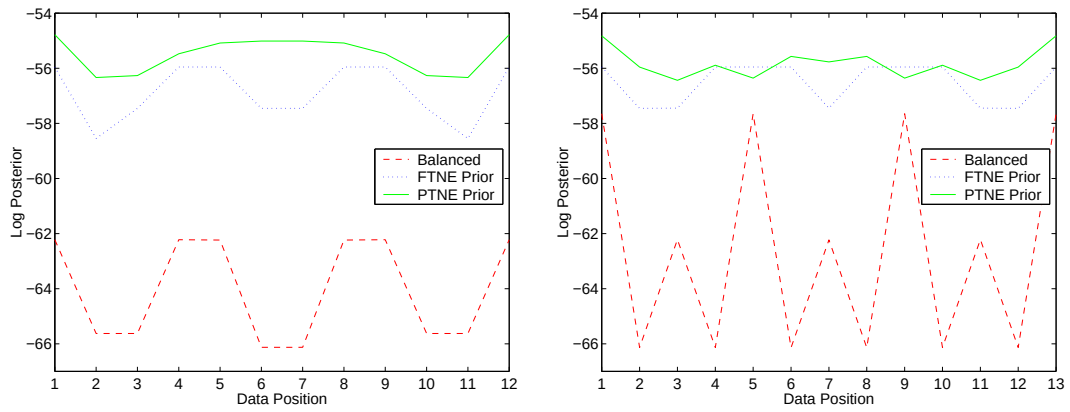
node organisation of the *part-time-employment* prior such that the balanced TSBN fits into this arrangement. This still gives a wide scope and we select the arrangement of Figure 3.12(a) with a large uniform number of 15 nodes in each layer above the leaves, so that the number of nodes does not limit the type of structures we can create.

Each node in the tree is a binary variable taking on values of either white or black. We set the θ_l's to be equal for all layers l, with values of 0.99 on the diagonal and 0.01 off diagonal, and the P_ls $= (0.75, 0.25)$ on all layers. In the P_l vectors the first element is the prior of the node being white so we are favouring white roots by a factor of 3 to 1. This is determined by the statistics of the images under consideration, which in this instance have approximately three times as many white pixels as black.

The affinities a_k and βs are relevant to all the layers except the very top as any active node at this level only has the option of connecting to a null parent. For the sparse node arrangement we no longer have the notion of a "natural" parent as used by the *full-time-node-employment* prior, so the affinity profiles must be designed to encourage formation of tree structures. To get coherent trees such as in Figure 3.13, nodes at higher levels must connect to parents further away than those in the lower layers, so each layer is treated independently. We adopt the notation of affinity profiles and replace the individual affinities a_k that a node has for a particular parent with templates encompassing all nodes in the parent layer. These templates are set so that a node favours parents at or near the desired distance indicated by the peaks in the profiles of Figure 3.13. We use a reference affinity of 1 for the most favoured node(s), decreasing by 1 for pairs of nodes further from these points, eg. the 4th level affinity profile is of the form $(\ldots, -1, -1, 0, 0, 1, 0, 1, 0, 0, -1, -1, \ldots)$.

Within this framework we similarly set the affinities for the "null" parents for each of the lower layers. We choose, $a_{null} = 0.0$ for all levels, and set the β's to 1.5. The prior for activating the nodes on the top level was set to $1/7$.

We illustrate the effects of translation by applying the black bar stimuli at successive positions in the image and find the unnormalised log-posterior $\log P(\mathbf{Z}|\mathbf{X}_v) \propto \log P(\mathbf{Z}) + \log P(\mathbf{X}_v|\mathbf{Z})$ for the particular $\mathbf{Z}$ configurations. Our interest is in the maximum a posteriori (MAP) $\mathbf{Z}$ for each image and we use simulated annealing combined



(a) 5 black nodes (b) 4 black nodes

Figure 3.14: Plots of the unnormalised log-posterior vs position of the input pattern for (a) the 5 black-nodes pattern and (b) 4 black-nodes pattern.

with our move set to search for these. We also calculate the posterior of the balanced TSBN architecture and compare it with the MAP configurations in Figures 3.14(a) and (b). The affinity templates and other model parameters have been set such that the highest prior probability $P(\mathbf{Z})$ will be that of the balanced TSBN arrangement.

In the Figures the x-axis denotes the position of the left hand end of the bar, and the y-axis shows the log-posterior probability.

We see from these Figures that the plot of the posterior of the *part-time-node-employment* prior is almost flat indicating very little translation variation compared to the heavily peaked and troughed posterior of the balanced TSBN. Notice also from the Figures that the *part-time-node-employment* prior achieves a flatter posterior than the *full-time-node-employment* prior. This is explained by extra nodes being available, enabling similar structures to be formed at each of the different bar positions, as opposed to the shortage of nodes at some positions encountered in the posterior of the *full-time-node-employment* prior.

Some exploration of the parameter space has yielded even flatter posterior plots for the *sparse dynamic tree* than the ones shown in Figure 3.14, however setting the parameters is not trivial. It should be possible to learn the model parameters using an approach

such as EM in much the same way that the CPTs and priors are learned for fixed architecture networks. This question shall be addressed for the *dynamic tree* model in Chapter 5.

3.5 Discussion

The above experiments demonstrate that the DT model overcomes the "blocky" segmentation problem of TSBNs and that it has a greater translation invariance than balanced TSBNs. Its dynamic architecture enables the creation of structures which we have shown better explain the binary image data under consideration. Other authors such as Sallans *et al.* (1998) have considered dynamic networks in which the structure can switch, but they allow general DAGs rather than just trees. So too Hinton *et al.* (2000) also use general DAGs for their credibility networks. We believe that it is useful to find out what can be achieved using TSBNs.

We also note the similarity between posterior tree configurations and parses generated by a context-free grammar (CFG). CFGs have a $O(n^3)$ algorithm to determine the MAP parse; however, this algorithm depends crucially on the one-dimensional ordering of the inputs. We believe that the possibility of crossed links in the (S)DT architecture means that this kind of algorithm is not applicable to the DT case. Also, *dynamic tree* models can be applied to 2-d images, where the $O(n^3)$ algorithm is not applicable.

Generating from the *dynamic tree* produced 2-d images which did not exhibit the same blockiness inherent in fixed architecture TSBNs. Since the generative capabilities of a model are usually a good indicator of its inferential ability this suggests the DT offers a better representation of real-world images than fixed architecture TSBNs. The more rigorous comparison of Section 3.3 clearly shows that even in a small set of toy images there are many instances where even choosing an optimal fixed architecture from the whole set of possible fixed architecture configurations is inferior to the full *dynamic tree* model.

We have also seen that simulated annealing methods are successful at finding trees that have high posterior probability. Though annealing is an effective means of finding good solutions from a distribution we cannot fully enumerate, simulated annealing (as with sampling techniques) is computationally costly and probably too slow for use in a practical system.

While it is true that inference over DTs is believed to be NP-hard, we do retain a "clean" semantics based on the fact that we expect that each pixel should belong to one object, which may lead to useful approximation schemes as an alternative to sampling from the true posterior ($P(\mathbf{Z}, \mathbf{X}_h|\mathbf{X}_v)$) distribution. One possibility is to use a mean-field-type approximation to the posterior of the form $Q_Z(\mathbf{Z})Q_h(\mathbf{X}_h)$ (Zoubin Ghahramani, personal communication, 1998). This is considered in the next Chapter.

Up until now we have conducted all the experiments solely on toy examples. For the DT model to be interesting it must be scalable to real images, and this is probably the primary issue for consideration. In tandem with this though is also a requirement to investigate learning of the model's parameters as setting their values will no longer be trivial for large structures. Chapters 6 and 5 respectively will address these concerns.

Finally we note that the SDT is an important first step towards further enhancing the *dynamic tree* model. It addresses the fact that there are occasions where higher up in the *dynamic tree* there are sometimes too few nodes to build the structures that the model really wants to by adding extra nodes which are switchable. It maintained all the advantages of the DT but the results suggest that there is clearly much more work which could be done. There are also many other interesting research directions including the introduction of additional information at the nodes; for example one might use real-valued variables in addition to the multinomial variables considered above. These additional variables might be used to encode information such as that concerning the instantiation parameters of objects.

Chapter 4

Mean Field Dynamic Tree

4.1 Introduction

In the previous Chapter the *dynamic tree* model was introduced. It was seen to exhibit improved performance over the fixed architecture TSBN in terms of blockiness, but at a cost of tractable inference.

There are two key approaches which can be adopted in such circumstances to approximate the posterior distribution of the *dynamic tree*. The first which involves sampling or using search over the true posterior distribution has already been considered, and in Section 3.2.2.2 simulated annealing was adopted. It was however seen to tend to be slow to converge. The alternative is to use variational methods. Here a tractable approximating distribution is fitted to the true posterior, and a standard technique involves the use of a factorised distribution (the mean field approach) (Saul and Jordan, 1996; Ghahramani, 1997). This Chapter[1] will demonstrate that such an approximation is useful for *dynamic trees*.

Comparisons are made between this mean field approach and a maximum a posteriori

[1]The work of this Chapter was completed in collaboration with Amos Storkey and constitutes the subject of the paper Adams, Storkey, Ghahramani and Williams (2000). The full derivation of the mean field algorithm applied to the *dynamic tree* is given in Appendix B, and the author wishes to thank Zoubin Ghahramani for producing the original analysis.

approach using simulated annealing. We will explore the relative merits of the two approaches and assess both analytically and qualitatively the solutions that each find in order to assess whether mean field can provide a practical alternative to sampling.

Section 4.2 of this Chapter describes the theory behind the mean field approach to DTs, and experiments comparing it with other methods are described in Section 4.3.

4.2 Mean Field for Dynamic Trees

We now introduce the mean field theory for *dynamic trees*. This Section gives only the basic outline. For a full derivation consult the Appendix B.

The *dynamic tree* can be considered as an ordered set U of nodes $i = 1, 2, \ldots, U$, each of which taking on one of C possible states, $1, 2, \ldots, c$. If $Z = \{z_{ij}\}$ is used to denote the set of possible directed tree structures over these nodes, where z_{ij} is an indicator variable, then $z_{ij} = 1$ indicates that the node i is connected to parent j. By ordering the nodes such that upper level nodes have lower indices than those in the layer(s) below then it means that $z_{ij} \equiv 0$ for $j \geq i$. Finally defining $X = \{x_i^k\}$ to be the set of all of the states of the nodes, then analogously with the z indicator variables, $x_i^k = 1$ if node i is in state k, and is zero otherwise.

Recall that there are two components to the *dynamic tree*, a prior over tree structures $P(\mathbf{Z})$, and the likelihood $P(\mathbf{X}|\mathbf{Z})$ of being in a particular state conditioned on the structure $\mathbf{Z}$. Conditional probability tables $\boldsymbol{\theta}$ define the state transition probabilities across connected links and act as a conditional prior over node states. For the prior we use the fully factorised *full-time-node-employment* prior described in Section 3.2.1.3, and it is given again in the Equation below.

$$P(\mathbf{Z}|\boldsymbol{\phi}) = \prod_{i=1,\, j=0}^{U} \pi_{ij}^{z_{ij}} \tag{4.1}$$

The index $j = 0$ is used to denote the special case of a connection to the *null* parent which produces a disconnection, and π_{ij} is the probability that node i chooses the

parent j. $\boldsymbol{\phi}$ is used to provide consistent treatment for the representation of the set of parameters which govern the prior $P(\mathbf{Z})$. In this Chapter they are realised as explicit probabilities π_{ij}, but subsequent Chapters will introduce alternative representations.

Given these definitions the joint prior distribution can be written as follows

$$P(\mathbf{Z},\mathbf{X}|\boldsymbol{\theta},\boldsymbol{\phi}) = \prod_{i=1}^{U}\prod_{j=0}^{U}\pi_{ij}^{z_{ij}}\prod_{k,l=1}^{C}[\theta_{ij}^{kl}]^{x_i^k x_j^l z_{ij}} \tag{4.2}$$

where the indicator variables z, x pick out the correct probabilities.

The nodes of the *dynamic tree* constitute two distinct sets. The first contains evidential or visible units $\mathbf{X}_v$ which are instantiated with the image data. The second are the hidden units $\mathbf{X}_h$ whose value has to be inferred. Conditioning on the training data (visible units) the posterior distribution of the *dynamic tree* takes the form

$$P(\mathbf{Z},\mathbf{X}_h|\mathbf{X}_v) = \frac{P(\mathbf{Z},\mathbf{X})}{P(\mathbf{X}_v)} \tag{4.3}$$

In the mean field variational approach this posterior distribution is approximated by a factorising distribution of the form $Q(\mathbf{Z}|\mathbf{X}_v)Q(\mathbf{X}_h|\mathbf{X}_v)$ where $Q(\mathbf{Z}|\mathbf{X}_v)$ approximates over the $\mathbf{Z}$ distribution and $Q(\mathbf{X}_h|\mathbf{X}_v)$ the hidden units $\mathbf{X}_h$. Appendix B gives the full derivation for the interested reader. Here the basic approach is described. For notational simplicity the conditioning on the image data $\mathbf{X}_v$ in the variational approximation is dropped and shall be assumed.

The Kullback-Liebler (KL) divergence provides a convenient measure of the divergence between two probability distributions. Choosing good forms for the Q distribution is achieved by minimising the KL divergence between the approximating $Q(\mathbf{Z})Q(\mathbf{X}_h)$ and the true posterior distribution. The KL divergence for this is given by

$$\begin{aligned} KL(Q||P) &= \sum_{\mathbf{Z},\mathbf{X}_h} Q(\mathbf{Z})Q(\mathbf{X}_h)\log\left(\frac{Q(\mathbf{Z})Q(\mathbf{X}_h)}{P(\mathbf{Z},\mathbf{X}_h|\mathbf{X}_v)}\right) \\ = \log P(\mathbf{X}_v) &- \sum_{\mathbf{Z},\mathbf{X}_h} Q(\mathbf{Z})Q(\mathbf{X}_h)\left[\log P(\mathbf{Z},\mathbf{X}) - \log Q(\mathbf{Z}) - \log Q(\mathbf{X}_h)\right] \end{aligned} \tag{4.4}$$

By assuming a factorising form for the Qs mean field makes the structure variables $\mathbf{Z}$ be independent of the node states $\mathbf{X}$ which allows us to treat them separately. In the true posterior distribution they are of course not independent – so an exact fit is unlikely – however this makes the computation tractable. The quality of the approximations found are assessed against MAP samples from the true posterior in Section 4.3. We start by optimising with respect to the $Q(\mathbf{Z})$ distribution.

4.2.1 Calculating Q(Z)

To optimise the KL divergence for the $\mathbf{Z}$s, $Q(\mathbf{X}_h)$ is fixed and a minimisation with respect to $Q(\mathbf{Z})$, subject to the constraint $\sum_j Q(z_{ij}) = 1, \forall i$ is performed. The results of the analysis produces the following expression for $Q(\mathbf{Z})$

$$Q(\mathbf{Z}) = \prod_{ij} \frac{\exp(z_{ij}\lambda_{ij})}{\sum_s \exp(\lambda_{is})} \tag{4.5}$$

where

$$\lambda_{ij} = \log \pi_{ij} + \sum_{kl} \langle x_i^k x_j^l \rangle_{Q(X)} \log \theta_{ij}^{kl} \tag{4.6}$$

So with $Q(\mathbf{X}_h)$ fixed we can explicitly calculate the optimal $Q(Z)$, and interestingly this results in a fully factorised form for $Q(\mathbf{Z})$.

4.2.2 Calculating Q(X)

To complete the optimisation we now need to minimise the KL divergence for $Q(\mathbf{X}_h)$ keeping $Q(\mathbf{Z})$ fixed. Substituting for $P(\mathbf{Z},\mathbf{X})$ from Equation (4.2) into the expression for the KL divergence (Equation (4.4)) gives

$$
\begin{aligned}
KL(Q||P) &= \log P(\mathbf{X}_v) + \sum_{\mathbf{Z}} Q(\mathbf{Z}) \log Q(\mathbf{Z}) + \sum_{\mathbf{X}_h} Q(\mathbf{X}_h) \log Q(\mathbf{X}_h) \\
&\quad - \sum_{\mathbf{Z},\mathbf{X}_h} Q(\mathbf{Z}) Q(\mathbf{X}_h) \log \left[\prod_{i=1}^{u} \prod_{j=0}^{u} \pi_{ij}^{z_{ij}} \prod_{kl} [\theta_{ij}^{kl}]^{x_i^k x_j^l z_{ij}} \right] \\
&= -\sum_{ij} \mu_{ij} \left[\log \pi_{ij} + \sum_{kl} \langle x_i^k x_j^l \rangle_{Q(\mathbf{X}_h)} \log \theta_{ij}^{kl} \right] + \sum_{\mathbf{X}_h} Q(\mathbf{X}_h) \log Q(\mathbf{X}_h) + f(\mathbf{Z}) \qquad (4.7)
\end{aligned}
$$

where $\mu_{ij} = \langle z_{ij} \rangle_{Q(\mathbf{Z})}$, and $f(\mathbf{Z})$ contains only $Q(\mathbf{Z})$ terms which are constant in the minimisation with respect to $Q(\mathbf{X}_h)$. Performing the minimisation and solving for $\mathbf{X}_h$ gives a solution of the form

$$
Q(\mathbf{X}_h) = \frac{1}{K_1} \exp\left(\sum_{ij} \mu_{ij} \sum_{kl} x_i^k x_j^l \log \theta_{ij}^{kl}\right) \qquad (4.8)
$$

where K_1 is the partition function, and this is essentially a layered Boltzmann Machine (as μ_{ij} is zero for all connections that are not between adjacent layers). Thus we see that the full *dynamic tree* structure can be broken down into a series of layer Equations. However these computations are still exponential in the number of nodes on a level (see Section 3.2.1.2) and therefore intractable for usefully sized models. To proceed any further it is necessary to make a simplifying assumption. One particularly convenient assumption is that of a fully factorised form for $Q(\mathbf{X}_h)$

$$
Q(\mathbf{X}_h) = \prod_i Q(x_i) = \prod_{i,k} (m_i^k)^{x_i^k} \qquad (4.9)
$$

In the above expression m_i^k is the probability that node i is in state k. It is the mean that node i will be in state k independent of the other nodes in the network. This results in the fully factorised distribution for $Q(\mathbf{Z}, \mathbf{X}_h)$ which is the characteristic hallmark of the mean field approximation and is what makes it the simplest of all the variational techniques.

This assumption allows us to make the following substitution

$$\langle x_i^k x_j^l \rangle_{Q(X)} = m_i^k m_j^l \tag{4.10}$$

Care must be taken to ensure that $\sum_k m_i^k = 1, \forall i$ and this is achieved simply by the addition of a Lagrange Multiplier term $\sum_\alpha \rho_\alpha \left(\sum_\beta m_\alpha^\beta - 1 \right)$. A straightforward application of calculus gives the following expression for the means

$$m_s^r = \frac{\exp(\gamma_s^r)}{\sum_{r'} \exp(\gamma_s^{r'})} \tag{4.11}$$

where

$$\gamma_s^r = \sum_{j<i} \sum_l \mu_{sj} m_j^l \log \theta_{sj}^{rl} + \sum_i \sum_k \mu_{is} m_i^k \log \theta_{is}^{kr} \tag{4.12}$$

Equations (4.12) form a set of coupled mean field equations which can be solved by an iterative update (Saul and Jordan (1996)).

This update is performed asynchronously on each of the nodes and repeated cyclically until convergence is reached.

4.2.3 Putting it all Together

Equations (4.5) and (4.11) provide the necessary results to perform an optimisation on the KL divergence. The complete procedure is as follows.

1. Initialise all μ_{ij} and m_i^k.
2. Cyclically update Equations (4.11) to find the local optimum for the means[2] m_i^k.
3. The $Q(\mathbf{Z})$s can then be calculated directly from Equation (4.5).
4. Repeat steps 2–3 until converged.

[2] Note that $\langle x_i^k x_j^l \rangle_{Q(X)}$ needs only to be computed for $j < i$ as $z_{ij} \equiv 0$ for $j \geq i$.

Note that each step of the process is guaranteed not to increase the KL divergence (4.4), and is bounded from below by 0, so convergence is assured.

Mean field therefore is an attractive approach as it replaces the intractable dependencies between $\mathbf{X}$ and $\mathbf{Z}$ in the true posterior with simpler computations which can be solved iteratively. This allows us to fully enumerate over tree structures $\mathbf{Z}$, but at the cost of now only having an approximation to the true posterior distribution. In the next Section a comparison is made between mean field and sampling from the true posterior. This will assess the quality of the mean field approximation of *dynamic trees*.

4.3 Experiments

We explore and contrast the performance of the mean field approach with that of simulated annealing (Section 3.2.1.5) using a 6 layer binary tree. With this architecture we have 1-d images with 32 pixels. Initially we shall consider the case where the node states are binary variables and the images are black and white.

A standard DT model of the above architecture was used. The prior over node states was set to be uniform, with conditional probabilities of 0.99 down the diagonal and 0.01 off-diagonal. The probability of nodes choosing to become a root (disconnecting) were set to be more favourable than connecting to the *nearest neighbour*, but less favourable than connecting to the *natural parent*. This was achieved in the same way as described in Chapter 3, using the prior $P(z_{ij}|\boldsymbol{\phi}) = \exp(\beta a_{ij}) / \sum_k \exp(\beta a_{ik})$. The affinities, a_{ij}, were set as 1 for the *natural* parent, and $1 - N$ for the *N'th nearest neighbours* of the *natural* parent, with $\beta = 1.25$. The affinity for becoming a root, a_{null}, was 0.5. The model was sampled to generate a suite of training data of some 600 images which were used in our experiments. With this prior the images averaged an equal number of black and white pixels of variable object size.

In the experiments we use simulated annealing as described in Section 3.2.1.5 to find the maximum a posteriori (MAP) configuration of the DT for each of the images. This was conducted from a starting temperature of 1.0 and exponentially decreased by a

factor of 0.9. At each temperature up to 2000 proposals could be made, although transition to the next temperature would occur after 200 accepted steps. The run was deemed to have converged after five successive temperature steps were made without accepting a single step.

For the mean field approach we order the nodes from the bottom nodes to those on the higher levels, and sweep through them updating the *m*s asynchronously a total of 20 times each. This was found to be sufficient to allow these simultaneous equations to reach their equilibrium state. The $Q(\mathbf{Z})$s can then be recalculated. Typically the algorithm converged[3] after 4 or 5 iterations. Mean field was found to be of the order 100 times faster than simulated annealing.

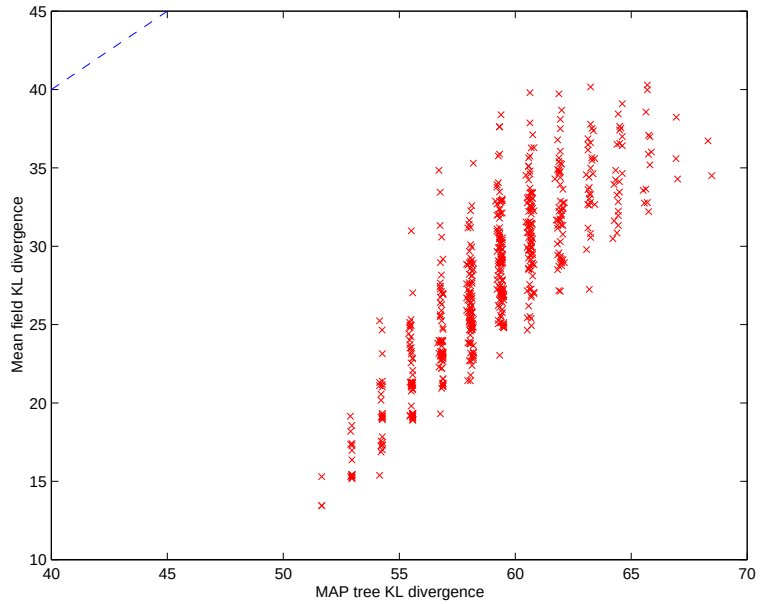
To compare mean field and simulated annealing we plot the KL divergences[4] $KL(Q||P)$ against $KL(R||P)$, where R is the MAP tree configuration (see Figure 4.1(a)). From the comparative plot of Figure 4.1(a) it is clear that the KL divergence of the mean field solutions is significantly lower than that of the MAP *dynamic tree* in all instances. Typically we see from Figure 4.3(a), a difference in KL divergence of about 30 between the mean field example and the corresponding MAP tree. These results can be understood when we realise that although the mean field approximation requires the assumption that $P(X)$ can be factorised, ie. $P(\mathbf{X}) = \prod_i P(x_i)$, it maintains a distribution over $P(\mathbf{Z})$. For the MAP case we usually choose a tree with greater posterior probability, but we are only basing our estimate of the KL divergence on a single structure, which is unlikely to account for a high proportion of the probability mass of the posterior distribution. It can be seen that the distribution of points is grouped into a series of energy bands for the MAP model, whereas for the mean field method they are more evenly spread. This is probably due to the discrete nature of choices over tree structure and node state in the true posterior distribution.

We can also compare the posterior[5] probability of the MAP tree found by annealing

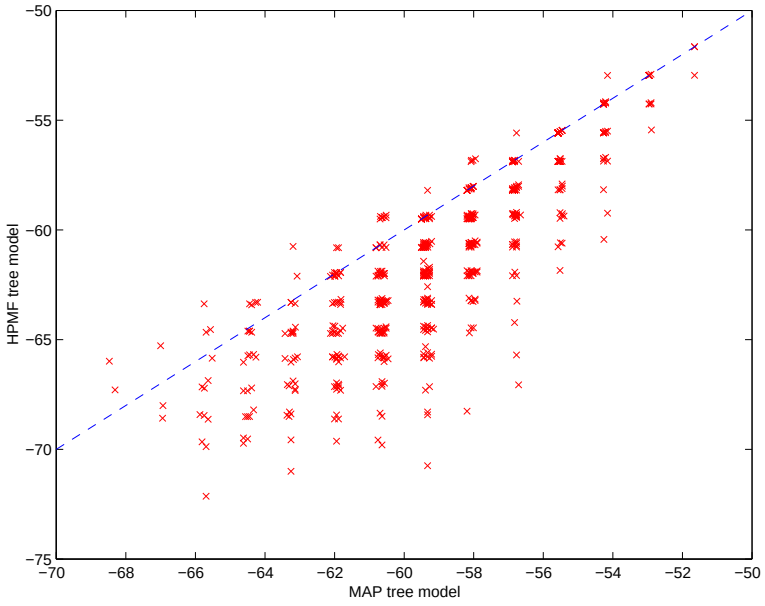
[3] A threshold change of less than 0.1 in the KL divergence between successive steps was found to be sufficient to allow the $Q(\mathbf{Z})$ to stabilise on a particular configuration.

[4] The KL divergence can be computed up to the addition of a constant dependent solely on the probability of the image data, $P(\mathbf{X}_v)$.

[5] We define the posterior as $P(\mathbf{Z}|\mathbf{X}_v) \propto P(\mathbf{Z})P(\mathbf{X}_v|\mathbf{Z})$ and ignore the normalising term $P(\mathbf{X}_v)$ which is constant across the two approaches.



(a)



(b)

Figure 4.1: Comparison of (a) KL divergence, and (b) the unnormalised log-posterior of the MAP tree against the corresponding mean field DTs.

and posterior of the highest probability tree structure found by mean field (HPMF tree), where the connected links in the HPMF tree are the z_{ij}s of highest probability from the

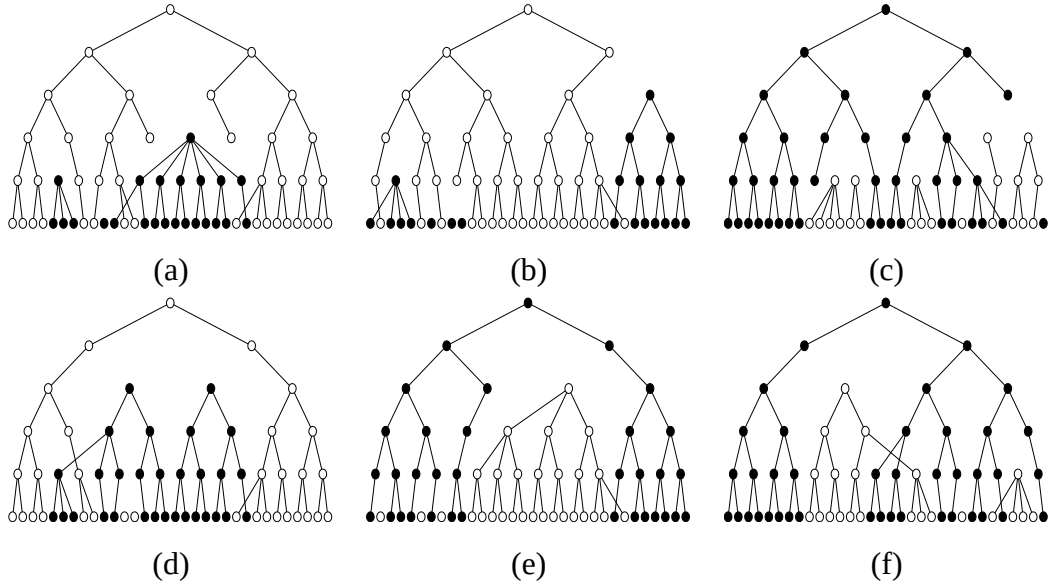
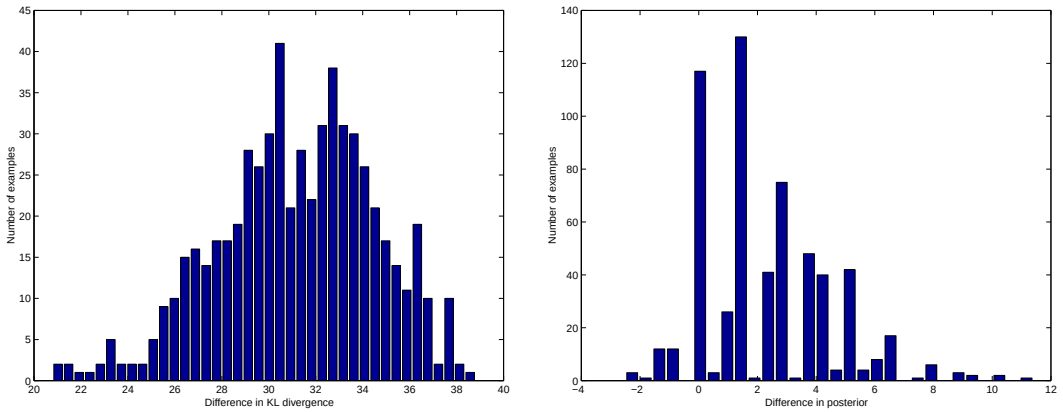


Figure 4.2: (a)-(c) The HPMF structures for 3 different images, and (d)-(f) the corresponding MAP trees found by annealing.

$Q(\mathbf{Z})$ distribution. Such a comparison is made in Figure 4.1(b), where the line in the figure denotes the boundary of equal log-posterior.

We notice that in most cases (81.8%) the annealed tree has a higher posterior, but in 11.5% of cases it is the same, and for 6.7% of examples the mean field approach actually found a higher posterior tree (see Figure 4.3(b)). The latter is probably an indication that though the annealer generally finds very good optima, it cannot guarantee finding the global solution. Mean field by attempting to fit a distribution better explores the landscape and is able find some of the harder solutions. However as we cannot exactly fit the posterior we should usually expect the sampling approach to give better results. These results are encouraging in that they demonstrate that the mean field algorithm is able to find interpretations of the data which are comparable in performance to the MAP structures found by sampling, at only a fraction of the computational cost.

A qualitative examination of the types of structures the mean field technique finds is quite instructive. Three examples of HPMF trees found for different images are shown in Figures 4.2(a)–(c), and the MAP trees found by simulated annealing with the same data are shown below them in Figures 4.2(d)–(f). It can be seen that there is a high



(a) (b)

Figure 4.3: Histograms of the difference (MAP – HPMF) in, (a) KL divergence, and (b) posterior probability, between the MAP and corresponding HPMF DTs.

degree of similarity in their structure, with both methods picking out objects in the image as separate trees. We see that noisy pixels are largely ignored in both approaches and left isolated. This is a facet of the strongly diagonal CPTs and further experiments with weaker CPTs of 0.9 down the diagonal still produced interesting structures, but allowed the noisy pixels to be absorbed by the object that contained them.

Plotting the difference in KL divergence and the difference in posterior probability of the MAP and HPMF approaches (as in Figures 4.3(a) and 4.3(b)) proves informative. We see from Figure 4.3(a) of the difference in KL divergence that this difference can be approximated by a unimodal bell-shaped curve. For the difference between posteriors, Figure 4.3(b), we notice that it has no distinct form, but most of the mass is close to or at zero and this indicating that the mean field method usually finds HPMF solutions that are very close to MAP.

4.4 Discussion

This Chapter has sought to explore techniques for approximating the true (intractable) posterior distribution of the *dynamic tree* in contrast to sampling and annealing which are computationally costly. The simplest of the variational methods, mean field, has

been considered and both qualitative and quantitative comparisons of its performance has been made against simulated annealing, using structural and KL divergence plots respectively. We shall now discuss some of the more interesting aspects this work has brought to light.

In the absence of any image data at the leaf level, the mean field approach as applied to the binary (black and white) state variables of the nodes of the network can easily be seen to be bimodal. The units of the network will correlate and drive the means to 0 (all black) or 1 (all white). There is also an unstable equilibrium with the means at exactly half way, where the slightest perturbation will drive the means to one of the two extremities – an example of spontaneous symmetry breaking. This is analogous with a ferromagnetic system below its critical temperature.

The application of image data to the leaves of the network constrains the states of nodes in the immediate layers above, but as we ascend the hierarchy it exerts a diminishing influence and in the upper levels of the network the bimodal pattern can dominate the mean field solution. Thus the nodes further away from the image still tend towards 0 or 1. This is undesirable in certain circumstances when there is similar amounts of black and white in an image as one colour will tend to dominate at the higher levels and suppress the other, and this is evident in the types of tree structures seen. This trend is illustrated in Figure 4.2 where the smaller trees in the mean field solutions in Figures 4.2(a,c) are compressed and make less use of the hierarchical node arrangement than the corresponding MAP models in Figures 4.2(d,f). Though this is a limitation, we do still see interesting structures in the mean field trees and they can still model variable sized regions as distinct trees of nearly appropriate size, though they capture them less precisely than methods which find the MAP configuration.

The fact that the posterior of the HPMF trees is usually lower than the MAP tree is to be expected as in the sampling approaches such as simulated annealing we are setting out to find the MAP structure whereas for mean field we are concerned with averaging over the whole distribution. However we observed in the histogram plots in Figure 4.3(b) that most mean field solutions are only slightly worse.

Simulated annealing was used as it finds good solutions which are of primary interest,

however Metropolis sampling approaches could give a good insight into the true nature of the posterior distribution and is an avenue for further research.

We can conclude that the mean field approach provides significant advantages over structure searching for the MAP solution in that it produces an approximating distribution to the posterior, which is more informative than simply choosing a single example. Mean field was also able to find good HPMF solutions that rivalled the MAP structures found by simulated annealing. This was achieved with a considerable saving in computational effort and comes close to making real time inference in DTs viable.

We note however, that the assumption in mean field of a factorised distribution over $P(\mathbf{X})$ is not necessarily a good one, and an important direction for future work would be to focus further on distributions giving a closer approximation to the true posterior. An interesting extension to the standard mean field approach outlined above might be to specify an alternative distribution over the x_i^ks that more realistically captures their relation. One possibility of using a tree structure to reflect their hierarchical dependence upon each other is considered in Storkey (2000).

In order to apply the *dynamic tree* model to larger, more useful problems an element of learning at least some of the model parameters will be required – as we are unlikely to know the best parameters a priori. Given the success of the mean field approach at finding good trees and its significant speed advantage over annealing and sampling based techniques, mean field is a very attractive candidate for the inference engine needed to underpin an Expectation-Maximisation (EM) style learning algorithm. The next Chapter explores learning in the *dynamic tree* with a particular emphasis on mean field based approaches as a precursor to applying the *dynamic tree* to a database of real-world images in the subsequent Chapter.

Chapter 5

Learning the Dynamic Tree Parameters

5.1 Introduction

The *dynamic tree* model we have seen has two distinct sets of parameters, $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. The $\boldsymbol{\phi}$ parameters govern the construction of the prior $P(\mathbf{Z})$, whereas the $\boldsymbol{\theta}$s are state transition probability tables defining transitions between connected nodes. In previous Chapters we have been content to fix these by hand using our best intuitions, however for the larger models needed for real data this will almost certainly be sub-optimal, and a process for *learning* good parameters for the model from the data is required.

This Chapter[1] is concerned with developing a full suite of learning algorithms for the *dynamic tree* parameters. It builds on the theory of the previous Chapters and we shall consider both exact and approximation methods. We take an incremental approach to learning the parameters.

Ideally we would like to operate on the true *dynamic tree* probability distribution and optimise the model parameters so as to maximise the likelihood, but this is intractable for anything other than trivial structures. Mean field seen in the previous Chapter provides a useful alternative. However though it performed well in inference tasks as it only approximates the true posterior there is no guarantee it will be useful for learning

[1] Sections 5.3–5.4 of this Chapter have been published in Adams *et al.* (2001)

the *dynamic tree* parameters, so initially we consider learning of the CPTs $\boldsymbol{\theta}$, in small fixed architecture models with binary images. The fixed architecture is important as exact learning on the true posterior distribution is tractable and it allows us to make a comparison between exact learning and approximation techniques based upon the mean field algorithm derived in the previous Chapter. Furthermore for small trees we can fully enumerate the exact probability distribution of the data by summing over all possible patterns and this gives us an effectively infinite dataset. Thus learning can be evaluated under the most favourable conditions possible and enables us to ascertain the best that any algorithm is likely to perform.

We then increase the model complexity by allowing disconnections. This is an intermediate step between the fixed architecture model and the full *dynamic tree*, extending the model's capabilities while still tractable for exact approaches, and we compare again mean field with exact EM.

Optimisation algorithms for the prior $\boldsymbol{\phi}$ parameters completes the *dynamic tree* learning. Since we are now dealing with multiple tree structures exact approaches are intractable even for small models and we can no longer make a comparison. To assess the mean field algorithm we examine the stability of the learning rule by seeing whether it will deviate when initialised at the same parameters as the generative model used to create the training data, and also perturb its parameters to check that it does indeed learn from the data.

Thus we explore the characteristics of all of the individual facets of the *dynamic tree* model ($\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ parameters). This provides the necessary preparation for the task of applying *dynamic trees* to real image datasets, which will be described in Chapter 6.

Section 5.2 describes the fundamentals of the Expectation-Maximisation (EM) algorithm which lies at the heart of all the learning approaches considered in this Chapter. Section 5.3 then derives exact and mean field learning rules for the CPTs and compares their relative performance. Disconnections are introduced in Section 5.4 and in Section 5.5 learning of the prior is considered.

5.2 Overview of Learning

The *dynamic tree* model can be viewed as a mixture model where the mixture components are the set of possible tree configurations. Thus the probability of seeing a particular image vector $\mathbf{X}_v = \mathbf{v}$ under the model is given by

$$\begin{aligned} P(\mathbf{X}_v = \mathbf{v}) &= \sum_{\mathbf{Z}} P(\mathbf{X}_v = \mathbf{v}|\mathbf{Z})P(\mathbf{Z}) \\ &= \sum_{\mathbf{X}_h,\mathbf{Z}} P(\mathbf{X}_v = v, \mathbf{X}_h|\mathbf{Z})P(\mathbf{Z}) \end{aligned} \tag{5.1}$$

For the model parameters we define $\boldsymbol{\theta}$ as the parameter representing the CPTs and priors over roots, and $\boldsymbol{\phi}$ the affinities. Noting that $P(\mathbf{Z})$ is dependent on the $\boldsymbol{\phi}$ parameters, and $P(\mathbf{X}_v = v, \mathbf{X}_h|\mathbf{Z})$ only on the $\boldsymbol{\theta}$s, then Equation (5.1) becomes

$$P(\mathbf{X}_v = \mathbf{v}|\boldsymbol{\theta},\boldsymbol{\phi}) = \sum_{\mathbf{X}_h,\mathbf{Z}} P(\mathbf{X}_v = \mathbf{v}, \mathbf{X}_h|\mathbf{Z},\boldsymbol{\theta})P(\mathbf{Z}|\boldsymbol{\phi}) \tag{5.2}$$

$\mathbf{Z}$ is defined to be a matrix composed of row vectors describing each child node's connectivity. The elements of each vector are binary indicator variables which have a value 1 if the link is present, and zero otherwise. Thus $\mathbf{Z}$ describes a single structural configuration.

Now consider a data set of labelled images $\mathbf{X}_v^n, n = 1,2,\ldots,N$, and a *dynamic tree* model of $1,\ldots,U$ nodes, having a total of S forests of tree structures. Then the log-likelihood of these images (assuming i.i.d) is

$$\begin{aligned} \log P(\mathbf{X}_v) &= \sum_{n=1}^{N} \log P(\mathbf{X}_v^n) \\ &= \sum_{n=1}^{N} \log \sum_{s=1}^{S} \sum_{\mathbf{X}_h^{ns}} P(\mathbf{X}_v^p, \mathbf{X}_h^{ns}|\mathbf{Z}^{ns},\boldsymbol{\theta})P(\mathbf{Z}^{ns}|\boldsymbol{\phi}), \end{aligned} \tag{5.3}$$

where $\mathbf{X}_h^{ns}$ is describes states of hidden units of the *dynamic tree* , and $\mathbf{Z}^{ns}$ the structure for each of the forests s, and patterns n. This then can be used directly to derive

an update for the Expectation-Maximisation (EM) algorithm as described below in Section 5.2.1, or approximated using variational techniques (Section 5.2.2) which are useful when it is intractable to enumerate the full posterior.

For anything other than fixed architecture or disconnecting trees (see Section 5.4) the latter is invariably the case. The ability in the *dynamic tree* of the nodes to chose their own parents drastically increases (exponentially with tree depth) the number of configurations we need to enumerate, and even a small *dynamic tree* of depth 4 based upon a binary $(1-2-4-8)$ node arrangement has a total of 1.27×10^8 distinct structures. This means that we cannot ordinarily use exact approaches and have to resort to approximate inference methods such as mean field. In this Chapter we are primarily concerned with comparing exact and mean field learning so restrict ourselves to small structures where exact approaches are tractable. Below we derive a learning rule for the true posterior in Section 5.2.1, and in Section 5.2.2 we consider the mean field approach.

5.2.1 Learning from the True Posterior

For complex models it is not usually possible to calculate analytically the exact solution of the parameters using a standard maximum likelihood approach. This arises when we find the maximum likelihood solution constituting a set of highly non-linear coupled equations where we cannot separate out the parameters individually. An investigation into maximum likelihood leads us to consider an iterative approach based on initially "guessing" a set of parameters to the model and then minimising an error function using these "old" parameters to decouple the equations. Such a procedure has seen widespread use in the literature and is known as the *Expectation-Maximisation* (EM) algorithm.

It was mentioned earlier that the *dynamic tree* can be likened to a mixture model so we can derive the EM algorithm for the DT along similar lines. Many authors have dealt with EM on mixture-type models (eg. Ghahramani and Hinton (1996)) and we base our derivation on the one used in Bishop (1995).

The log-likelihood of the data under the model provides a convenient measure of fit of the model to the data. From Equation (5.3) for the full DT model parameters $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, this is given by

$$\log P(\mathbf{X}_v|\boldsymbol{\theta},\boldsymbol{\phi}) = \sum_{n=1}^{N} \log \sum_{s=1}^{S} \sum_{\mathbf{X}_h^{ns}} P(\mathbf{X}_v^n, \mathbf{X}_h^{ns}|\mathbf{Z}^{ns},\boldsymbol{\theta}) P(\mathbf{Z}^{ns}|\boldsymbol{\phi}) \tag{5.4}$$

Now observe the change in likelihood when we update the current model parameters ($\boldsymbol{\theta}$, $\boldsymbol{\phi}$) with a new set ($\tilde{\boldsymbol{\theta}}$,$\tilde{\boldsymbol{\phi}}$). For clarity of notation we shall omit reference to the dependency of $\mathbf{X}_h^{ns}$ on the structure $\mathbf{Z}^{ns}$ and simplify the summation over structures to $\sum_{\mathbf{Z}^n}$

$$\begin{aligned}
\log P(\mathbf{X}_v|\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\phi}}) - \log P(\mathbf{X}_v|\boldsymbol{\theta},\boldsymbol{\phi}) &= \sum_n \log \left\{ \frac{P(\mathbf{X}_v^n|\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\phi}})}{P(\mathbf{X}_v^n|\boldsymbol{\theta},\boldsymbol{\phi})} \right\} \\
= \sum_n \log \left\{ \frac{\sum_{\mathbf{Z}^n,\mathbf{X}_h^n} P(\mathbf{X}_v^n, \mathbf{X}_h^n|\mathbf{Z}^n,\tilde{\boldsymbol{\theta}}) P(\mathbf{Z}^n|\tilde{\boldsymbol{\phi}})}{P(\mathbf{X}_v^n|\boldsymbol{\theta},\boldsymbol{\phi})} \frac{P(\mathbf{Z}^n, \mathbf{X}_h^n|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi})}{P(\mathbf{Z}^n, \mathbf{X}_h^n|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi})} \right\}
\end{aligned} \tag{5.5}$$

Making use of Jensen's inequality, which states that given a set of numbers $\lambda_j \geq 0$ such that $\sum_j \lambda_j = 1$, then

$$\log\Big(\sum_j \lambda_j x_j\Big) \geq \sum_j \lambda_j \log x_j \tag{5.6}$$

and noting that $\sum_{\mathbf{Z}^n,\mathbf{X}_h^n} P(\mathbf{Z},\mathbf{X}_h|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi})$ in the numerator sums to unity, this gives

$$\begin{aligned}
\log P(\mathbf{X}_v|\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\phi}}) - \log P(\mathbf{X}_v|\boldsymbol{\theta},\boldsymbol{\phi}) &\geq \sum_n \sum_{\mathbf{Z}^n,\mathbf{X}_h^n} P(\mathbf{Z}^n, \mathbf{X}_h^n|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi}) \\
&\quad \times \log \left\{ \frac{P(\mathbf{X}_v^n, \mathbf{X}_h^n|\mathbf{Z}^n,\tilde{\boldsymbol{\theta}}) P(\mathbf{Z}^n|\tilde{\boldsymbol{\phi}})}{P(\mathbf{X}_v^n|\boldsymbol{\theta},\boldsymbol{\phi}) P(\mathbf{Z}^n, \mathbf{X}_h^n|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi})} \right\}
\end{aligned} \tag{5.7}$$

We can rewrite the right-hand side of Equation (5.7) in terms of the new parameter estimates $\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\phi}}$ as

$$B(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\phi}}, \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{n, \mathbf{Z}^n, \mathbf{X}_h^n} P(\mathbf{Z}^n, \mathbf{X}_h^n | \mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\phi}) \log P(\mathbf{Z}^n, \mathbf{X}_v^n, \mathbf{X}_h^n | \tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\phi}}) + Const \qquad (5.8)$$

where *Const* contains terms depending solely on the old parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. $B(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\phi}}, \boldsymbol{\theta}, \boldsymbol{\phi})$ is known as the *expected complete data log-likelihood*. By maximising $B(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\phi}}, \boldsymbol{\theta}, \boldsymbol{\phi})$ w.r.t $\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\phi}}$ we will obtain a new estimate of the model parameters which is guaranteed to not reduce the log-likelihood, and we can iteratively repeat this process to find a local maximum.

This is the basic EM algorithm for the *dynamic tree*. There are two sets of parameters defining this model. The $\boldsymbol{\theta}$s describe the state transitions between connected links, and the $\boldsymbol{\phi}$s define the prior over the connections. In Section 5.3 we derive and experiment on rules that learn the $\boldsymbol{\theta}$s. We keep the notation general so that there is no need to re-cover the same ground as we increase the model complexity up to the full *dynamic tree*.

5.2.2 Approximating the Posterior

We now derive a learning rule based upon the mean field approximation for *dynamic trees*. Such a rule is desirable as we have seen in the previous section that the EM approach requires knowledge of the normalised posterior which is usually intractable. One approach to this is to sample from the posterior distribution which is itself computationally expensive. Variational techniques, of which mean field is the most basic, approximate the posterior by a simpler tractable distribution.

Consider again the log-likelihood of a labelled set of images X_v^n where $n = 1, \ldots, N$

$$\log P(\mathbf{X}_v) = \sum_n \log \sum_{\mathbf{X}_h^n, \mathbf{Z}^n} P(\mathbf{X}_v^n, \mathbf{X}_h^n, \mathbf{Z}^n | \boldsymbol{\theta}, \boldsymbol{\phi})$$

$$
\begin{aligned}
&= \sum_n \log \sum_{\mathbf{X}_h^n, \mathbf{Z}^n} Q(\mathbf{X}_h^n, \mathbf{Z}^n | \mathbf{X}_v^n) \frac{P(\mathbf{X}_v^n, \mathbf{X}_h^n, \mathbf{Z}^n | \boldsymbol{\theta}, \boldsymbol{\phi})}{Q(\mathbf{X}_h^n, \mathbf{Z}^n | \mathbf{X}_v^n)} \\
&\geq \sum_{n, \mathbf{X}_h^n, \mathbf{Z}^n} Q(\mathbf{X}_h^n, \mathbf{Z}^n | \mathbf{X}_v^n) \log \frac{P(\mathbf{X}_v^n, \mathbf{X}_h^n, \mathbf{Z}^n | \boldsymbol{\theta}, \boldsymbol{\phi})}{Q(\mathbf{X}_h^n, \mathbf{Z}^n | \mathbf{X}_v^n)} \qquad (5.9)
\end{aligned}
$$

Equation (5.9) is obtained by an application of Jensen's inequality.

Let $\mathcal{L}(Q, \boldsymbol{\theta}, \boldsymbol{\phi})$ denote this lower bound (variational log-likelihood (VLL)), so that

$$
\begin{aligned}
\mathcal{L}(Q, \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{n, \mathbf{X}_h^n, \mathbf{Z}^n} \Big\{ & Q(\mathbf{X}_h^n, \mathbf{Z}^n | \mathbf{X}_v^n) \log P(\mathbf{X}_v^n, \mathbf{X}_h^n, \mathbf{Z}^n | \boldsymbol{\theta}, \boldsymbol{\phi}) \\
& - Q(\mathbf{X}_h^n, \mathbf{Z}^n | \mathbf{X}_v^n) \log Q(\mathbf{X}_h^n, \mathbf{Z}^n | \mathbf{X}_v^n) \Big\} \qquad (5.10)
\end{aligned}
$$

We see that (from Jordan *et al.* (1998); Jordan and Saul (1998)) we can perform an EM type algorithm on this by starting at an initial parameter vector $[\boldsymbol{\theta}^{(0)}, \boldsymbol{\phi}^{(0)}]$ and iterating over the following 2 steps

1. Maximise the lower bound $\mathcal{L}(Q, \boldsymbol{\theta}, \boldsymbol{\phi})$ with respect to the Q distribution (E step).
2. Fix Q and maximise the bound $\mathcal{L}(Q, \boldsymbol{\theta}, \boldsymbol{\phi})$ with respect to the parameters $\boldsymbol{\theta}$ for the CPTs, and $\boldsymbol{\phi}$ for the prior (M step).

This is coordinate ascent in $\mathcal{L}(Q, \boldsymbol{\theta}, \boldsymbol{\phi})$. We can relate this to the traditional EM algorithm by noting that for fixed Q the right-hand side of Equation (5.10) is a function of $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ only through the $P(\mathbf{X}_v^n, \mathbf{X}_h^n, \mathbf{Z}^n | \boldsymbol{\theta}, \boldsymbol{\phi})$ term. Thus the M step is equivalent to maximising the following function

$$
\sum_{n, \mathbf{X}_h^n, \mathbf{Z}^n} Q(\mathbf{X}_h^n, \mathbf{Z}^n | \mathbf{X}_v^n) \log P(\mathbf{X}_v^n, \mathbf{X}_h^n, \mathbf{Z}^n | \boldsymbol{\theta}, \boldsymbol{\phi}) \qquad (5.11)
$$

This is the variational equivalent to what is known as the *expected complete log-likelihood* in the EM literature. In the traditional presentation of the EM algorithm

the M step is defined by this same maximisation, but uses the true posterior in place of the approximating Q distribution. There are plenty of examples in the literature of its use – see for instance Neal and Hinton (1998); MacKay (1995).

These two approaches provide the framework for the learning algorithms discussed in this Chapter. We could also consider gradient descent on both the true and variational distributions, but to start with we shall only use EM.

Learning the CPTs in fixed architecture trees has already been investigated by a number of authors (eg. Feng and Williams (1998); Laferté *et al.* (2000)), and a sensible starting point would therefore be to compare this with mean field learning in fixed structures. This will be addressed in the next Section.

5.3 Learning the CPTs of a Fixed Architecture Tree

5.3.1 An Expectation-Maximisation Update for the CPTs

We wish to maximise $B(\tilde{\boldsymbol{\theta}},\boldsymbol{\theta})$ defined by Equation (5.8) and shown in Section 5.2.1 to be the condition which maximally increases the lower bound on the log-likelihood for the re-estimated parameters $\tilde{\boldsymbol{\theta}}$

$$B(\tilde{\boldsymbol{\theta}},\boldsymbol{\theta}) = \sum_n \sum_{\mathbf{Z}^n,\mathbf{X}_h^n} P(\mathbf{Z}^n,\mathbf{X}_h^n|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi}) \log P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\phi}}) \tag{5.12}$$

This is known as Baum's auxiliary function, and we need to constrain the maximisation so as to ensure legal CPT entries. To do this define $W(\tilde{\boldsymbol{\theta}},\boldsymbol{\theta}) = B(\tilde{\boldsymbol{\theta}},\boldsymbol{\theta}) - \sum_{ijl} \lambda_{ijl} \left(\sum_k \tilde{\theta}_{ij}^{kl} - 1 \right)$ where the summations k and l are over the $1,\ldots,C$ states of the nodes i and j respectively. Differentiating with respect to θ_{ij}^{kl} (the CPT entry $P(x_i = k|x_j = l)$, where we use indicator variables z_{ij} as before to pick out the parent $pa_i = j$) then gives

$$\frac{\partial W(\tilde{\boldsymbol{\theta}},\boldsymbol{\theta})}{\partial \tilde{\theta}_{ij}^{kl}} = \sum_{n,\mathbf{Z}^n,\mathbf{X}_h^n} \frac{P(\mathbf{Z}^n,\mathbf{X}_h^n|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi})}{P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\phi}})} \frac{\partial}{\partial \tilde{\theta}_{ij}^{kl}} \left[P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\phi}}) \right] - \lambda_{ijl}$$

$$
\begin{aligned}
&= \sum_{n,\mathbf{Z}^n,\mathbf{X}_h^n} \frac{P(\mathbf{Z}^n,\mathbf{X}_h^n|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi})P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\phi}})}{P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\phi}})\tilde{\theta}_{ij}^{kl}} \delta(x_i^n,k)\delta(x_j^n,l)z_{ij}^n - \lambda_{ijl} \\
&= \sum_{n,\mathbf{Z}^n,\mathbf{X}_h^n} \frac{P(\mathbf{Z}^n,\mathbf{X}_h^n|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi})}{\tilde{\theta}_{ij}^{kl}} \delta(x_i^n,k)\delta(x_j^n,l)z_{ij}^n - \lambda_{ijl} \\
&= \sum_{n,\mathbf{Z}^n} \frac{P(\mathbf{Z}^n,x_i^{kn},x_j^{ln}|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi})}{\tilde{\theta}_{ij}^{kl}} z_{ij}^n - \lambda_{ijl} \\
\Rightarrow \lambda_{ijl}\tilde{\theta}_{ij}^{kl} &= \sum_{n,\mathbf{Z}^n} z_{ij}^n P(\mathbf{Z}^n,x_i^{kn},x_j^{ln}|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi}) \\
\sum_{k=1}^{C} \lambda_{ijl}\tilde{\theta}_{ij}^{kl} &= \sum_{k=1}^{C}\sum_{n,\mathbf{Z}^n} P(\mathbf{Z}^n,x_i^{kn},x_j^{ln}|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi}) z_{ij}^n \\
\Rightarrow \lambda_{ijl} &= \sum_{k=1}^{C}\sum_{n,\mathbf{Z}^n} P(\mathbf{Z}^n,x_i^{kn},x_j^{ln}|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi}) z_{ij}^n
\end{aligned} \tag{5.13}
$$

as $\sum_{k=1}^{C} \theta_{ij}^{kl} = 1$.

The update for each entry in the CPT is therefore

$$
\hat{\theta}_{ij}^{kl} = \frac{\sum_{n,\mathbf{Z}^n} P(\mathbf{Z}^n,x_i^{kn},x_j^{ln}|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi}) z_{ij}^n}{\sum_{n,\mathbf{Z}^n}\sum_{k'} P(\mathbf{Z}^n,x_i^{k'n},x_j^{ln}|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi}) z_{ij}^n} \tag{5.14}
$$

with

$$
P(\mathbf{Z}^n,x_i^{kn},x_j^{ln}|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi}) = P(x_i^{kn},x_j^{ln}|\mathbf{X}_v^n,\mathbf{Z}^n,\boldsymbol{\theta})P(\mathbf{Z}^n|\mathbf{X}_v^n,\boldsymbol{\phi}) \tag{5.15}
$$

This update is a marginalisation over the joint distribution of trees and node states considering the transition between a parent node j being in state l to a child node i in state k along a connected link, normalised by the set of transition probabilities from all the states of the parent j. Thus it measures the value of a connection probability by summation of the joint distribution over all the tree configurations where the link $j \rightarrow i$ is made, so that θ_{ij}^{kl}s where this transition occurs frequently in higher probability regions become larger than those which are less used or are involved in low probability tree configurations.

$P(x_i^k, x_j^l|\mathbf{X}_v, \mathbf{Z}, \boldsymbol{\theta})$ can be found using Pearl's message passing algorithm (see Feng and Williams (1998)), and is given by the following Equation

$$P(x_i^k, x_j^l|\mathbf{X}_v, \mathbf{Z}, \boldsymbol{\theta}) = \frac{1}{\sum_{k'} \pi(pa_i^{k'})\lambda(pa_i^{k'})}\lambda(x_i^k|\boldsymbol{\theta})\theta_{ij}^{kl}\pi(pa_i^k|\boldsymbol{\theta}) \prod_{y \in s(x_i)} \lambda_y(pa_i^k|\boldsymbol{\theta}) \quad (5.16)$$

The λ-values are the probability of observing the evidence in the subtree beneath a particular node given that the node is in a state k, and the π-values are an estimate of the probability of a node being in a particular state given the evidence in the rest of the network above it. Their normalised product produces a consistent estimate of the probabilities of a node being in each of the states it can take on. Equation (5.16) obtains a local estimate of the joint probability of a transition from state l to k along a given link, using the λ-value of the child node i, the π-value of the parent j, along with the λ-messages of all the siblings of i.

As for the case of the affinities it is also sensible to share CPT parameters considering the limited training data available. Typically we share CPTs on a level by level basis, as it is not an unreasonable to assume stationarity of the transition probabilities across the whole image at a particular scale. In the *dynamic tree* node i can choose from a number of parents, and their CPTs are also shared.

If we define X_I as the set containing all nodes x_i sharing the same CPT, then

$$\hat{\theta}_I^{kl} = \frac{\sum_{n,\mathbf{Z}^n} \sum_{x_i \in X_I} P(\mathbf{Z}^n, x_i^{kn}, x_j^{ln}|\mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\phi}) z_{ij}^n}{\sum_{n,\mathbf{Z}^n} \sum_{x_i \in X_I} \sum_{k'} P(\mathbf{Z}, x_i^{k'n}, x_j^{ln}|\mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\phi}) z_{ij}^n} \quad (5.17)$$

The difference between this and the unshared case is simply a summation over all the nodes sharing the same CPT entry.

The complete EM algorithm for the *dynamic tree* is as follows.

1. Start with an initial estimation of parameters $\boldsymbol{\theta}$.
2. Calculate the posterior marginals $P(\mathbf{Z}^n, x_i^{kn}, x_j^{ln}|\mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\phi})$ using Pearl style message passing algorithm for the $\mathbf{Z}^n = 1, \ldots, S$ structures.

3. Use re-estimation formula given by Equation (5.14) to find new estimate for CPTs, $\hat{\boldsymbol{\theta}}$.

4. Repeat steps 2–3 using $\hat{\boldsymbol{\theta}}$ from previous step as the current starting point. Run until convergence is reached.

The joint distribution $P(\mathbf{Z}^n, x_i^{k'n}, x_j^{ln} | \mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\phi})$ like the posterior cannot tractably be normalised, making it necessary to resort to sampling to approximate this EM update for anything other than the simplest of structures. Since S, the number of possible configurations of the *dynamic tree* grows exponentially with tree depth this rules out all but the most trivial of *dynamic trees* (see Section 3.2.1.5 for a discussion). Hence a fixed architecture tree will be considered.

To start with we are using a single fixed structure tree which is equivalent to setting a prior $\pi_{ij} = \delta(pa_i, j)$. The EM update is therefore calculable, and the joint distribution over the model becomes

$$P(\mathbf{X}|\boldsymbol{\theta}, \boldsymbol{\phi}) \quad = \quad \prod_{ikl} [\theta_{ipa_i}^{kl}]^{x_i^k x_{pa_i}^l} \tag{5.18}$$

5.3.2 Mean Field EM update for the CPTs

We saw in Section 5.2.2 that we can perform an EM type algorithm on $\mathcal{L}(Q, \tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\phi}})$, a lower bound for the log-likelihood of the observed data, and that this bound for the mean field *dynamic tree* approximation is given by Equation (5.10).

The derivation of mean field approximation to the *dynamic tree* is given in full in Chapter 4. Recall that the prior is modelled by the joint distribution $P(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}, \boldsymbol{\phi}) = \prod_{ij} \pi_{ij}^{z_{ij}} \prod_{ijkl} [\theta_{ij}^{kl}]^{x_i^k x_j^l z_{ij}}$ which is approximated in mean field by the factorising distribution $Q(\mathbf{X}_h, \mathbf{Z}) = Q(\mathbf{X}_h) Q(\mathbf{Z})$. A further assumption is then required, that $Q(\mathbf{X}_h)$ fully factorises into $Q(\mathbf{X}_h) = \prod_{ik} m_i^k$, where m_i^k is the mean value of node i in state k. A straightforward application of calculus and use of Lagrange Multipliers to enforce probability constraints produced the mean field equations. These are summarised in Appendix B.3, where $\mu_{ij} = \langle z_{ij} \rangle_{Q(Z)}$ is also defined.

Differentiating the bound of Equation (5.10) with respect to $\tilde{\theta}_{ij}^{kl}$, and substituting for the mean field parameters, gives the following

$$\begin{aligned}\frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial \tilde{\theta}_{ij}^{kl}} &= \sum_{n,\mathbf{X}_h^n,\mathbf{Z}^n} \frac{Q(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n)}{P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\tilde{\boldsymbol{\theta}})} \frac{\partial}{\partial \tilde{\theta}_{ij}^{kl}} \left[P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\tilde{\boldsymbol{\theta}})\right] \\ &\quad + \sum_{n,\mathbf{X}_h^n} \frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial Q(\mathbf{Z}^n)} \frac{\partial Q(\mathbf{Z}^n)}{\partial \tilde{\theta}_{ij}^{kl}} + \sum_{n,\mathbf{Z}^n} \frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial Q(\mathbf{X}_h^n)} \frac{\partial Q(\mathbf{X}_h^n)}{\partial \tilde{\theta}_{ij}^{kl}} \qquad (5.19) \\ &= \sum_n \frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial \tilde{\theta}_{ij}^{kl}}\bigg|_{explicit} + \sum_{ni'j'} \frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial \mu_{i'j'}^n} \frac{\partial \mu_{i'j'}^n}{\partial \tilde{\theta}_{ij}^{kl}} + \sum_{nik} \frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial m_i^{kn}} \frac{\partial m_i^{kn}}{\partial \tilde{\theta}_{ij}^{kl}} \qquad (5.20)\end{aligned}$$

where in Equation (5.19) we have used the chain rule to show the explicit and implicit dependencies of $\mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})$ on $\tilde{\theta}_{ij}^{kl}$ through the posterior term $P(\mathbf{X}_v^n,\mathbf{X}_h,\mathbf{Z}|\tilde{\boldsymbol{\theta}})$ and the $Q(\mathbf{Z}^n)$s and $Q(\mathbf{X}_h^n)$s which define the mean field approximation. The latter are only implicitly dependent on $\tilde{\boldsymbol{\theta}}$ through μ_{ij} and m_i^k respectively, which gives rise to Equation (5.20).

This decomposition has been used in other learning applications such as in Baldi and Pineda (1991), and is advantageous as it enables us to greatly simplify the learning rule through the following argument.

We note that in the E step of our EM algorithm we are optimising $Q(\mathbf{Z}^n)$ and $Q(\mathbf{X}_h^n)$ holding $\tilde{\boldsymbol{\theta}}$ constant. The resulting equilibrium will be at $\frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial \mu_{ij}^n}\Big|_{\tilde{\boldsymbol{\theta}}} = \frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial m_i^{kn}}\Big|_{\tilde{\boldsymbol{\theta}}} = 0$, a local maximum in the mean field equations for the current $\tilde{\boldsymbol{\theta}}$. Therefore the implicit terms do not contribute at all to Equation (5.20) and the gradient becomes

$$\frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial \tilde{\theta}_{ij}^{kl}} = \frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial \tilde{\theta}_{ij}^{kl}}\bigg|_{explicit} \qquad (5.21)$$

From Equation (5.10)

$$\frac{\partial \mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}})}{\partial \tilde{\theta}_{ij}^{kl}}\bigg|_{explicit} = \sum_{n,\mathbf{X}_h^n,\mathbf{Z}^n} \frac{Q(\mathbf{X}_h^n)Q(\mathbf{Z}^n)}{P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\tilde{\boldsymbol{\theta}})} \frac{\partial}{\partial \tilde{\theta}_{ij}^{kl}} \left[P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\tilde{\boldsymbol{\theta}})\right]$$

$$= \sum_{n,\mathbf{X}_h^n,\mathbf{Z}^n} \frac{Q(\mathbf{X}_h^n)Q(\mathbf{Z}^n)P(\mathbf{Z}^n)}{P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\tilde{\boldsymbol{\theta}})} \frac{\partial P(\mathbf{X}_v^n,\mathbf{X}_h^n|\mathbf{Z}^n,\tilde{\boldsymbol{\theta}})}{\partial \tilde{\theta}_{ij}^{kl}} \tag{5.22}$$

We need to ensure that the CPTs sum row-wise to unity in order to be legal probabilities. So as in Section 5.3.1 we introduce the constraint $\sum_{ijl} \lambda_{ijl} \left(\sum_k \tilde{\theta}_{ij}^{kl} - 1 \right)$ and solve

$$\frac{\partial}{\partial \tilde{\theta}_{ij}^{kl}} \left[\mathcal{L}(Q^n,\tilde{\boldsymbol{\theta}}) - \sum_{ijl} \lambda_{ijl} \left(\sum_k \tilde{\theta}_{ij}^{kl} - 1 \right) \right] = 0 \tag{5.23}$$

We find during the working that once again the $P(\dots)$ terms cancel, and get

$$\lambda_{ijl} = \sum_{k'} \sum_{n,\mathbf{Z}^n} Q(x_i^{k'n})Q(x_j^{ln})Q(\mathbf{Z}^n)z_{ij}^n \tag{5.24}$$

and the mean field EM update equation for the CPTs is given by

$$\hat{\theta}_{ij}^{kl} = \frac{\sum_{n,\mathbf{Z}^n} Q(x_i^{kn})Q(x_j^{ln})Q(\mathbf{Z}^n)z_{ij}^n}{\sum_{k'} \sum_{n,\mathbf{Z}^n} Q(x_i^{k'n})Q(x_j^{ln})Q(\mathbf{Z}^n)z_{ij}^n} \tag{5.25}$$

In the case of the fixed architecture tree the nodes have no choice over parent so the z_{ij}s are superfluous and the prior $P(Z) = \prod_{ij} \delta(pa_i, j)^{z_{ij}}$. Equation (5.25) then simplifies to

$$\hat{\theta}_{ipa_i}^{kl} = \frac{\sum_n Q(x_i^{kn})Q(x_{pa_i}^{ln})}{\sum_{k'} \sum_n Q(x_i^{k'n})Q(x_{pa_i}^{ln})} \tag{5.26}$$

5.3.3 Comparing Mean Field EM with Exact EM for Fixed Architectures

We now make comparisons between the exact and variational approaches discussed in the previous Sections. The model used was of fixed balanced architecture and therefore with the same characteristics as in Feng and Williams (1998). We chose a 4 layer binary $(1-2-4-8)$ configuration which produces 1-d, 8 pixel images. The nodes were restricted to binary state variables giving a maximum of 256 different images potentially generatable from the model. The aim was to use a sufficiently complex model while still being able to calculate their exact frequency, and hence $P(\mathbf{X}_v|\boldsymbol{\theta},\boldsymbol{\phi})$.

In making comparisons between the true EM and mean field EM learning rules the basic strategy was to choose a generative model and perform two sets of runs. The first used an initial starting position the same as that of the generative model and tested the stability of the algorithms by looking for any deviation from the true parameters during learning. The second used a starting position with parameters perturbed from that of the generative model, examining the ability of the algorithm to learn the parameters of the generative model.

The choice of dataset here is important. Initially we are interested in seeing the best that the algorithms can perform so as to discover their limitations. The choice of model that can generate only 256 different images enables evaluation of all images in turn, and the update for each in the learning algorithm can be weighted by the probability of the particular pattern under the generative model. This effectively gives an infinite training set so that it would be hoped that if the learning algorithm is reasonable it should recover the generative model parameters exactly.

Selecting the generative model is not straightforward. There is a dependency between the CPT entries and the prior at the root of the tree in that any degeneracy between states from a child to its parent can be captured either way. This becomes even more pronounced when latterly disconnections lower down the tree are allowed (Section 5.4).

Setting a uniform prior $(0.5\ 0.5)$, on the root node was found to be helpful to mean

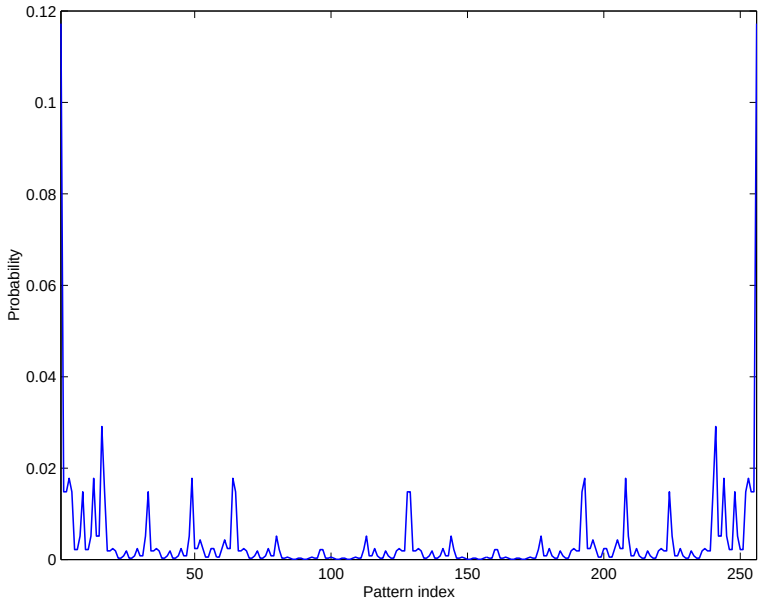
field learning as any bias to one state or another in the prior tended – by virtue of the fact that the mean field variational method assumes independence between the nodes by use of a factorised distribution – to be captured as degenerate CPTs in all layers above the leaf level. The leaf level CPTs being close to the instantiated units were better constrained by the data.

The CPT entries were set to be 0.9 on the diagonal and 0.1 off diagonal, as it is a reasonable expectation that ordinarily a child would prefer to be in the same state as its parent. (With the full *dynamic tree* architecture the child nodes can choose suitable parents so that this would be the case.) These then give a generative model which though simple is of the sort of form we would anticipate to be similar in nature to what we would expect the full *dynamic tree* with real datasets to take on. This generates data with the probability distribution shown in Figure 5.1(a). The peaks correspond with patterns where whole sub-trees span regions in the image of the same pixel value, a facet of the fixed balanced architecture and highlights nicely the "blockiness" difficulty with such structures.

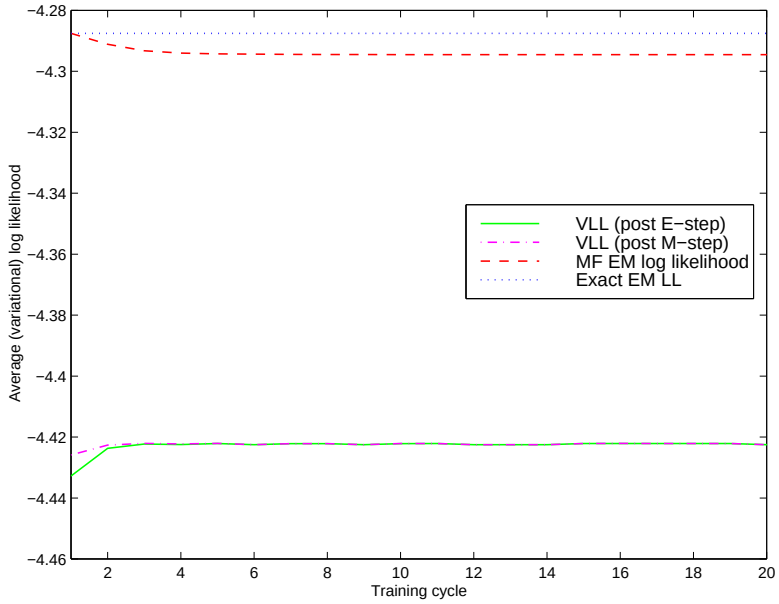
Two runs were made with this model. The first was with the same starting parameters as the generative model to test any deviation of the parameters from the true model. The results are plotted in Figure 5.1(b). A second run had perturbed parameters, achieved by setting the CPTs to 0.7 down the diagonal and 0.3 off-diagonal. This is shown in Figure 5.2(a).

Each run was performed for 300 training cycles so as to be sure that convergence was reached. Only the initial part of these runs prior to convergence is plotted in the figures. The variational log-likelihood of the mean field approach before and after the M step is shown, as also is the true log-likelihood of both the mean field and exact EM approaches after each parameter update.

From Figure 5.1(b) it can be seen that the exact EM run starting at the generative model does not deviate. This is to be expected as exact EM is working on the true posterior and will not be able to find a higher likelihood solution than this given that we are effectively using an infinite data set so the posterior is exact. For mean field EM the log-likelihood decreases slightly before stabilising. This would appear worrisome until



(a)



(b)

Figure 5.1: (a) Probability of the dataset under the generative model (b) Comparison of mean field with exact EM learning starting at generative model for generative model CPTs 0.9 and uniform prior at the root.

it is noted that with variational methods we are only approximating the true posterior and though it was shown in Chapter 4 that mean field performed rather well, the fact

that it uses a fully factorised distribution whereas the true joint distribution is definitely not fully factorisable makes it far from perfect.

To see the implications of this consider what the log-likelihood of the data equates to. This is given in the Equation below.

$$\begin{aligned}\sum_n \log P(\mathbf{X}_v^n) &= \sum_{n,\mathbf{X}_h^n,\mathbf{Z}^n} Q(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n)\log\frac{P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\boldsymbol{\theta},\boldsymbol{\phi})}{Q(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n)} \\ &\quad - \sum_{n,\mathbf{X}_h^n,\mathbf{Z}^n} Q(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n)\log\frac{Q(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n)}{P(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\phi})} \\ &\triangleq \text{Variational Log-Likelihood} - KL(Q||P) \qquad (5.27)\end{aligned}$$

Thus it is not possible to find a $Q \equiv P$ and reduce the KL-divergence, $KL(Q||P)$, to zero so the M-step is then optimising with respect to a slightly different distribution to the true posterior.

The lower bound given in Section 5.2.2 uses the fact that the KL divergence between the true and approximating distributions $KL(Q||P)$ is always greater than or equal to zero, stating that the log-likelihood of the data is always greater than the variational log-likelihood. In the E-step $KL(Q||P)$ is minimised using the mean field algorithm. The Q distribution is then fixed at this point and the variational log-likelihood is maximised in the M-step to find the new optimal $\tilde{\boldsymbol{\theta}}$. This changes the model and there is no longer any guarantee that $KL(Q||P)$ remain minimal, in fact it should ordinarily increase. Thus from Equation (5.27) it can be seen that this could cause the log-likelihood of the data to decrease. This is exactly what can be seen to happen in Figure 5.1(b) where dashed-dot line is the log-likelihood of the model. The variational log-likelihood which it is maximising (Equation (5.9)) does however increase as expected.

Since mean field only approximates the true posterior by a factorising distribution and is not exact, then it is plausible to speculate that a good mean field approximation for a particular dataset could have different model parameters even to that of the true generative model. The mean field approach is biased towards a factorised distribution, so will tend towards factorised solutions as best the data will allow. This is indeed

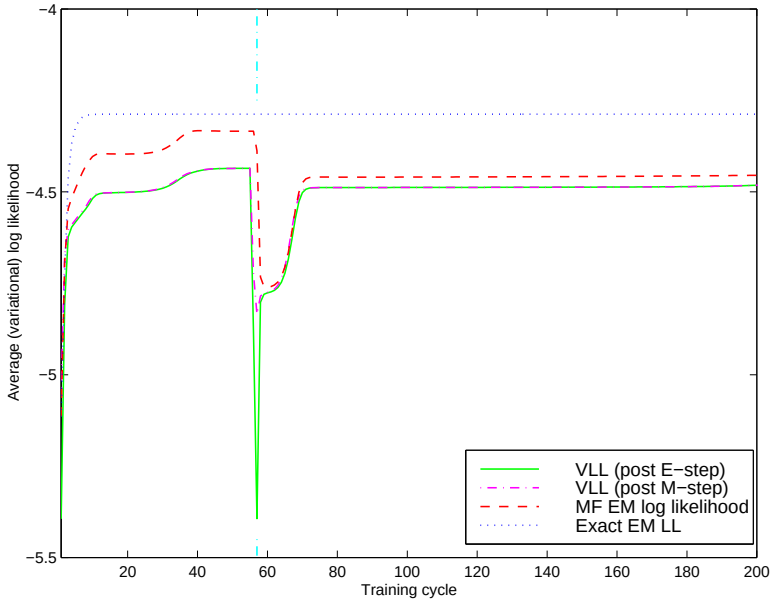
Level	Start CPT	Exact EM	Mean Field EM
1	$\begin{pmatrix} 0.5 & 0.5 \end{pmatrix}$	$\begin{pmatrix} 0.5 & 0.5 \end{pmatrix}$	$\begin{pmatrix} 0.5108 & 0.4892 \end{pmatrix}$
2	$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 0.8737 & 0.1263 \\ 0.0915 & 0.9085 \end{pmatrix}$
3	$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 0.9334 & 0.0666 \\ 0.0906 & 0.9094 \end{pmatrix}$
4	$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 0.9005 & 0.0995 \\ 0.0923 & 0.9077 \end{pmatrix}$

Table 5.1: Learned CPTs starting at the generative model.

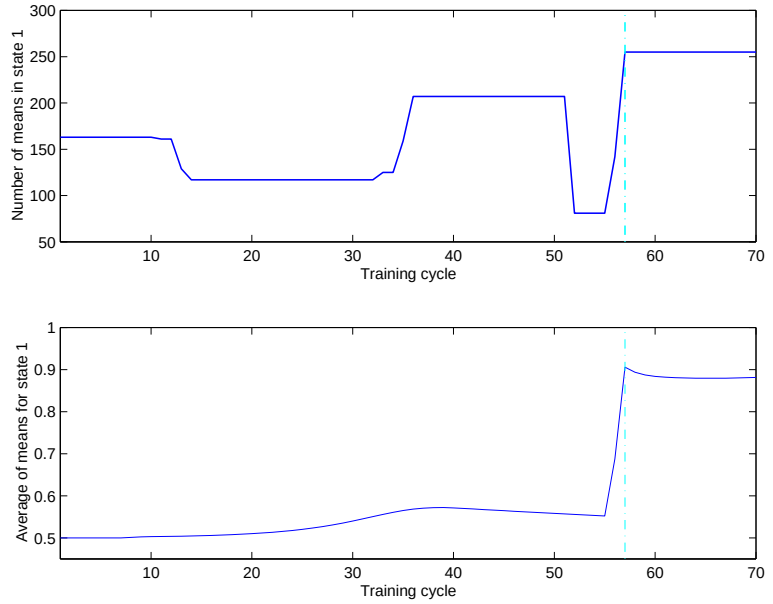
what we see happening in the plot. There is a set of parameters with higher variational log-likelihood than those of the generating model. The learning algorithm finds these and we see a slight increase in the variational free energy.

The table illustrates this. It gives the parameters learned by the different methods after 40 iterations. Level 1 parameters are those at the top of the tree, and the level 4 ones relate to the leaves at the bottom. At level 1 there is only the prior, and this is given. Lower down it is the CPTs which are relevant. As expected the True EM algorithm has not deviated at all from the initial CPT values. Examination of the final learned CPTs of mean field EM shows them to have remained fairly close with the worst probability only differing by 0.0334, and most being significantly less. The fact that they have remained close despite the mean field weaknesses, would suggest that in this instance the mean field approximation is fairly close to the true posterior.

Starting perturbed from the generative model can be seen in Figure 5.2(a) to produce quite interesting results. Exact EM again performs as expected, monotonically increasing the log-likelihood until by training cycle 26 it has recovered the generative model parameters exactly and can do no better. Ordinarily we would not necessarily expect EM to do so well, except in the limit of an infinite data set as we have here.



(a)



(b)

Figure 5.2: (a) Comparison of mean field with exact EM learning starting perturbed from the model for generative model CPT of 0.9, and uniform prior (b) Analysis of the mean of the root node during mean field learning of the CPTs showing the number out of the 256 patterns where state 1 (white) was preferred by the root node, and its average value.

Mean field EM starts off well, but on cycle 57 we see a dramatic fall in the variational log-likelihood. Under standard EM this would be a worry as the theory precludes this possibility. A close investigation of the CPTs around this point show that at cycle 55 the prior on the root has a probability of $P(White) = 0.46$, so favouring black states and of the 256 images the means found by mean field at the root node prefer to be white for only 81 patterns. In the next step $P(White) = 0.54$ and for 142 of the patterns the root node prefers being white, but the slight perturbation off the uniform prior $P(White) = P(Black) = 0.5$ pushes mean field from its unstable equilibrium of not favouring any particular state and in the next cycle $255/256$ patterns prefer to have a *white* root node.

Figure 5.2(b) illustrates this effect. It gives an analysis of the preference of the root node during mean field learning for each of the patterns. The top plot is a count of the number of patterns for which the root chooses to be in state $White$ (the mean value is greater than 0.5), and the intermediate plot is a sum of the mean values for the $White$ state. The vertical dashed line in the plots indicates cycle 57 where the drop in log-likelihood occurs. At this point the shift to choosing $White$ for the root node in $255/256$ patterns is clearly seen.

Table 5.2 shows the learned model parameters after 300 iterations. Also shown are the mean field EM model parameters a few cycles before the onset of spontaneous symmetry breaking (after 50 iterations). The resultant mean field EM prior has a $P(White) = 0.91$, and at the lower levels the CPT entry for $P(X_i = Black|Pa_i = Black)$ is forced to 1 to try to offset this. Prior to the point of spontaneous symmetry breaking the CPTs were moving towards that of the generative model. Clearly then it is possible to over-train using mean field EM even on an infinite data set, and spotting and stopping training prior to the point of spontaneous symmetry breaking could be the answer.

From Figure 5.2(a) the learning curve is seen to plateau out on a high log-likelihood for nearly 20 cycles before this onset, so the inclusion of some convergence detection criterion could be used to halt training at a point such as this and so avoid the over-training which appears to cause this phenomenon.

Overall mean field EM learning compares favourably with exact EM in fixed architec-

Level	Start CPT	Exact EM	Mean Field EM	
			50 iterations	300 iterations
1	$\begin{pmatrix} 0.5 & 0.5 \end{pmatrix}$	$\begin{pmatrix} 0.5 & 0.5 \end{pmatrix}$	$\begin{pmatrix} 0.4608 & 0.5392 \end{pmatrix}$	$\begin{pmatrix} 0.9097 & 0.0903 \end{pmatrix}$
2	$\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$	$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 0.7599 & 0.2401 \\ 0.1708 & 0.8242 \end{pmatrix}$	$\begin{pmatrix} 0.8242 & 0.1758 \\ 0.0253 & 0.9747 \end{pmatrix}$
3	$\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$	$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 0.9999 & 0.0001 \\ 0.1680 & 0.8320 \end{pmatrix}$	$\begin{pmatrix} 0.8053 & 0.1947 \\ 0.0000 & 1.0000 \end{pmatrix}$
4	$\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$	$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 0.8798 & 0.1202 \\ 0.0614 & 0.9386 \end{pmatrix}$	$\begin{pmatrix} 0.8256 & 0.1744 \\ 0.0000 & 1.0000 \end{pmatrix}$

Table 5.2: Learned CPTs starting perturbed from the generative model.

tures for this example dataset. Despite the limitations of the mean field approximation it still seems fairly robust to parameter drift for starting points at or near the generative model, and for perturbed starting positions finds solutions with log-likelihoods close to that of exact EM. Exact EM by working on the true posterior distribution should always do better.

5.4 The Disconnecting Tree

The disconnecting tree is an intermediate step between a single fixed architecture TSBN, and the *dynamic tree*. In it each node is allowed the choice of connecting to a single parent (we restrict this to the *natural* parent which is the one it would have if it were part of a balanced TSBN) or being disconnected.

There are a number of ways in which disconnections can be viewed within the *dynamic tree* framework. These shall be described with reference to inference where the bounds given by each formalism will be compared.

5.4.1 Inference in the Disconnecting Tree

5.4.1.1 Standard Disconnecting Tree

This approach is a straightforward simplification of the full *dynamic tree* model. Instead of allowing nodes to pick a parent from a set of potential candidates we restrict their choice to either the "natural parent" or the "null" parent. The indicator variable z_i is defined to be the indicator variable describing the connectivity between the child node i and its "natural parent". It takes on the value 1 for connect and 0 for disconnect. Defining P_{d_i} as the probability with which the node makes this choice then the prior for the model is as follows

$$P(\mathbf{Z}|\phi) = \prod_i P_{d_i}^{1-z_i}(1-P_{d_i})^{z_i} \tag{5.28}$$

The mean field theory discussed earlier applies, and the full CPT update Equation (5.25) can be used with the nodes only having two choices.

5.4.1.2 Effective CPTs

In the standard *dynamic tree* disconnections are modelled by having a *null* parent on each layer which a node can choose to connect to with a particular probability. This gives excellent interpretability as it is immediately obvious from a nodes' indicator vector z_i whether it has disconnected or chosen to connect to a particular parent, however for learning this presents a problem. The reason is clear, that for a disconnecting tree of n nodes with each having the choice of connecting or disconnecting, then there are 2^{n-1} distinct configurations (the top level node can only be a root), which quickly becomes intractable to enumerate.

There is an alternative way of handling disconnections which arises by viewing the prior state vector as a degenerate CPT made up of identical row vectors each a copy of the prior. By making the assumption that any degeneracy in a CPT is a contribution to the prior then it is possible to fold the prior into the CPT, and after training exactly

recover the prior, disconnection probability and CPT from the learned CPT. Defining P_d as the prior disconnection probability for a node, $\boldsymbol{\pi}$ as the prior (a row vector), and θ the CPT, then we can fold in disconnections using

$$\theta_{eff} = (1 - P_d)\theta + P_d \mathbf{1}\boldsymbol{\pi} \tag{5.29}$$

to get an effective CPT, θ_{eff}. To recover the probabilities after learning use

$$P_d = \sum_k min_l(\theta^{kl}_{eff}) \tag{5.30}$$

$$\pi_k = min_l(\theta^{kl}_{eff})/P_d \tag{5.31}$$

$$\theta = (\theta_{eff} - P_d \mathbf{1}\boldsymbol{\pi})/(1 - P_d) \tag{5.32}$$

where $\mathbf{1}$ is a column vector of 1s, and θ^{kl}_{eff} the effective CPT whose rows index the child state k, and columns the parent state l. Equation (5.30) extracts the degenerate probability component for each child state and attributes it to the disconnection probability. Equation (5.31) finds the normalised ratio of these which is the prior vector, and in (5.32) these are weighted by P_d and subtracted from the effective CPT. By folding in disconnections in this way the disconnecting tree can be represented in a single structure and learning then is identical to the fixed architecture TSBN.

The advantage of this method is that disconnections are represented compactly with the probability that any given node in the tree being disconnected more readily obtainable, than keeping it explicit where it is necessary to sum over all the tree configurations where the node is disconnected.

The disadvantage is that by merging it with the true CPT their meaning is blurred, and the only sensible reconstruction of the two is to assume that all degeneracy in the CPT is due to the prior. This means that every column in the CPT must be assumed to have at least one transition which is zero as the degenerate component of each column will be given by the smallest element, but that isn't necessarily the case. Indeed to handle noise we may wish to set each zero element to a small value.

This approach can be used in true EM or mean field EM learning. Exact EM becomes intractable as soon as nodes are allowed to chose their parents in the full *dynamic tree* and in the mean field solution disconnection probabilities become unrecoverable, and their meaning lost. The issue then is one of interpretability against concise treatment.

5.4.1.3 Exact Inference and Polytrees

Polytrees are an extension to the simple tree configuration which allows for multiple parents. They are a form of singly connected networks whose inference algorithm is described in Section 4.3 of Pearl (1988a). Instead of multiple parents however each node has a single connected parent to which it can make the usual state transitions using the CPT, and a gating variable whose prior encodes the disconnection probabilities. This gating variable thus replaces the "null" parent of the standard DT formalism.

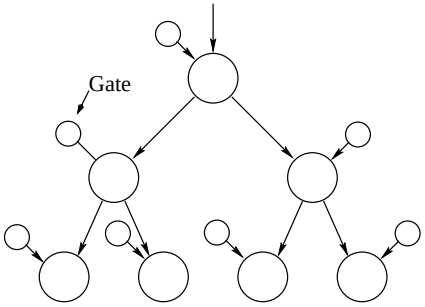


Figure 5.3: Using a gating variable polytree to signal disconnections.

The gating parent (Figure 5.3) is a node with 2 states, disconnect and connect. The probability of disconnection P_d constitutes a prior over its state, and the equilibrium belief of the gating units gives a measure of the effective disconnection probability of their child.

Polytrees are a special type of singly connected network because although we now have multiple parents we retain the basic tree structure. Thus the Pearl message passing algorithm can still be performed in two sweeps. First the λ sweep runs from leaves to root. At each stage the gating units are required to transmit a π message to its child. Then the second sweep from root to leaves completes the belief calculation and

while doing so λ messages can be passed up to the gating units and used to update the effective disconnection probabilities.

This is an elegant approach and is very attractive when exact inference is tractable. In the section below it is used to calculate the true log-likelihood of the model given an infinite dataset and compared with bounds found from mean field methods using explicit and folded-in disconnection probabilities.

5.4.2 Comparison of Disconnection Representations

The above representations were each considered using the same generative model parameters as in Section 5.3.3. Here though a prior is set at each level. The full range of disconnection probabilities were investigated, and the results plotted in Figure 5.4 show the average (variational) log-likelihood using the infinite dataset.

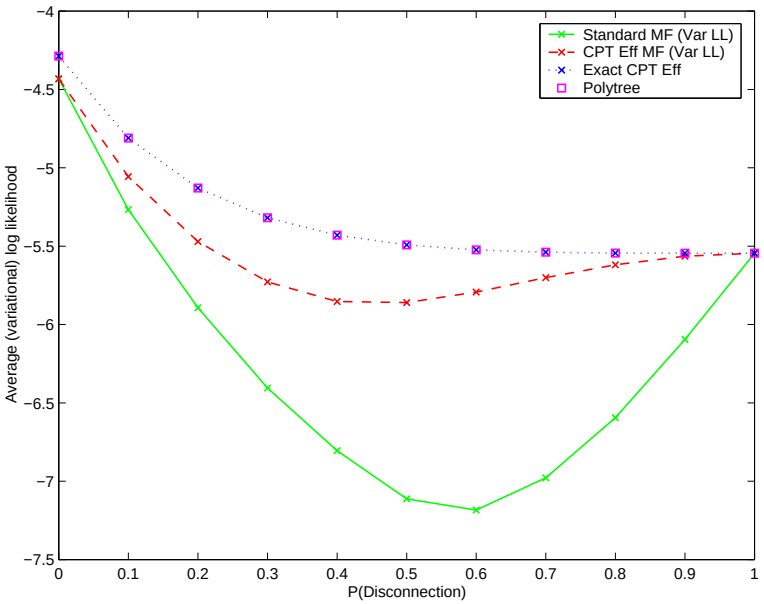


Figure 5.4: Comparison of (variational) log-likelihoods for different model representations over range of disconnection probabilities.

Exact inference methods give the exact log-likelihoods that should be expected for the given disconnection probabilities. In the plot the polytree algorithm provides this. An

alternative exact approach was also considered obtained by folding in the disconnections to create effective CPTs. This reduces the 2^{n-1} disconnecting tree configurations to a single structure making exact inference tractable. Reassuringly they can be seen to be equivalent.

For mean field, both the explicit (using $Q(Z)$s) and folded-in CPT (CPT_{eff}) representations are given. Across the full range of disconnection probabilities the folded-in representation gives higher log-likelihoods, than representing the disconnections explicitly. At first sight this might be thought surprising. The explicit representation has more degrees of freedom and so it might be thought has greater power to model the disconnecting architecture. In reality the key issue appears to be due to the nature of the mean field approximation. Mean field makes a fully factorised assumption for posterior distribution, which it has been noted is not an accurate reflection of the true posterior. By allowing $Q(Z)$s in the explicit representation gives the mean field algorithm more scope to exercise its bias towards factorised solutions – which it can't do in the effective CPT where the architecture is fixed. Thus folding in the CPTs actually gives better performance.

5.4.3 Learning in the Disconnecting Tree

Introducing disconnections to the fixed architecture produces a more interesting model, while folding them into the CPTs as described in Section 5.4.1.2 allows the 2^{n-1} configurations to be represented in a single structure making it tractable to compare mean field with exact EM, as was done with the fixed architecture model.

This representation of the CPTs gives a slightly altered interpretation to the generative model parameters. By attributing all of the degenerate components in the CPT to disconnections we get a more robust representation that removes the over-parameterisation ambiguity. The effect on the CPT is that the lowest probability elements are driven down to zero and for the binary state CPT this gives the identity matrix. The generative model with same prior and CPTs as were used in the previous experiments was considered with disconnection probabilities 0.1, 0.2 and 0.5. Con-

verting to effective CPTs using the formulae of Section 5.4.1.2 retains the uniform prior, but transforms the CPTs to the identity matrix and the effective disconnection probabilities to 0.28, 0.36 and 0.6 respectively.

Runs were made for mean field keeping the prior and CPTs distinct[2] (Standard MF EM), mean field with CPTs folded in (CPT_{eff} MF EM) and exact EM with folded in CPTs (Exact EM), for the two disconnection probabilities. Their learning curves are compared in Figure 5.5.

Standard mean field EM performs the worst in each case, giving the lowest bound. This is perhaps not surprising as by keeping the CPTs and priors distinct gives more degrees of freedom. Folding in the CPTs greatly improves the mean field performance, though exact EM is still noticeably better.

An examination of the effective disconnection probabilities found after 40 training cycles for mean field learning is interesting. They were extracted from the final CPTs using Equation (5.30) and are given in the table for levels 2 down to the leaves at level 4, for each run performed. The top level contains only the root node which is permanently disconnected and is not shown. The prior and CPT associated with each was as for the generative model, uniform with the CPT being the identity matrix.

From table 5.3 it can be seen that mean field tends to choose to make all the upper levels independent by preferring to disconnect and the structure of the CPTs learned is degenerate. In the lowest level (closest to the data) it recovers the generative model parameters exactly. This type of behaviour is not really surprising considering that mean field uses a factorised approximation. Exact EM nearly finds the generative model for low disconnection probabilities, but for high disconnections it struggles on the higher levels, getting progressively worse the further from the data the parameters are.

Table 5.4 compares the log-likelihoods of the CPT_{eff} mean field with those found by exact EM. For the runs starting at the generative model we observe the same drop in log-likelihood as was observed during fixed architecture learning, and again differ-

[2] Unlike the folded in case we are required to use the $Q(Z)$ distribution in mean field. The disconnection probabilities were fixed to avoid the over-parameterisation problem.

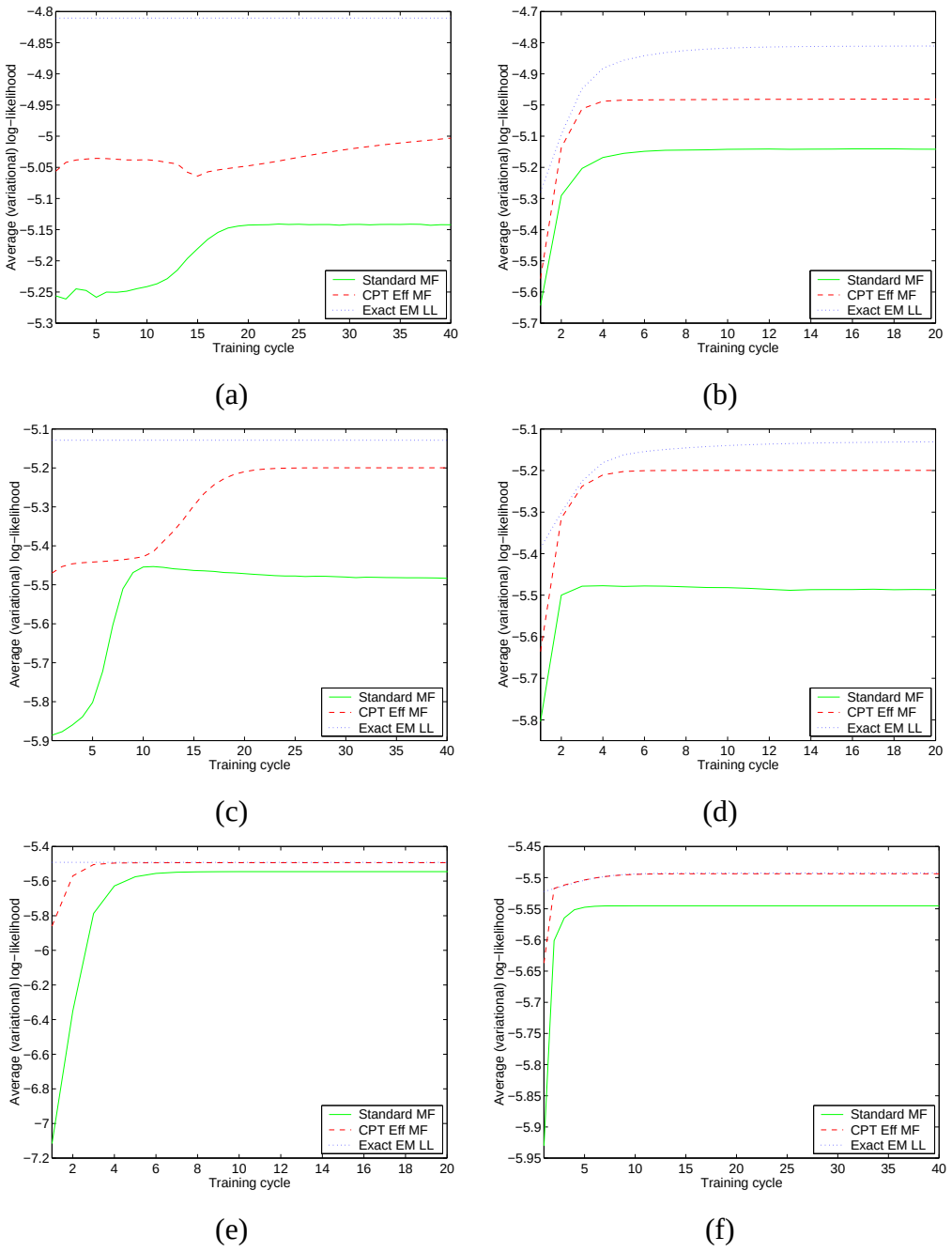


Figure 5.5: Learning of the CPTs starting at the generative model (left above), and with perturbed CPTs of 0.7 on the diagonal (right above), for generative model disconnection probabilities of 0.28 (a)–(b), 0.36 (c)–(d), and 0.6 (e)–(f).

ences between the true posterior and the mean field approximation are likely to account for this. Starting perturbed from the generative model gives rise to likelihoods which

	Target $P_d = \mathbf{0.28}$	
Level	CPT_{eff} MF	Exact EM
2	1.000	0.281
3	0.991	0.280
4	0.28	0.28

	Target $P_d = \mathbf{0.36}$	
Level	CPT_{eff} MF	Exact EM
2	1.00	0.390
3	1.00	0.355
4	0.36	0.36

	Target $P_d = \mathbf{0.6}$	
Level	CPT_{eff} MF	Exact EM
2	1.0	0.797
3	1.0	0.734
4	0.6	0.590

Table 5.3: Learned disconnection probabilities from a start CPT of 0.7, for generative model disconnection probabilities of 0.28, 0.36 and 0.6.

	At generative model		**Perturbed**	
Target P_d	CPT_{eff} MF	Exact EM	CPT_{eff} MF	Exact EM
0.28	-4.8715	-4.8105	-4.9805	-4.8105
0.36	-5.1996	-5.1288	-5.1996	-5.1289
0.6	-5.4938	-5.4917	-5.4938	-5.4924

Table 5.4: Log-likelihoods of converged models learned from a start CPTs of 0.9 and 0.7, with effective generative model disconnection probabilities of 0.28, 0.36 and 0.6.

slowly increase during the course of training converging to those shown in the table. It is interesting to see that for disconnection probabilities of 0.36 and 0.6 these are the same as for the run starting at the generative model and suggests a stable equilibrium at this configuration. This is very encouraging.

5.5 Learning the Prior Probabilities

The above has dealt with learning the CPTs and extended the work to dealing with disconnections. The *dynamic tree* model as well as having these components also sets a prior $P(\mathbf{Z}|\boldsymbol{\phi})$ over the structural configurations $\mathbf{Z}$ whose parameters $\boldsymbol{\phi}$ also need to be learned.

To complete the learning picture we now turn our attention to deriving the re-estimation formulae for these parameters. These determine the form of the distribution $P(\mathbf{Z})$ and can be viewed in a number of ways. Two such ways are as a set of discrete probabilities π_{ij} for each of the parent-child connections; or alternatively we can specify them as a function of affinity parameters a_{ij} which apportion a share of the probability mass to each of the potential parents of node i.

Learning rules for both approaches are derived below, and their relative merits discussed.

5.5.1 Considering the Prior as a set of Probabilities

For this approach we model the prior as

$$P(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{i,j=0}^{U} \pi_{ij}^{z_{ij}} \tag{5.33}$$

where U is the number of nodes in the Dynamic Tree. z_{ij} is the indicator variable denoting to which parent j the node i is connected. For a node i we can combine its associated indicator variables into a connectivity vector $\mathbf{z}_i$ which will have a 1 at the position j representing its connected parent and zeros elsewhere. z_{i0} is used to indicate a disconnected root node, and thus the size of this vector $\mathbf{z}_i$ is $U+1$. Alternatively it is a simple matter to fold in the disconnections as described in Section 5.4.1.2 by omitting them, and does not change the analysis. Here though they are included explicitly.

We wish to maximise Baum's auxilary function given by Equation (5.12) with respect

to the prior parameters. In the Equation $\boldsymbol{\phi}$ was used to denote these parameters – which for the analysis of this section we replace by the $\boldsymbol{\pi}$ used for this formulation of the prior distribution. Once again we introduce a Lagrange Multiplier to ensure legal probabilities, and maximise $B(\tilde{\boldsymbol{\pi}}, \boldsymbol{\pi}) - \sum_i \lambda_i \left(\sum_j \tilde{\pi}_{ij} - 1 \right)$

$$\begin{aligned} \frac{\partial}{\partial \tilde{\pi}_{ij}} \left[B - \sum_i \lambda_i \left(\sum_j \tilde{\pi}_{ij} - 1 \right) \right] &= \sum_{n, \mathbf{Z}^n, \mathbf{X}_h^n} \frac{P(\mathbf{Z}^n, \mathbf{X}_h^n | \mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\pi})}{P(\mathbf{Z}^n | \tilde{\boldsymbol{\pi}})} \frac{\partial P(\mathbf{Z} | \tilde{\boldsymbol{\pi}})}{\partial \tilde{\pi}_{ij}} - \lambda_i \\ &= 0 \end{aligned} \tag{5.34}$$

Where

$$\frac{\partial P(\mathbf{Z}^n | \tilde{\boldsymbol{\pi}})}{\partial \tilde{\pi}_{ij}} = z_{ij}^n \frac{P(\mathbf{Z}^n | \tilde{\boldsymbol{\pi}})}{\tilde{\pi}_{ij}} \tag{5.35}$$

The notation is again simplified as per Section 5.2.1 for reasons of clarity.

Substituting $\frac{\partial P(\mathbf{Z}^n | \tilde{\boldsymbol{\pi}})}{\partial \tilde{\pi}_{ij}}$ into Equation (5.34) gives

$$\begin{aligned} \lambda_i &= \sum_{n, \mathbf{Z}^n, \mathbf{X}_h^n} \frac{P(\mathbf{Z}^n, \mathbf{X}_h^n | \mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\pi})}{P(\mathbf{Z}^n | \tilde{\boldsymbol{\pi}})} z_{ij}^n \frac{P(\mathbf{Z}^n | \tilde{\boldsymbol{\pi}})}{\tilde{\pi}_{ij}} \\ &= \frac{1}{\tilde{\pi}_{ij}} \sum_{n, \mathbf{Z}^n} P(\mathbf{Z}^n | \mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\pi}) z_{ij}^n \\ \Rightarrow \lambda_i \tilde{\pi}_{ij} &= \sum_{n, \mathbf{Z}^n} P(\mathbf{Z}^n | \mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\pi}) z_{ij}^n \\ \Rightarrow \sum_{j=0}^{U} \lambda_i \tilde{\pi}_{ij} &= \sum_{j=0}^{U} \sum_{n, \mathbf{Z}^n} P(\mathbf{Z}^n | \mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\pi}) z_{ij}^n \\ \Rightarrow \lambda_i &= \sum_{n, \mathbf{Z}^n} P(\mathbf{Z}^n | \mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\pi}) \end{aligned} \tag{5.36}$$

Which gives rise to the update Equation

$$\hat{\pi}_{ij} = \frac{\sum_{n, \mathbf{Z}^n} P(\mathbf{Z}^n | \mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\pi}) z_{ij}^n}{\sum_{n, \mathbf{Z}^n} P(\mathbf{Z}^n | \mathbf{X}_v^n, \boldsymbol{\theta}, \boldsymbol{\pi})} \tag{5.37}$$

Intuitively we can understand this update as the sum of the contribution from the posterior of all the trees where the link z_{ij} is present, normalised by the total posterior for all the tree configurations. So links which are used more frequently will get higher probabilities than those which are less important.

We observe that the EM update includes a summation of the posterior $P(\mathbf{Z}^n|\mathbf{X}_v^n,\boldsymbol{\theta},\boldsymbol{\pi})$ over all the possible tree configurations S which is intractable for large *dynamic trees*. Thus to work on the true posterior directly it would be necessary to resort to sampling. Alternatively it could be approximated using variational techniques.

The simplicity of this learning rule makes it very attractive. However, representing the prior parameters as πs in this way is inefficient and there is likely to be far more πs than the usually limited training data would allow for good training. For a tree with a branching factor b then the nodes at level d would have $b^{(d-2)}$ parent nodes to choose from (excluding the *null* parent) and the total number of π parameters for a tree of depth D is given by

$$\sum_{d=1}^{D} b^{(d-2)} = \frac{1-b^{(D-2)}}{1-b} \tag{5.38}$$

which is exponential in tree depth. Parameter sharing would be very desirable, and by using the obvious symmetry of the network they can be reduced by $1/b$. It would be attractive to share parameters between nodes on the same level where each would have the same number of potential parents. However edge effects caused by nodes on the peripheries having less near parents on one side than those in the centre mean the πs are not identical throughout and further sharing possibilities are limited.

An obvious solution is to define the π parameters in terms of a set of hyper-parameters which map down to the appropriate probabilities through a known function. It should be possible to share these parameters with nodes on the same level and perhaps even all nodes, so drastically reducing the number of parameters needed to be learned. The affinity formulation described in the next Section is one way of achieving this.

5.5.2 Constructing the Prior from Affinities

5.5.2.1 Affinities not shared

The affinity formulation has already been seen when the *dynamic tree* was described in Chapter 3. We define the affinity prior

$$P(\mathbf{Z}|\mathbf{a}) = \prod_{ij} \left(\frac{\exp(\beta a_{ij})}{\sum_{j'} \exp(\beta a_{ij'})} \right)^{z_{ij}} = \prod_{ij} \pi_{ij}^{z_{ij}} \tag{5.39}$$

where z_{ij} the indicator variable picking out the node connections. The a_{ij}s are the affinity parameters and βs are positive constants. The advantage of using these is we have a direct measure of the preference of a node for connecting to particular parents.

The softmax in the expression for the prior in Equation (5.39) prevents us from obtaining a straightforward expression for the EM update, so we consider instead the gradient based approaches. The simplest form adjusts the model parameters so as to take small steps along a suitably defined surface S, where at each iteration τ there are update rules of the form

$$\Delta a_{ij}^{(\tau)} = \eta(\tau) \nabla S^{(\tau)}|_{a_{ij}^{(\tau)}} \tag{5.40}$$

The log-likelihood of the data provides a convenient measure with which to optimise the affinities. It is given below.

$$\log P(\mathbf{X}_v|\mathbf{a}) = \sum_n \log \sum_{\mathbf{X}_h^n, \mathbf{Z}^n} P(\mathbf{X}_v^n, \mathbf{X}_h^n|\mathbf{Z}^n, \boldsymbol{\theta}) P(\mathbf{Z}^n|\mathbf{a}) \tag{5.41}$$

By differentiating with respect to the affinities a gradient learning rule can be derived.

So

$$\begin{aligned}
\frac{\partial}{\partial a_{ij}}\left[\log P(\mathbf{X}_v|\mathbf{a})\right] &= \sum_{n,\mathbf{Z}^n} \frac{\partial \log P(\mathbf{X}_v^n|\mathbf{a})}{\partial P(\mathbf{Z}^n|\mathbf{a})} \frac{\partial P(\mathbf{Z}^n|\mathbf{a})}{\partial a_{ij}} \\
&= \sum_{n,\mathbf{Z}^n} \frac{\sum_{\mathbf{X}_h^n} P(\mathbf{X}_v^n, \mathbf{X}_h^n|\mathbf{Z}^n, \boldsymbol{\theta})}{\sum_{\mathbf{X}_h'^n, \mathbf{Z}'^n} P(\mathbf{X}_v^n, \mathbf{X}_h'^n|\mathbf{Z}'^n, \boldsymbol{\theta}) P(\mathbf{Z}'^n|\mathbf{a})} \frac{\partial P(\mathbf{Z}^n|\mathbf{a})}{\partial a_{ij}} \\
&= \sum_{n,\mathbf{Z}^n} \frac{P(\mathbf{X}_v^n|\mathbf{Z}^n, \boldsymbol{\theta}) \frac{P(\mathbf{Z}^n|\mathbf{a})}{P(\mathbf{Z}^n|\mathbf{a})}}{\sum_{\mathbf{Z}'^n} P(\mathbf{X}_v^n|\mathbf{Z}'^n, \boldsymbol{\theta}) P(\mathbf{Z}'^n|\mathbf{a})} \frac{\partial P(\mathbf{Z}^n|\mathbf{a})}{\partial a_{ij}} \\
&= \sum_{n,\mathbf{Z}^n} \frac{P(\mathbf{X}_v^n|\mathbf{Z}^n, \boldsymbol{\theta}) P(\mathbf{Z}^n|\mathbf{a})}{\sum_{\mathbf{Z}'^n} P(\mathbf{X}_v^n|\mathbf{Z}'^n, \boldsymbol{\theta}) P(\mathbf{Z}'^n|\mathbf{a})} \frac{\partial \log P(\mathbf{Z}^n|\mathbf{a})}{\partial a_{ij}} \\
&= \sum_{n,\mathbf{Z}^n} r_z^n \frac{\partial \log P(\mathbf{Z}^n|\mathbf{a})}{\partial a_{ij}} \qquad (5.42)
\end{aligned}$$

where $r_z^n = \frac{P(\mathbf{X}_v^n|\mathbf{Z}^n, \boldsymbol{\theta}) P(\mathbf{Z}^n|\mathbf{a})}{\sum_{\mathbf{Z}'^n} P(\mathbf{X}_v^n|\mathbf{Z}'^n, \boldsymbol{\theta}) P(\mathbf{Z}'^n|\mathbf{a})}$. This is the responsibility component for tree Z in the joint distribution.

Taking the log of Equation (5.39) produces the following expression for the affinities,

$$\log P(\mathbf{Z}|\mathbf{a}) = \sum_{i,j=0} z_{ij} \log \pi_{ij} \qquad (5.43)$$

which we differentiate

$$\begin{aligned}
\frac{\partial \log P(\mathbf{Z}^n|\mathbf{a})}{\partial a_{ij}} &= \frac{\partial}{\partial a_{ij}} \left[\sum_{c,d} z_{cd} \beta a_{cd} - \sum_{c,d} z_{cd} \log \sum_{d'} \exp(\beta a_{cd'}) \right] \\
&= \beta \left[z_{ij} - \sum_{c,d} z_{cd} \frac{\sum_{d'} \exp(\beta a_{cd'})}{\sum_{d''} \exp(\beta a_{cd''})} \delta_{ci} \delta_{d'j} \right] \\
&= \beta \left[z_{ij} - \sum_{c,d,d'} z_{cd} \pi_{cd'} \delta_{ci} \delta_{d'j} \right] \\
&= \beta \left[z_{ij} - \sum_{d} z_{id} \pi_{ij} \right] \\
&= \beta [z_{ij} - \pi_{ij}] \qquad (5.44)
\end{aligned}$$

as $\pi_{cd'} = \frac{\exp(\beta a_{cd'})}{\sum_{d''} \exp(\beta a_{cd''})}$ from the definition (5.39) above.

The complete gradient expression is thus given by

$$\frac{\partial \log P(\mathbf{X}_v|\mathbf{a})}{\partial a_{ij}} = \beta \sum_{n,\mathbf{Z}^n} r_z^n[z_{ij} - \pi_{ij}]. \tag{5.45}$$

The gradient of the log-likelihood with respect to the affinity parameter a_{ij} is thus a summation over all posterior trees weighted by the difference between the number of times the affinity parameter participates in an active link z_{ij}, and the connection probability of the link it represents y_{ij}. So when the affinity participates in many higher posterior tree structures we get a bigger update, but it is curtailed if the prior probability of making this link is already high.

5.5.2.2 Shared affinities

The above analysis is for the most general case of independent affinities. The *sparse dynamic tree* (see Section 3.4) extends individual affinities to templates for each layer, and this provides the ideal framework for use in learning. To explore how templates can be incorporated consider for simplicity of notation a 1-d node configuration. Ordinarily this would be a binary tree, though we are not restricted to such node arrangements. (Extension to 2-d image models is not difficult, though careful selection of node ordering is necessary.)

The most obvious form of sharing would be per layer, and would seem intuitively reasonable as nodes on the same level might be expected to perform the same roles. Let I denote the index of a set of templates $1, \ldots, D$ (1 per layer), where D is the depth of the tree. Each node sharing the template by virtue of being on the same level will have the same number of potential parents. Let I_W then denote an appropriately sized window spanning the required number of affinities to accommodate each potential parent of the children on level I, and J be an ordered set of potential parents so that the first parent on the layer has index 0. Each child node will have a different *natural* parent, so define s_i as the offset of the start of the window such that the *natural* parent of node i will have affinity a_{nat}, then for a tree with branching factor b, and current

depth d, where the top level is depth 1, these offsets are given by

$$s_i = \text{floor}\left(\frac{b^{(d-1)-i}}{b}\right) \tag{5.46}$$

Figure 5.6 below summarises these definitions. The numbering of the affinities in the Figure refers to their distance from the *natural* parent and adopting a symmetrical template would allow further sharing of the parameters, though this is not necessary.

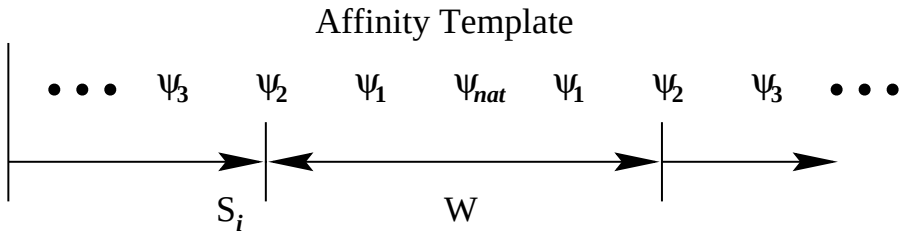


Figure 5.6: Sharing the affinities using a template.

The affinity a_{ij} from node i to parent j thus updates the template parameter $a^{I_w}_{s_i+\alpha}$ where α is the index of the parent j amongst the ordered set of parent nodes J.

We define a sharing hyper-parameter ψ such that $\frac{\partial a_{ij}}{\partial \psi} = 1$ when a_{ij} is included in the set of affinities shared by ψ, and zero otherwise.

The derivation of the log-likelihood with respect to the prior is the same as above, except that instead of differentiating the prior by the affinities a_{ij} we now use the shared parameter ψ.

$$\begin{aligned}\frac{\partial \log P(\mathbf{Z}^n|\mathbf{a})}{\partial \psi} &= \sum_{i,j} \frac{\partial \log P(\mathbf{Z}^n|\mathbf{a})}{\partial a_{ij}} \frac{\partial a_{ij}}{\partial \psi} \\ &= \beta \sum_{i,j:\{i,j\}\in\psi} [z_{ij} - \pi_{ij}]\end{aligned} \tag{5.47}$$

and the gradient becomes

$$\frac{\partial \log P(\mathbf{X}_v|\mathbf{a})}{\partial a_{ij}} = \beta \sum_{n,\mathbf{Z}^n} r_z^n \sum_{i,j:\{i,j\}\in\psi} [z_{ij} - \pi_{ij}]. \tag{5.48}$$

To extend templates to 2-d image models we might consider for example a quad-tree arrangement, which is highly suitable for 2-d image modelling. On each successive layer there will be four times as many nodes as on the previous one. Two adjacent layers are shown in Figures 5.7(a) and (b).

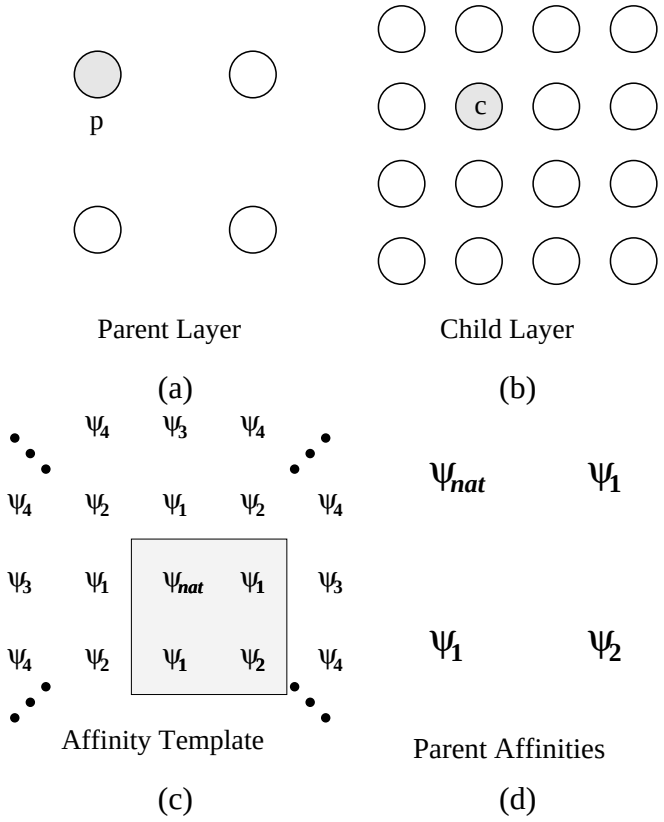


Figure 5.7: (a) Parent nodes, and (b) child nodes, from a quad-tree arrangement capable of modelling 2-d images. (c) gives the affinity template, and (d) the overlay of the parent affinities for child c.

A suitable affinity template then might take the form of that in Figure 5.7(c), where a_{nat} is the affinity of the *natural* parent, and the indices denote affinities for nodes with a distance $1, 2, \ldots$ away from the natural parent. For the child c with a *natural* parent p then overlaying the parent configuration at the right offset on the affinity template gives the set of affinities for the particular child node. This is illustrated in Figure 5.7(d)

5.5.3 Mean Field Learning Rule for Softmax Affinities

To derive the gradient for the mean field approach we cannot start with the log-likelihood as with using the true posterior. Instead we begin at the lower bound for the log-likelihood, known as the variational log-likelihood.

$$\begin{aligned}
\mathcal{L}(Q,\boldsymbol{\theta},\mathbf{a}) &= \sum_{n,\mathbf{X}_h^n,\mathbf{Z}^n} Q(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n)\log\frac{P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\boldsymbol{\theta},\mathbf{a})}{Q(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n)} \\
&= \sum_{n,\mathbf{X}_h^n,\mathbf{Z}^n} Q(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n)\log P(\mathbf{X}_v^n,\mathbf{X}_h^n|\mathbf{Z}^n,\boldsymbol{\theta}) \\
&\quad + \sum_{n,\mathbf{Z}^n} Q(\mathbf{Z}^n|\mathbf{X}_v^n)\log P(\mathbf{Z}^n|\mathbf{a}) - H(Q) \qquad (5.49)
\end{aligned}$$

Considering the shared case, then differentiate with respect to ψ

$$\begin{aligned}
\frac{\partial\mathcal{L}(Q,\boldsymbol{\theta},\mathbf{a})}{\partial\psi} &= \frac{\partial\mathcal{L}(Q,\boldsymbol{\theta},\mathbf{a})}{\partial P(\mathbf{Z}|\boldsymbol{\psi})}\frac{\partial P(\mathbf{Z}|\boldsymbol{\psi})}{\partial\psi} \\
&= \sum_{n,\mathbf{Z}^n} Q(\mathbf{Z}^n|\mathbf{X}_v^n)\frac{\partial\log P(\mathbf{Z}^n|\boldsymbol{\psi})}{\partial\psi} \\
&= \beta\sum_{n,\mathbf{Z}^n} Q(\mathbf{Z}^n|\mathbf{X}_v^n)\sum_{i,j}\left[z_{ij}-\sum_{i,d} z_{id}\pi_{ij}\right]\frac{\partial a_{ij}}{\partial\psi} \\
&= \beta\sum_n\sum_{i,j}\left[\sum_{\mathbf{Z}^n} Q(\mathbf{Z}^n|\mathbf{X}_v^n)z_{ij}-\sum_{\mathbf{Z}^n} Q(\mathbf{Z}^n|\mathbf{X}_v^n)\pi_{ij}\right]\frac{\partial a_{ij}}{\partial\psi} \\
&= \beta\sum_n\sum_{i,j:\{i,j\}\in\psi}\left[\mu_{ij}-\pi_{ij}\right] \\
&= \left(\beta\sum_n\sum_{i,j:\{i,j\}\in\psi}\mu_{ij}\right)-\beta N\Psi\pi_{ij} \qquad (5.50)
\end{aligned}$$

where N is the number of training patterns, and Ψ the number of shared affinities referenced by the hyper-parameter ψ.

5.5.4 Learning the Affinities

We now evaluate the learning rules derived for mean field learning of the affinities and once again we adopt a similar procedure to that used in the disconnecting tree and CPT learning previously discussed.

The aim is to discover whether mean field can learn the affinities for each of the different parent classes (natural, null, nearest neighbour, etc) so we require a dataset which is poorly modelled by a fixed architecture tree. Section 3.3 investigates such issues and we choose parameters for the generative model where the gap between *dynamic tree* and best fixed tree is large. This gives a generative model whose nearest neighbour affinity is 0.25 and null parent affinity -2.25. These are relative to the natural parent whose affinity is held constant at zero. β – which provides a lever to "squeeze" the affinity profile – is set to 1.0 so that the affinities quoted are not scaled. The CPTs were set to 0.99 down the diagonal and the state prior was uniform.

The $1-2-4$ binary node configuration was chosen once again as this is at the very limit of tractability (within a reasonable time frame) for the exact method against which mean field is compared. This arrangement has 8 leaf nodes and for binary pixel intensities gives a total of 16 images. For the full *dynamic tree* model this node arrangement allows a choice of two parents (including the null parent) for each of the two nodes in the second layer, and the leaves each have a choice of three parents. This produces a total of 324 tree structures.

Naive gradient ascent learning of the affinities was tried alongside the more sophisticated scaled conjugate gradient (SCG) optimisation. While SCG optimisation predictably converged much faster (typically 40-50 times faster) the quality of solutions it achieved were no better than straightforward gradient ascent. The results shown in the following Section are from gradient ascent optimisation runs as this illustrates more clearly what is happening.

5.5.4.1 Gradient ascent optimisation

The procedure involves fully marginalising out the hidden $\mathbf{X}_h$, and connection variables $\mathbf{Z}$, from the joint distribution $P(\mathbf{X}_v, \mathbf{X}_h, \mathbf{Z})$ given by the generative model to produce a probability distribution over images. This distribution is then supplied to the learning algorithm. Once again we weight each of the 16 images by their probability of occurrence and so have an effectively infinite training set.

Gradient ascent requires a learning rate term η, which is ordinarily a constant less than one. A small amount of experimentation was done, but in practice it was found that it only affected the rate of convergence and not the final solution. An η of 0.3 was used in the following runs.

Initially runs were made with starting nearest neighbour and null parent affinities of $+1$, and then a second run was performed with them at 0.5 and -2 respectively. These represent starting perturbed a long way from the generative model and beginning very close to it, and between them were designed to examine the stability of the learning algorithm and the radius of any basin of attraction which one hopes would exist. The traces of the affinities and log-likelihoods for both runs are shown in Figure 5.8.

From the affinity trace plots in Figures 5.8(a) and 5.8(c) it can be seen that the exact approach (shown as a continuous line) using the true posterior distribution does learn the target parameters of the generative model (dashed line). Its log-likelihood is seen to increase rapidly initially, and then remain constant at -1.57. This would suggest that the log-likelihood surface around the target model is almost flat, and a number of different solutions perform almost as well.

For the mean field runs the null parent affinity converges to around -2.55 in both cases and this is interesting as it seems to show that there is a stable fixed point for the null parent affinity under the mean field approximation at this point. The fact that it finds this solution even when perturbed a long way away from it shows it has a large basin of attraction.

The nearest neighbour affinities appear to perform very well under mean field. Typically from the runs it settles on a solution of between 0.22 and 0.07 which compares

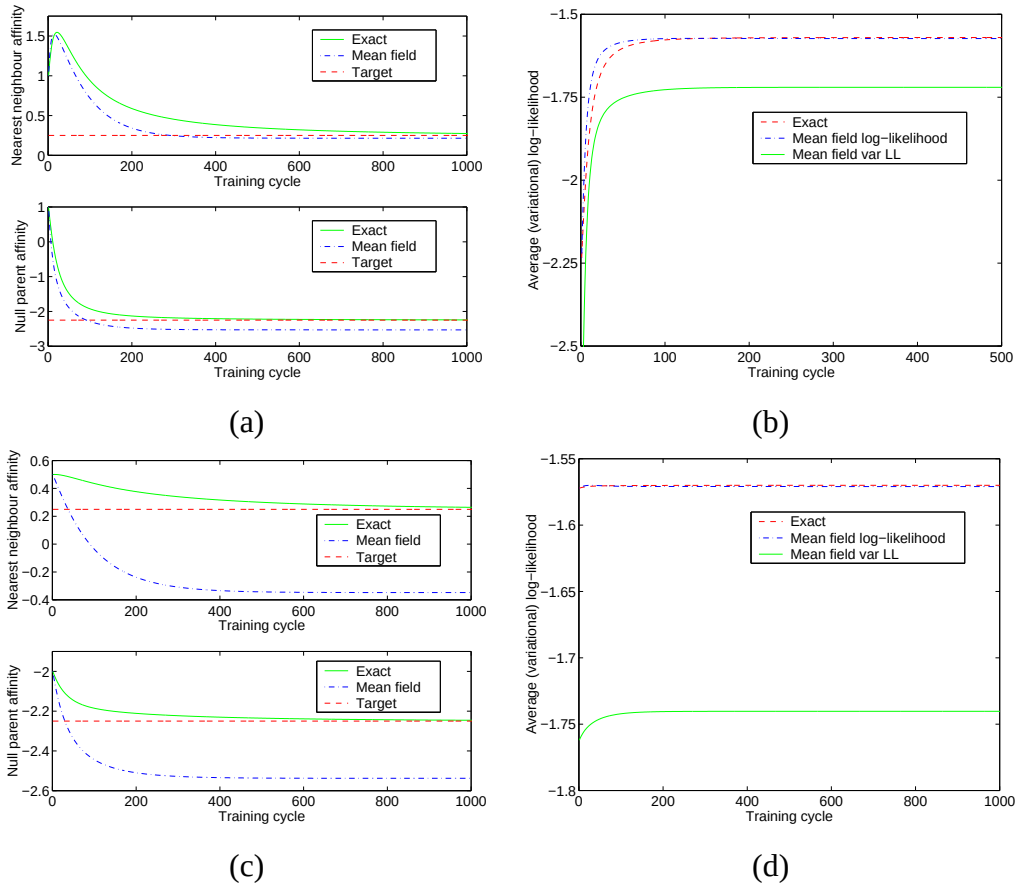


Figure 5.8: Affinity traces and log-likelihoods starting (a)–(b) perturbed far from, and (c)–(d) close to the generative model, with the initial affinities all larger than the target ones.

favourably to the target of 0.25. Clearly then the nearest neighbour concept is approximated satisfactorily by mean field, and it is the case that setting it too small initially will cause it to be suppressed. It was noticed that when the nearest neighbour affinity was initialised at values less than 0.5 very often it would be suppressed. Care then needs to be taken over starting positions for learning runs, and setting initial affinities higher rather than lower is the safer option.

For these runs the converged log-likelihood for the mean field approach was -1.5709 this is extremely close to the correct log-likelihood found by the exact approach, yet the affinity values are clearly different. This illustrates the flatness of the likelihood surface immediately about the optimal solution where many parameterisations give very similar likelihoods. However this could just be a facet of the small models used

which tend not to have the variability of larger more complicated models and we would not necessarily expect this to be the case for bigger models applied to real data.

Changes in null parent affinity happen far more quickly during learning than for the nearest neighbour. This may in part be due to the fact that the null parent affinity is shared between both of the lower two layers so is given more emphasis, whereas the nearest neighbour affinities operate solely in the bottom level connections.

From the likelihood plots though it appears that when the null parent affinities change rapidly so does the average log-likelihood of the model. In contrast the nearest neighbour affinity changes produce a barely perceptible effect. This suggests that in models of this size the null parent plays a dominant role.

The previous runs were all undertaken with starting affinities greater than the generative model. To test whether this is important similar runs were performed to the above only perturbing the affinities of the nearest neighbour and null parent lower than the generative model. Starting affinities of -4 for both the nearest neighbour and null affinities was used as a far perturbation from the generative model, and a nearest neighbour affinity of 0 and null affinity of -2.5 was a close perturbation. Figure 5.9 shows the affinity traces and corresponding log-likelihood plots for these runs.

Once again the exact approach learns the generative model correctly, though it takes considerably longer. Figures 5.9(b) and 5.9(d) show that the variational log-likelihood increases rapidly in the initial training cycles, but then remains virtually flat from then on. For the perturbed far from the generative model case (Figure5.9(b)) the variational log-likelihood is seen to increase in two steps. Interestingly this occurs at points of rapid change in the null parent affinity.

Mean field performs disappointingly from these starting positions as the dot-dashed line in the affinity trace plots shows. We see that the best it achieved for the null parent affinity was -2.7 and for the nearest neighbour it diverged from the target. The first interesting thing to note is whether perturbed close to the generative model or a long way away mean field still learned the same affinity values for both parent types. This is very encouraging as it suggests that the learning algorithm is stable, however since mean field is only an approximation to the true posterior it is not unlikely that

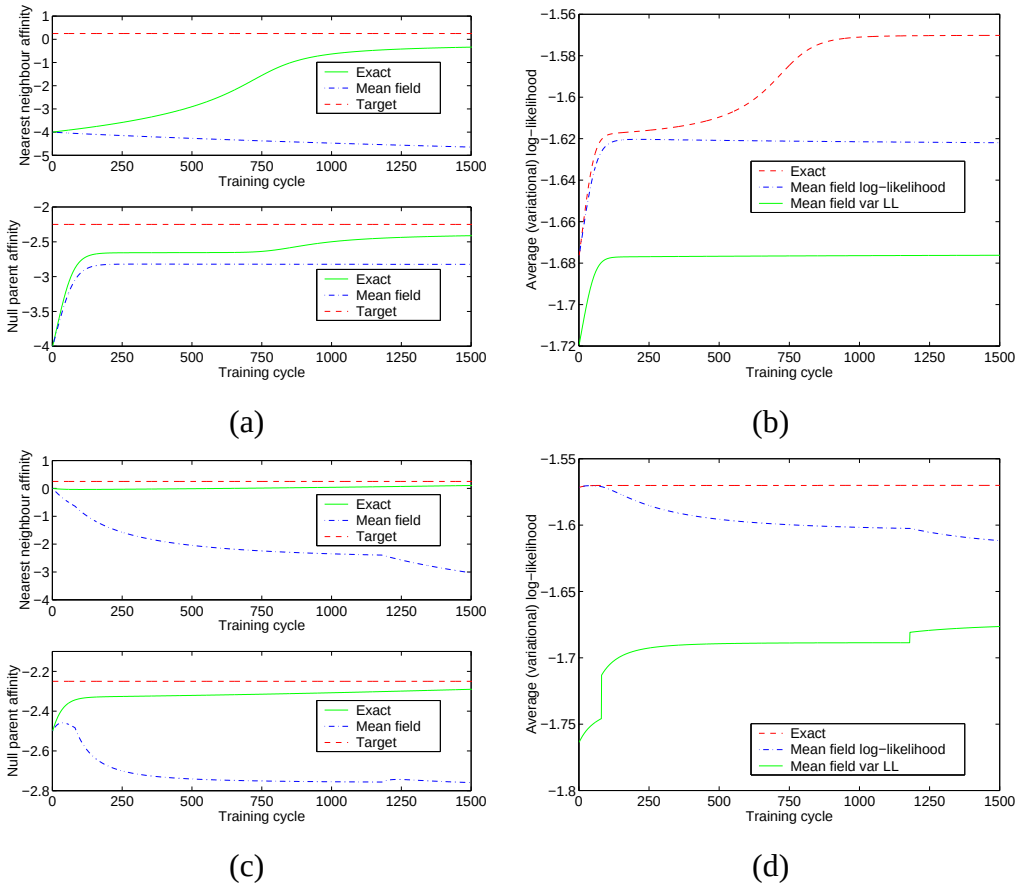


Figure 5.9: Affinity traces and log-likelihoods starting (a)–(b) perturbed far from, and (c)–(d) close to the generative model, with the initial affinities all smaller than the target ones.

it will have slightly different optima. By initialising the affinities to very small values in comparison to the natural parent which is held constant at zero we are in reality making it very difficult for them to assert any influence at all. Similar behaviour is well documented for neural networks where if the weights are initialised too small they frequently are suppressed and set to zero. It could well be the case that something similar is happening with the nearest neighbour affinity as the more negative an affinity becomes the smaller is its connection probability.

In an attempt to improve the mean field results a number of things have been tried. This included initialising successive mean field cycles with the optimised solution found in the previous training cycle for that data point. This was found to eliminate spiking in the variational log-likelihood where occasionally it gets stuck in lower valued optima,

but did not improve on the final solution. Multiple (5) runs were performed for each starting point and the results shown were typical of those produced.

Overall mean field was found to successfully learn solutions for *null* and *nearest neighbour* affinities comparable to the generative model, though care needs to be taken not to initialise them too low.

5.6 Discussion

We have seen that an EM style learning algorithm based upon mean field performs encouragingly in a comparison with exact EM in fixed architecture trees, and shows good potential for use in larger structures where exact EM becomes intractable. Using small tractable models has enabled us to make a thorough comparison between the two approaches and given valuable insights into the capabilities of mean field EM learning invaluable for future work. Spontaneous symmetry breaking was seen to be a weakness of mean field which can affect learning, but with careful monitoring of training error this can be avoided.

In the disconnecting tree mean field EM finds the generative model parameters at the level nearest the data, but higher levels become degenerate. It means that mean field has collapsed down the hierarchical model into a single layer as it can still obtain good log-likelihoods for the toy dataset considered. It was noted in Chapter 4 that mean field driven by its factorised approximation has a tendency to do this, but for more complex datasets it made use of higher levels.

For learning of the prior $P(\mathbf{Z}|\boldsymbol{\phi})$ the affinity approach seems very attractive, as it allows parameters to be readily shared using templates. In the comparisons the exact approach was able to fully learn the generative model, and when initialised at the generative model did not deviate. The mean field approximation had more difficulty though it still found solutions which increased the variational log-likelihood. On the toy data it appears to be much harder to learn the nearest neighbour probabilities than disconnections, and an important consideration on initialising the model is that they are not set too low. Otherwise they are suppressed to zero from which they cannot recover.

The same issues regarding the mean field approximation for the CPTs is also applicable to the affinities, and though predictably the simpler mean field approximation is inferior to using the true posterior it is tractable and has been seen to produce satisfactory results.

This completes the theory for DT learning, and in the next Chapter the *dynamic tree* model is applied to a real image data task and its performance evaluated.

Chapter 6

Learning on Real Images

6.1 Introduction

So far we have introduced the *dynamic tree* model and in subsequent Chapters developed and evaluated efficient inference and learning algorithms for the model. In this final experimental Chapter we tie all these things together with an application on real data.

The ultimate test of any model's capabilities is how it performs on real world data, and here we apply the DT to a database of outdoor scenes.

In Williams and Feng (1999) the usefulness of Tree-Structured Belief Network for coding images is explored, and it would be interesting to examine whether the *dynamic tree* can further extend the advantages noted therein. This is also discussed in this Chapter.

Section 6.2 discusses practical issues related to scaling up to real-world sized problems, Section 6.3 summarises the exact and mean field EM learning algorithms used for learning the *dynamic tree* parameters (see Chapter 5 for a full derivation), and experiments comparing the *dynamic tree* to a fixed architecture model showing it offers a significant improvement are given in Section 6.4.

6.2 Scaling up to Real Images

There are a number of issues associated with scaling up to using real images, including choosing a suitable programming language, balancing speed against memory efficiency and taking precautions to protect against numerical errors – an issue because of the finite arithmetic range of computers. These are discussed below.

6.2.1 Choosing a Programming Language

The choice of programming language is an important consideration for any learning algorithm. Hitherto Matlab 5.1 has been primarily used as it had the useful properties of being rapid to prototype with its powerful matrix formalism, and easy to test from within the command line driven environment of the application.

This provided the ideal environment from which to develop and test the initial learning algorithms on toy data and proved invaluable for gaining insights into the effectiveness of mean field for learning in the *dynamic tree* in Chapter 5. However the flexibility it gives is at the cost of speed, and for scaling up to real-world problems something more powerful is required.

Obvious candidates are Fortran or C for speed, but the object-orientedness of Java allows for a more structured programming approach which is desirable when implementing complex models such as the *dynamic tree*. C++ was finally chosen as it provides a good balance between all of these desirable qualities.

The full source code of the C++ implementation written for this work and the suite of Matlab functions which complement it are fully documented and available at **`http://www.anc.ed.ac.uk/code/adams/dt/dt.tgz`**. Here only a brief outline is given to illustrate some of the practical issues associated with moving from small toy problems to large dimensional real-world data.

The *dynamic tree* model is composed of three main classes. The `Node` object is the smallest and contains all variables pertinent to a single node in the tree, such as whether it is instantiated with data (leaf node) and pointers to any probability tables it may require.

`Tree` is the base class of the *dynamic tree* grouping the nodes and describing their connectivity. It implements all of the crucial functionality such as calculating the likelihood of the model given data.

Derived from class `Tree` are various `Prior` classes each of which add the necessary functionality for performing operations on the given prior type. A host of different priors have been considered in this work on the *dynamic tree* of which the product `Prior_p1` and conditional `Prior_c4` implement the *part* and *full-time node-employment-priors* described in Chapter 3. Figure 6.1 shows this class hierarchy in full.

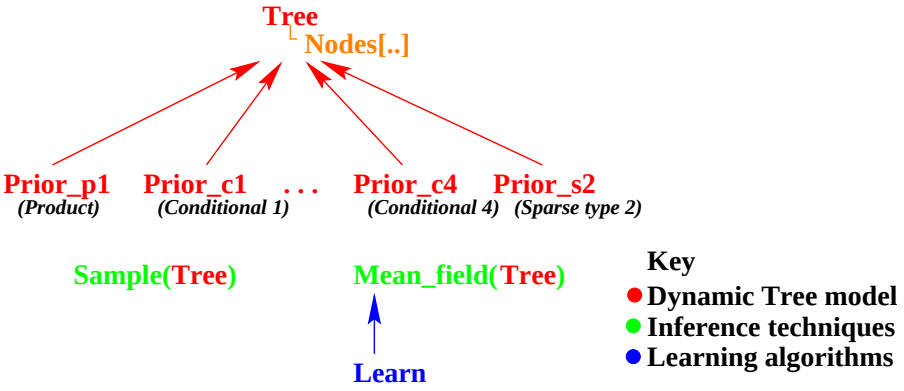


Figure 6.1: The *dynamic tree* model C++ class hierarchy.

Thus the `Tree` class accessed through the desired derived `Prior` class produces the full *dynamic tree* model. To conduct inference further classes are defined which take the DT model as an argument. These are `Sample` for the sample and search type approaches including Gibbs Sampling and Simulated Annealing. Class `Mean_field` conducts the alternative variational method of inference, mean field, as described in Chapter 4.

The highest level class `Learn` is derived from the inference classes whose methods it extensively uses for learning each of the various DT parameters.

This then is the basic structure of the *dynamic tree* code. The advantages of using an object oriented language for such a complex task have been considerable especially in areas of flexibility and reliability.

In implementing the model a number of practical issues arose. These commonly occur

and are well-documented arising from the fact that a computer is a finite machine with finite memory and finite arithmetic. The key issues encountered were

- Speed versus memory efficiency
- Scaling arrays to avoid numerical errors
- Handling missing data

The handling missing data is best described with reference to the learning algorithm and so will be discussed after the theory in Section 6.3.3. The other two are discussed below in the following Sections.

6.2.2 Speed Versus Memory Efficiency

Learning in large models such as required for real image data is computationally very intensive and ideally we would wish to pay as small an overhead as possible in accessing data and variables. The optimum would be working on full matrices with immediate access to each element as and when required, but this is not always possible.

For the *dynamic tree* model used on the 96×64 pixel images the number of leaf nodes is 6144. Add to this the hierarchy of nodes in the layers above arranged in a quad-tree formation (except for the second level where there are 6 nodes instead of the usual 4), then the total number of nodes becomes 8191. This presents a problem in specifying the distributions $\pi(\mathbf{Z})$ and $Q(\mathbf{Z})$ between each node and all of the others as to fully represent them $O(10^8)$ elements are required for each. Assuming that each element is a *float* of 8 bytes in size, then the total space required per matrix is $\sim$0.8 GB. Clearly this is impractical.

Fortunately the majority of elements will be zero as most nodes can pick only one of a handful of parents, and this suggests the solution of implementing the connectivity matrix as a sparse matrix where only the non-zero elements are stored. However there

is a cost in access time as zero elements can only be retrieved by firstly ascertaining that they are not among the list of non-zero elements.

It is therefore vital that that non-zero elements are organised as efficiently as possible so that as few search steps as possible are required to find any element or ascertain that it is zero. Hashing functions are used frequently in computers to solve problems such as these. The connectivity matrix for *dynamic trees* have a particular property of having only a few elements in each row (each node can pick only one of a handful of parents) and they further tend to be grouped together as the indexes of the parents a node can choose commonly lie consecutively (neighbouring nodes in the layer above).

This suggests the following mapping could efficiently represent the $\pi(\mathbf{Z})$ and $Q(\mathbf{Z})$ matrices sparsely.

$$\text{matrix}[m][n] \quad \rightarrow \quad \text{sparse\_matrix}[m][n \bmod K]$$

where m and n are the row and column indices of the matrix, and K is an integer. This effectively splits the matrix into smaller matrices of width K. These are then overlaid and any clashing elements are chained on a linked list. The resultant superposition of matrices is then stored sparsely.

For this task $K = 13$ was used to strike a balance between ensuring that very few non-sparse elements occupied the same bin (so needing to be chained as a linked list) while not wasting too much memory allocating space which is not used. Access times achieved were typically 2–2.5 times slower than the full matrix representation.

For further details consult the C++ code of the `SparMat` class of the *dynamic tree* source code available on-line.

6.2.3 Scaling Arrays to Avoid Numerical Errors

In working with probabilities – especially in large models – there is always the danger that some may underflow to zero which can result in transition probability updates for some states being erroneously recalculated as having zero probability when in reality they have a small but significant value.

For the Pearl message passing algorithm for exact inference this was a particular problem with the λ state of the top level root node frequently being set to $(0,\ldots,0)$. (See Appendix A, and Pearl (1988a) for details of the algorithm.)

Commonly this can be solved by rescaling the vector by dividing through by a constant which is retained and recombined separately. This constant is best kept within the log domain as it may become too small to represent within the standard range of the computer and if the vector elements are also converted to logs at a later date the constant becomes additive.

To avoid underflow a good choice for the constant is the smallest element of the vector. This proved very effective initially, however later on in training the *dynamic tree* the difference between the highest and lowest elements in the λ vector grew to such an extent as to cause overflow. Under such circumstances the smallest element is of no significance as the belief states represented by the elements with high values dominate, so instead normalising with respect to the highest valued element in a λ vector both prevents it underflowing to $(0,\ldots,0)$ and avoids the numerical overflow which is catastrophic for the learning algorithm. This then was the approach which was adopted.

Given that these practical issues related to performing learning in realistic models within the limitations of current computer technology have been suitably resolved we can now turn our attention to learning the *dynamic tree* model parameters on a database of real world images. This is the subject of the rest of this Chapter, where in Section 6.3 the theory is briefly summarised, before experimental procedure, results and analysis are given in Sections 6.4–6.5.

6.3 Learning in Dynamic Trees

Before progressing to the experiments we briefly summarise the learning rules for *dynamic trees* derived in the previous Chapter. Section 6.3.1 gives the exact EM learning rule for the CPTs, and then in Section 6.3.2 the full mean field updates for all *dynamic tree* parameters are summarised.

6.3.1 An EM update for learning the CPTs

The *dynamic tree* model is made up of two components. A prior $P(\mathbf{Z}|\boldsymbol{\phi})$ defines a probability distribution over tree structures $\mathbf{Z}$ and is conditional on a set of parameters $\boldsymbol{\phi}$, which are used in its construction and to be learned during training. Then there are state variables for the nodes with CPTs $\boldsymbol{\theta}$ governing the transition probabilities between connected nodes.

To assign each parent-child combination its own unique CPT however would lead to massive over-parameterisation for the limited training data usually available, so it was deemed sensible to share the CPTs among nodes on the same level (scale). θ_I is used to denote the shared CPT for the set of nodes $\mathbf{X}_I$.

The derivation of a a learning rule for the CPTs involves maximising the log-likelihood of a set of training images $\mathbf{X}_v^n, n = 1 \ldots N$, which produces the following EM update

$$\hat{\theta}_{Ij}^{kl} = \frac{\sum_{n,\mathbf{Z}^n} \sum_{x_i \in \mathbf{X}_I} P(x_i^{k(n)}, x_j^{l(n)} | \mathbf{X}_v^n, \mathbf{Z}^n, \boldsymbol{\theta}) P(\mathbf{Z}^n | \mathbf{X}_v^n, \boldsymbol{\phi})}{\sum_{n,\mathbf{Z}^n} \sum_{x_i \in \mathbf{X}_I} \sum_{k'} P(x_i^{k'(n)}, x_j^{l(n)} | \mathbf{X}_v^n, \mathbf{Z}^n, \boldsymbol{\theta}) P(\mathbf{Z}^n | \mathbf{X}_v^n, \boldsymbol{\phi})} \tag{6.1}$$

where

$$P(x_i^k, x_j^l | \mathbf{X}_v^n, \mathbf{Z}^n, \boldsymbol{\theta}) = \frac{1}{\sum_{l'} \pi(x_j^{l'}) \lambda(x_j^{l'})} \lambda(x_i^k | \boldsymbol{\theta}) \theta_{Ij}^{kl} \pi(x_j^l | \boldsymbol{\theta}) \prod_{y \in s(x_i)} \lambda_y(x_j^l | \boldsymbol{\theta}) \tag{6.2}$$

The λs and πs are the Pearl messages used to pass information to a node about the states of its children and parents respectively Pearl (1988a), and $s(x_i)$ is the set of siblings of node i. This derivation is an extension of that of Feng and Williams (1998) used for fixed architecture TSBNs, full details of which were given in Section 5.3.1.

6.3.2 Mean Field EM in Dynamic Trees

In the mean field *dynamic tree* (Chapter 4) the true posterior distribution $P(\mathbf{X}_h, \mathbf{Z} | \mathbf{X}_v, \boldsymbol{\theta}, \boldsymbol{\phi})$ is approximated by a factorising distribution, $Q(\mathbf{X}_h, \mathbf{Z} | \mathbf{X}_v) = Q(\mathbf{X}_h) Q(\mathbf{Z})$. This can be used to find a lower bound on the log-likelihood of the data

$$\sum_n \log P(\mathbf{X}_v^n) \geq \sum_{n,\mathbf{X}_h^n,\mathbf{Z}^n} Q(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n) \log \frac{P(\mathbf{X}_v^n,\mathbf{X}_h^n,\mathbf{Z}^n|\boldsymbol{\theta},\boldsymbol{\phi})}{Q(\mathbf{X}_h^n,\mathbf{Z}^n|\mathbf{X}_v^n)} \stackrel{def}{=} \mathcal{L}(Q,\boldsymbol{\theta},\boldsymbol{\phi}) \tag{6.3}$$

which can be similarly used to derive an EM style learning algorithm based upon mean field (Section 5.2.2). The CPTs $\boldsymbol{\theta}$ and the affinities $\boldsymbol{\phi}$ constitute two very distinct parameter types and as such require different treatment.

Learning the CPTs: The derivation uses a similar methodology to that of exact EM (see Section 5.3.2 for full details). Performing this optimisation on the DT this gives rise to the following update rule for the CPTs

$$\hat{\theta}_{Ij}^{kl} = \frac{\sum_{n,\mathbf{Z}^{(n)}} \sum_{X_i \in \mathbf{X}_I} Q(X_i^{k(n)}) Q(X_j^{l(n)}) Q(\mathbf{Z}^{(n)}) z_{ij}^n}{\sum_{k'} \sum_{n,\mathbf{Z}^{(n)}} \sum_{X_i \in \mathbf{X}_I} Q(X_i^{k'(n)}) Q(X_j^{l(n)}) Q(\mathbf{Z}^{(n)}) z_{ij}^n} \tag{6.4}$$

Learning the Affinities: The affinities set a prior over tree structures. As for the CPTs affinities also can be shared between nodes to reduce parameterisation. We define $\boldsymbol{\phi}$ to denote such sets of shared affinities, and a_{ij} is the individual *affinity* a node i has for connecting to parent j.

To obtain an update we maximise the bound on the log-likelihood (Equation (6.3)) with respect to the affinities by differentiating Equation (6.3) with respect to the shared affinities to produce

$$\frac{\partial \mathcal{L}(Q,\boldsymbol{\theta},\boldsymbol{\phi})}{\partial \phi} = \sum_n \sum_{i,j,\{i,j\}\in\phi} \left[Q(z_{ij}|\mathbf{X}_v^n,\boldsymbol{\phi}) - \sum_{\mathbf{Z}^n} Q(\mathbf{Z}^n|\mathbf{X}_v^n,\boldsymbol{\phi}) \pi_{ij} \right]. \tag{6.5}$$

This can be optimised wrt $\boldsymbol{\phi}$ using standard gradient optimisation techniques such as conjugate gradients. (See Section 5.5.2 for the complete derivation.)

The Complete Learning Algorithm therefore consists of fixing the model parameters θ and ϕ, and running mean field until the Qs reach convergence.

Then we fix the Qs and calculate the update for the CPT using Equation (6.4). A gradient based optimiser can then be used on Equation (6.5) to traverse one or more steps along the affinity gradient.

The θ and ϕ parameters are then updated, and the process repeated.

6.3.3 Handling Missing Data

A reality of dealing with real world is that frequently we are given examples which are incomplete, broaching the issue of how do we deal with missing data? If we simply set the missing pixel to a random value we may well end up learning noise, and even a uniform instantiation with all states equi-probable could still be saying something which is untrue.

Assuming that the data is missing at random (see Little and Rubin (1987)) the correct solution is to marginalize out the uninstantiated variables. For exact inference using Pearl message passing in belief networks this is achieved automatically by the algorithm when the λ value of the uninstantiated node is set to $(1,1,\ldots,1)$. With the mean field approximation a little care is required as it is unclear as to whether an uninstantiated leaf node may exert any unintended influence on the resultant equilibrium distribution. The solution we have used is to temporarily modify the connection probability table so that missing data leaf nodes have a probability of disconnection of 1.0. They then do not contribute anything towards the mean field equilibrium distribution. The implications for the learning rule update equations is that such nodes should be ignored for the given example.

An alternative to marginalising out unlabelled pixels would be to model them as a class in their own right. This is entirely possible within the framework of the current algorithm and can be justified within a principled approach. However, as unlabelled pixels are only an artifact of the labelling scheme and are not a property of the distribution that underlies the images which we are trying to model we prefer the solution above.

6.4 Experiments

This Section comprises of two parts. Section 6.4.1 discusses the origins of data used in the experiments and how it was presented to the *dynamic tree* model. Then in Section 6.4.2 the experiments are performed and analysed.

6.4.1 The Image Data

We use a set of colour images of out-door scenes from the Sowerby Image database[1] for our experiments. These feature both rural and urban examples and contain many of the typical objects you would expect to find – such as the roads, cars and buildings of urban environments to the country lanes and fields of the rural. The original scenes were photographed using small-grain 35mm transparency film under carefully controlled conditions at a number of locations in and around Bristol, UK. The analogue transparencies were then digitised using a calibrated scanner to produce high quality 24-bit colour representations.

In addition to the raw images the database also contains corresponding labelled examples created by over-segmenting the images and then hand labelling each region produced. This gave rise to 92 labels, organised hierarchically.

The fixed TSBN model has already been applied to this database (Feng *et al.*, 2001) in which the 92 class labels are merged down to 7 super classes. Since the fixed TSBN can be viewed as a special case of the *dynamic tree* where the architecture is not allowed to change this provides an excellent model for comparison and consequently we chose to adopt the same class labels.

The labels distinguish all of the key regions of interest in the image representing, “sky”, “vegetation”, “road markings”, “road surface”, “building”, “street furniture” and “mobile object.” Such is the nature of gathering real data that circumstances inevitably

[1]This database can be made available to other researchers. Please contact Dr Andy Wright or Dr Gareth Rees, Advanced Information Processing Department, Advanced Technology Centre - Sowerby, BAE SYSTEMS Ltd, PO Box 5 Filton, Bristol, BS34 7QW, UK, email: `gareth-s.rees@baesystems.com` for details.

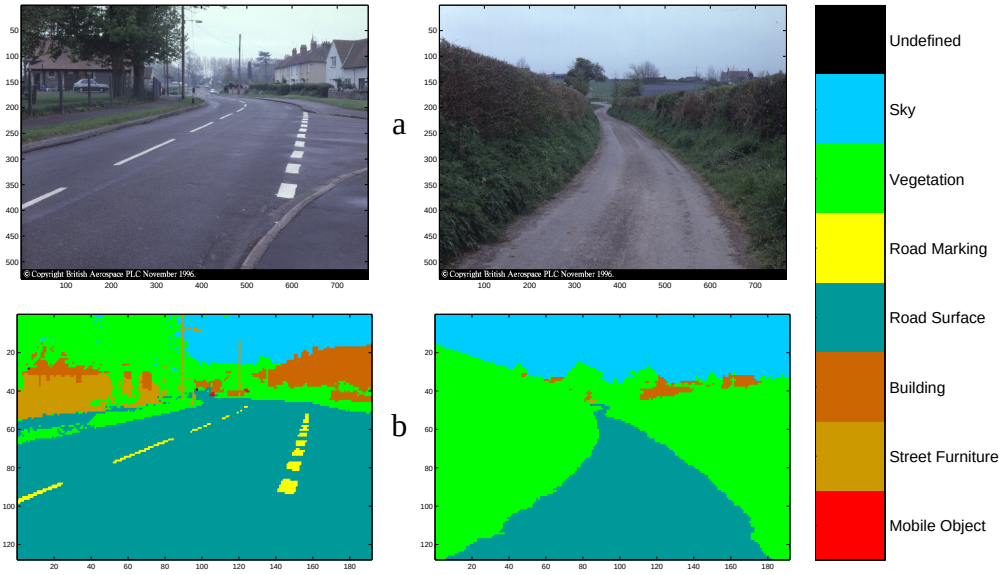


Figure 6.2: (a) An urban and a rural scene from the Sowerby database and (b) their corresponding labelled images. To the right is a key defining the labels used.

arise where there is missing data, so a further dummy "unlabelled" class is added to accommodate this. Unlike the others the unlabelled class is not learned since it is an artifact of the labelling strategy adopted and is dealt with as described in Section 6.3.3. Figure 6.2(a) shows one urban and one rural scene from the database.

The *dynamic tree* as formulated operates with discrete classes hence it is necessary to use the labelled images instead of the real valued data. However there are a wealth of established techniques which can be used to produce a mapping from real valued pixels to a discrete number of classes. For example in Feng and Williams (1998) a multi-layer perceptron (MLP) is trained to do this. It takes as its input an 18 dimensional feature vector comprising elements such as normalised red-green-blue, colour hue, contrast, texture and grey scale whose statistics are calculated directly from the images. There is also no reason why some discretised form of the raw feature vector itself could not be used in place of class labels for unsupervised learning, but the advantage of having class labels representing real world features is that it allows analysis which is directly interpretable. The reverse mapping from labels to real valued pixels is equally straightforward, eg. using Gaussian mixture models. The MLP and Gaussian mixture model methods are compared in Feng *et al.* (2001).

6.4.2 Experimental Procedure

The Sowerby database contains 104 images which were randomly divided by Feng and Williams (1998) into a training set of 61 images and the rest allocated as a test set. For comparative purposes we use the same training and test sets to learn the *dynamic tree* model parameters.

The full size images comprise of 768×512 pixels, which Feng *et al* (Feng *et al.*, 2001) reduce by a factor of 4 in each dimension to 192×128 pixels. This was achieved by sub-dividing the full size image into regions of 4×4 pixels and adopting a majority voting strategy to chose the winning class label. In cases where there was a tie a label was chosen probabilistically from the competing classes based on the prior probabilities of the given labels being seen in the images. The class label prior probabilities for the training and test sets are given in Table 6.1.

Class	**P(Label)**	
	Training Set	Test Set
Unlabelled	0.0036	0.0418
Atmospheric phenomena	0.1443	0.1140
Vegetation	0.3703	0.3899
Road surface markings	0.0012	0.0008
Road surface	0.4210	0.3804
Building	0.0473	0.0569
Street furniture	0.0056	0.0112
Mobile object	0.0067	0.0050

Table 6.1: Image pixel class priors for the training and test sets.

We adopt the same procedure and further downsample the images to 96×64 pixels. The label urban and rural examples of before are shown at this resolution in Figure 6.3 and it can be clearly seen that most of the detail is still present.

The downsampling process adopted ensures that the statistics of each class is maintained so there should be negligible loss incurred. Furthermore once trained the model

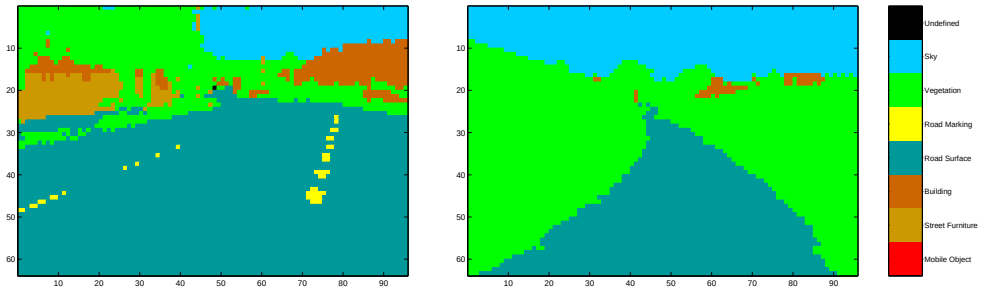


Figure 6.3: Examples of an urban and a rural scene label images down-sampled to 96×64 pixels.

could easily be extended to the full size image by using duplicates of the learned Conditional Probability Tables (CPTs) and prior for the lowest (leaf) level.

For the experiments we use a 7-level model. The node arrangement is quad-tree for all bar the second level where there are 6 instead of 4 nodes. The latter is to accommodate the $3:2$ aspect ratio of the training images and produces the desired image size of 96×64 pixels.

Setting the initial model parameters is an open question and probably a paper in itself. This issue has been explored in previous work (see Chapter 5, Feng *et al.* (2001)) and in keeping with the philosophy that a child node would favour being in the same state as its parent we make the CPTs strongly diagonal with probability 0.9 and the rest of the probability mass is shared equally among the 6 other states. The prior for root nodes being in a particular state was set to be uniform. All models were initialised with the above CPTs and state prior.

The *dynamic tree* model has a further set of parameters called affinities, which are used to set the prior over tree structures (Section 3.2.1.2). They are given relative to a parameter called the *natural parent* affinity which is the parent a given node would have if it were part of a balanced tree. This is set to 0 for reference purposes. Important other connections a node may wish to make is to its nearest neighbour(s) or to disconnect and become a root (*null* node). The raw affinity values are translated into probabilities using the following Equation

$$\pi_{ij} = \frac{\exp(a_{ij})}{\sum_{j'} \exp(a_{ij'})} \tag{6.6}$$

where z_{ij} is the boolean indicator variable of the connection between nodes i and j. In Section 3.3.2 investigation showed a useful working range for the *dynamic tree* was for affinity values of 0 and -3 for the nearest neighbour and null connections respectively. Other connections are given a probability of zero. The interpretation for this is that nearest neighbour connections are equi-probable with the natural parent and disconnections are possible but with a far lower probability.

Specifying each of the model parameters individually is unwise considering the limited amount of training data usually available. Here we chose to share the CPTs on a level-by-level basis and learn the affinities a_{nn} and a_{null} as single parameters over the whole *dynamic tree*.

We make a comparison between the fixed architecture balanced tree-structured belief network of Feng *et al.* (2001) and the *dynamic tree*. In the experiments on the DT the mean field EM learning algorithm described in Section 6.3.2 was used. Firstly mean field was run over all of the training examples to produce an approximation of the joint distribution for the E-step of the algorithm. This involved updating the means cyclically for a number of iterations until equilibrium was reached, then updating the structure $Q(\mathbf{Z})$ in single step and repeating until convergence. In practice the 5 complete iterations were sufficient but the algorithm was allowed to terminate early if the variational log-likelihood altered by less than 0.05 between cycles.

The M-step of the algorithm is a single step update for the CPTs, but for the affinities a gradient method is necessary. Conjugate gradients was used with the optimiser being allowed to take up to 3 steps. Three steps were necessary in order to take advantage of the conjugate gradients – performing only one would simply be gradient descent. After calculating new estimates for the CPTs and affinities all of the model parameters were updated and the process repeated.

A comparison was made between three types of model, the fixed quad-tree (fixed architecture), a *dynamic tree* where only the CPTs were learned (CPT-only DT) and the *dynamic tree* model where all parameters were learned (full DT). All used the mean field EM algorithm as summarised in Section 6.3.2 (and is described fully in Section 5.2.2), and additionally exact EM learning was performed on a fixed architecture

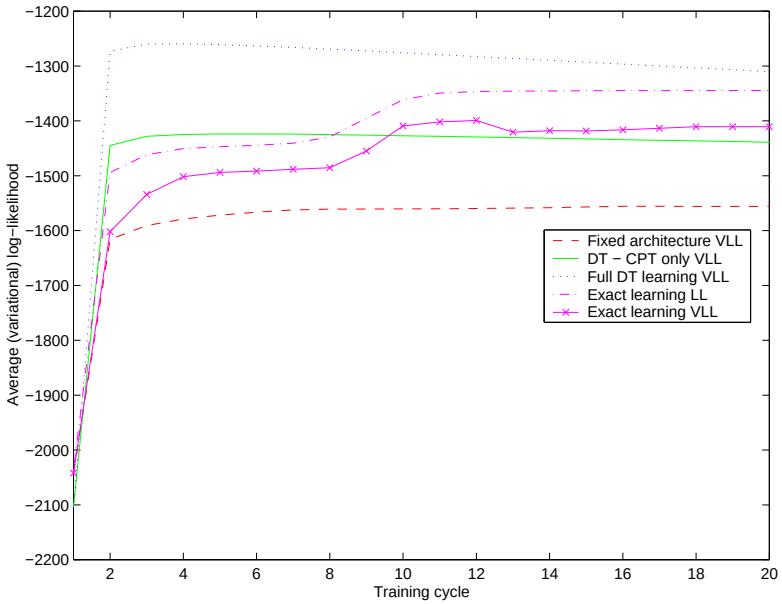


Figure 6.4: Learning curves on the 61 image training set for the fixed tree, CPT only learn *dynamic tree*, fully learned *dynamic tree* models, and exact EM learning on the fixed tree.

quad-tree model using the exact EM learning update given in Section 6.3.1. (For a further comparison of mean field EM against the exact method in fixed trees see Section 5.3.3.)

The learning curves on the training set of 61 patterns are given in Figure 6.4, showing the variational log-likelihoods obtained by the three DT models learned using the mean field EM approach, and the log-likelihood of the fixed architecture quad-tree model learned by exact EM.

Considering firstly mean field EM learning, it can be seen from the Figure that while the fixed architecture mean field EM model shows good improvement in variational log-likelihood during training it can only go so far. Learning only the CPTs of a *dynamic tree* model whose affinities were set from observed optimal parameters in toy data (see Section 3.3.2) appears to offer an advantage over the fixed architecture model with increased variational log-likelihood over the training set. The full DT model does better still, indicating that having variable architecture offers an advantage over the fixed architecture model.

Exact EM on the fixed architecture model can be seen to obtain a higher log-likelihood

than variational log-likelihood bound given by both the fixed and CPT-only mean field learning models. This is not surprising as the exact approach using the true probability distribution is able give the actual log-likelihood where-as mean field gives only a lower bound on the likelihood of the data $P(\mathbf{X}_v)$, and the comparisons made in Chapter 5 seemed to suggest that it may not be that tight. To confirm this on the real data the mean field variational log-likelihood was calculated during training at each iteration of the model learned under exact EM. This is shown as the crossed line in the plot of Figure 6.4 and we see clearly that the variational log-likelihood does indeed lie significantly below the exact log-likelihood. However, we observe that the variational log-likelihood obtained by mean field learning on the full *dynamic tree* beats the exact log-likelihood for EM training of the fixed architecture model, so clearly even though we can't perform exact learning on the *dynamic tree* model – which we would like to do – it still out-performs the fixed architecture model learned by exact approaches.

Though apparently fairly stable, it can be seen from the plot that the average variational log-likelihood for both variants of the DT appears to decline after 3–4 training iterations. A known weakness of the mean field learning algorithm (Section 5.3.3) is that it tends to settle at unstable equilibria where even a slight perturbation of the parameters can cause "spontaneous symmetry breaking." On the toy data learning of Chapter 5 it was observed that in subsequent iterations the model tries to correct this by hardening the CPTs resulting in a drop in performance, and this is probably what is happening here.

We can evaluate the quality of the learned models by calculating the variational log-likelihood on the test set of 43 images. This can then be used to obtain the coding cost in bits/pixel, where

$$\text{Coding cost} = -\frac{\log_2 P(\mathbf{X}_v^p)}{\#\text{ labelled pixels in image}} \tag{6.7}$$

An important point to remember is that with variational methods we are calculating a lower bound on the likelihood (an upper bound on coding cost), and the likelihood can only be at least as good or better.

Table 6.2 gives these coding costs for the various models and is compared with the

Model		**After Training Cycle**	**Bound on Coding Cost (bpp)**	
			Full	Less than 33% missing
Mean field	Fixed architecture	15	0.8588	0.3918
	DT - CPT only	2	0.4089	0.3228
	Full *dynamic tree*	2	0.3805	0.2942
Exact EM	Fixed architecture	10	0.3421	0.3253
JPEG-LS		–	0.3810	0.3782
Independent pixels		–	1.7872	1.7358

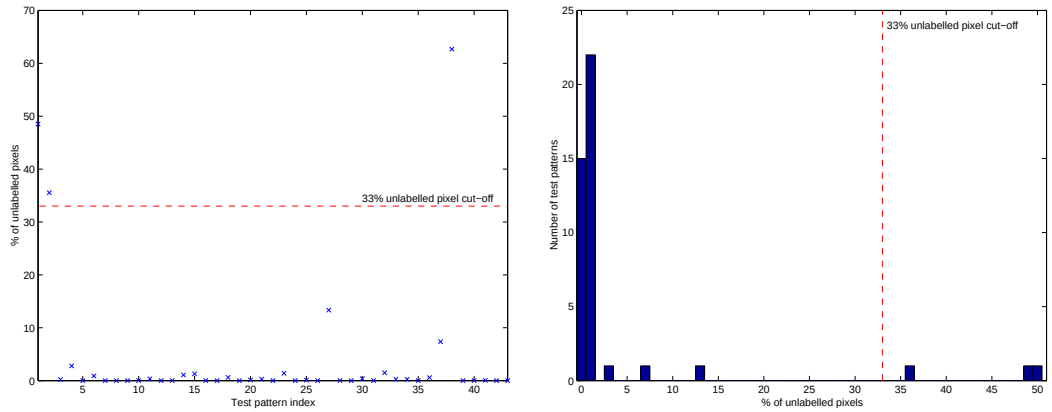
Table 6.2: Performance on the test set of 43 images. Independent pixels is the coding cost under the assumption that each pixel is independent of all of the others.

lossless JPEG-LS codec[2] – which is available from `http://www.hpl.hp.com/loco/`. Under the JPEG-LS scheme a model of the image is constructed by firstly conditioning the current sample on the previously observed ones. The resulting distribution is then used to assign shorter codes to more the probable events.

The second column of the table gives the number of training epochs used to train each of the models before applying them on the test data. In an attempt to minimise over-training this usually occurred at the point where the training error first peaked (see Figure 6.4). We see from the third column that on the full test set JPEG-LS outperforms all except the full DT and exact EM models.

However an examination of the percentage of unlabelled pixels in each of the images of the test set (Figure 6.5) shows 3 images to have greater than 33% unlabelled pixels which is extremely unusual. Ordinarily we might expect some unlabelled pixels and so need a robustness to them, but images degraded to that extent could reasonably be rejected as bad data. Removing these 3 images gives average coding costs as listed in the final column of the table. As can be seen now even the fixed tree is comparable to JPEG-LS and the full *dynamic tree* offers significant improvements over them both – the DT model was found to have a higher variational log-likelihood than the fixed architecture model in $42/43$ of the test images.

[2] In the case of JPEG-LS the coding cost was obtained by compressing the images and measuring the size of the compressed files.



(a) (b)

Figure 6.5: Percentage of unlabelled pixels in each of the 43 test set images (a) per image and, (b) as histogram.

Exact EM learning of the fixed architecture quad-tree tree also performs well. Over the full set of test images it achieves a lower coding cost than even the full *dynamic tree*. It has already been noted that the mean field approximation is very weak when there is little data so it is likely to perform badly on images with many missing pixels, and indeed we see this to be the case. (The variational log-likelihoods of the 3 images with more than 33% missing pixels are significantly lower than those of the other 40 images.) So it is not surprising that the fixed architecture model learned by exact EM achieves a slightly lower coding cost than the *dynamic tree* over the full test set of 43 images where there are many unlabelled pixels – especially as it has the advantage of using the true probability distribution for the model. However, after removing the 3 images with more than 33% missing pixels the *dynamic tree* model achieves a significantly lower coding cost (0.294 bpp) than the exact-EM trained quad-tree model (0.325 bpp). On this set of 40 test images, lossless JPEG obtains 0.378 bpp compression.

6.5 Discussion

In this Chapter we have considered learning the full set of parameters of the model *dynamic tree* and applied it to learning a database of outdoor scenes.

Using real world images of outdoor scenes the *dynamic tree* was seen to exhibit a higher variational log-likelihood during training than the log-likelihood of the fixed architecture model using exact EM. As mean field is only an approximation to the true distribution it is likely to produce a weaker bound than the actual log-likelihood obtained from the true posterior, so a higher likelihood is clearly indicative of the *dynamic tree*'s superiority over the fixed architecture model.

However, mean field is a bad approximation technique when there are significant amounts of unlabelled data in an image. On the full test set of 43 images the exact EM model achieves a lower coding cost than the full *dynamic tree*. When the three images with greater than 33% missing pixels are discarded though, the DT outperforms even the exact EM learned fixed architecture quad-tree model on the coding task, obtaining a theoretical 0.294 bpp compression against 0.325 bpp for the fixed architecture model learned under exact EM.

On a set of 40 previously unseen images with low levels of missing data (less than 33% unlabelled pixels in each image) from the same database the *dynamic tree* model again had a higher likelihood and calculation of the coding cost showed that the *dynamic tree* can achieve greater compression than the loss-less JPEG (JPEG-LS). In the comparison the full DT achieves 0.294 bpp compression compared to 0.378 bpp for lossless JPEG on images of 7 colours.

An alternative to discarding unlabelled pixels would be to model them as a class in their own right. This is entirely possible within the framework of the current algorithms and can be justified within a principled approach (JPEG-LS codes in this way). However as unlabelled pixels are only an artifact of the labelling scheme and not the distribution that underlies the images which we are trying to model we prefer to consider the unlabelled class as noise.

Chapter 7

Conclusions

The focus of this work has been on developing a novel hierarchical, probabilistic image model which we have called the *dynamic tree*. Successive Chapters have introduced the theory and developed inference techniques which are then used to underpin learning algorithms. Finally the full model was evaluated on a real world dataset of outdoor scenes. From this work the following conclusions can be drawn.

Chapter 3: The *dynamic tree* model

- The *dynamic tree* model overcomes the blocky segmentation problem of TSBNs and that it has a greater translation invariance than balanced TSBNs. Its dynamic architecture enables the creation of structures which we have shown better explain the image data under consideration. (Section 3.2)
- Generating from the *dynamic tree* produced 2-d images which did not exhibit the same blockiness inherent in fixed architecture TSBNs. Since the generative capabilities of a model are usually a good indicator of its inferential ability this suggests the DT offers a better representation of real-world images than fixed architecture TSBNs. (Section 3.2)
- A more rigorous comparison of DTs against fixed architecture TSBNs clearly shows that even in a small set of toy images there are many instances where even

choosing an optimal fixed architecture from the whole set of possible fixed architecture configurations is inferior to the full *dynamic tree* model. (Section 3.3)

- Simulated annealing methods were seen to be successful at finding trees that have high posterior probability. Though annealing is an effective means of finding good solutions from a distribution we cannot fully enumerate, simulated annealing (as with sampling techniques) is computationally costly and probably too slow for use in a practical system. (Section 3.5)

- An extension to the DT prior, the *sparse dynamic tree* was seen to be an important first step towards further enhancing the *dynamic tree* model. It addressed the fact that there are occasions where higher up in the *dynamic tree* there are sometimes too few nodes to build the structures that the model really wants to by extra adding nodes which are switchable. It maintained all the advantages of the DT and achieved an even flatter profile in the translation experiments, but the results suggest that there is clearly much more work which could be done. Further discussion as to possible improvements is given in the next Section. (Section 3.4)

Chapter 4: Mean field inference

- The simplest of the variational methods, mean field, has been considered and both qualitative and quantitative comparisons of its performance has been made against simulated annealing, using structural and KL divergence plots respectively. They illustrate that mean field was able to find good highest probability mean field solutions that rivalled the MAP structures found by simulated annealing. (Section 4.3)

- The application of image data to the leaves of the network constrains the states of nodes in the immediate layers above, but as we ascend the hierarchy it exerts a diminishing influence and in the upper levels of the network the equilibrium states inherent in the model can dominate the mean field solution. Thus the nodes further away from the image still tend towards the same state for all the nodes. This is undesirable as one colour will tend to dominate at the higher levels and

suppress the others, and this was evident in the types of tree structures seen. It resulted in subtrees being compressed and less use was made of the hierarchical node arrangement than in corresponding MAP models of the same images found by simulated annealing. Though this is a limitation, we do still see interesting structures in the mean field trees and they can still model variable sized regions as distinct trees of nearly appropriate size, though they capture them less precisely than methods which find the MAP configuration. (Section 4.3)

- The posterior of the highest probability mean field trees was usually lower than the MAP tree, but this is to be expected as in the sampling approaches such as simulated annealing we are setting out to find the MAP structure where-as for mean field we are concerned with averaging over the whole distribution. (Section 4.3)

- We can conclude that the mean field approach provides significant advantages over structure searching for the MAP solution in that it produces an approximating distribution to the posterior, which is more informative than simply choosing a single example. This was achieved with a considerable saving in computational effort and comes close to making real time inference in DTs viable. (Section 4.4)

- We note however, that the assumption in mean field of a factorised distribution over $P(\mathbf{X})$ is not necessarily a good one, and an important direction for future work would be to focus further on distributions giving a closer approximation to the true posterior. (Section 4.4)

Chapter 5: Developing learning algorithms

- An EM style learning algorithm based upon mean field performs encouragingly in a comparison with exact EM in fixed architecture trees, and shows good potential for use in larger structures where exact EM becomes intractable. Using small tractable models has enabled us to make a thorough comparison between the two approaches and given valuable insights into the capabilities of mean field EM learning invaluable for future work. Spontaneous symmetry breaking was

seen to be a weakness of mean field which can affect learning, but with careful monitoring of training error this can be avoided. (Section 5.3)

- For learning of the prior $P(\mathbf{Z}|\boldsymbol{\phi})$ the affinity approach seems very attractive, as it allows parameters to be readily shared using templates. In the comparisons the exact approach was able to fully learn the generative model, and when initialised at the generative model did not deviate. The mean field approximation had more difficulty though it still found solutions which increased the variational log-likelihood. On the toy data it appears to be much harder to learn the nearest neighbour probabilities than disconnections, and an important consideration on initialising the model is that they are not set too low. Otherwise they are suppressed to zero from which they cannot recover. The same issues regarding the mean field approximation for the CPTs is also applicable to the affinities, and though predictably the simpler mean field approximation is inferior to using the true posterior it is tractable and has been seen to produce satisfactory results. (Section 5.5)

Chapter 6: A real-world application

- On real world images of outdoor scenes the *dynamic tree* was seen to exhibit a higher variational log-likelihood during training than the log-likelihood of the fixed architecture model using exact EM. As mean field is only an approximation to the true distribution it is likely to produce a weaker bound than the actual log-likelihood obtained from the true posterior, so a higher likelihood is clearly indicative of the *dynamic tree*'s superiority over the fixed architecture model. (Section 6.4)

- However mean field is a bad approximation technique when there are significant amounts of unlabelled data in an image. On the full test set of 43 images the exact EM model achieves a lower coding cost than the full *dynamic tree*. When the three images with greater than 33% missing pixels are discarded though, the DT outperforms even the exact EM learned fixed architecture quad-tree model

on the coding task, obtaining a theoretical 0.294 bpp compression against 0.325 bpp for the fixed architecture model learned under exact EM. (Section 6.4)

- Using a set of 40 previously unseen images with low levels of missing data (less than 33% unlabelled pixels in each image) from the same database the *dynamic tree* model again had a higher likelihood and calculation of the coding cost showed that the *dynamic tree* can achieve greater compression than the loss-less JPEG (JPEG-LS). In the comparison the full DT achieves 0.294 bpp compression compared to 0.378 bpp for lossless JPEG on images of 7 colours. The fixed architecture model in contrast could only rival JPEG-LS. (Section 6.4)

Overall then the *dynamic tree* model which was motivated by fixed architecture TSBNs being prone to blocky segmentations is shown to be successful on real image data. It overcomes the blockiness of the fixed architecture TSBNs and from a coding perspective achieves lower coding costs than loss-less JPEG.

Such is the nature of research however that one question is usually answered by two or three more, and the final Section of this work will be devoted to some of these.

7.1 Opportunities for Future Investigation

The analysis in the previous Chapters have suggested the following key areas which appear very promising for future investigation

- Using real-valued nodes
- Further develop the *sparse dynamic tree* prior
- More complex variational approximations
- Other datasets

Each shall be discussed in turn in the following Sections.

7.1.1 Using Real-Valued Nodes

The introduction of additional information at the nodes could be used for example to add extra contextual information perhaps making explicit longer range dependencies which would otherwise have to filter implicitly through the model's architecture. This could be in the form of real-valued variables in addition to the usual multinomial variables of the *dynamic tree* model. These additional real-valued variables might be used to encode information such as that concerning the instantiation parameters of objects.

Currently objects are only picked out by root nodes in the *dynamic tree*'s forest of trees structure, but adding instantiation parameters would enable construction of higher level objects from identified sub-components in the image. Thus we might for example be able to identify a man from limb, torso and head components.

In a possible extension of the *dynamic tree* model to image sequences such parameters could potentially facilitate tracking of objects between frames.

Alternatively real-valued node variables could replace the current discrete ones completely to create a real-valued version of the *dynamic tree*. Gaussians would be an obvious choice of probability distribution to govern them due to their tractable properties. Such a model could then operate directly on real-valued pixel data and as it would allow textures to be generated datasets such as those from synthetic aperture radar (SAR) data – already very popular with authors such as De Bonet *et al* (De Bonet and Viola, 1998) in their wavelet models – might be considered.

A comparison the between the relative performance of the *dynamic tree* against these other models such of these would also be very interesting.

7.1.2 Further Develop the Sparse Dynamic Tree Prior

The *sparse dynamic tree* was introduced in Section 3.4 as an extension to the basic *dynamic tree* model. The motivation was to accommodate richer structures at higher levels by having switchable nodes which could be activated if doing so provided a higher probability interpretation of the image data, as the conventional *dynamic tree* sometimes had too few nodes at higher levels.

Though it did show an improvement over the *dynamic tree* with an even flatter posterior when compared against the DT in the translating 1-d bars test certain aspects of the prior were less than ideal.

The biggest problem was the mechanism for turning on nodes in the top layer. The SDT strategy was to treat them all as being independent and allow any node to turn on with a small finite probability. The probability of turning on n nodes was thus given by the Binomial distribution

$$P(n) = \binom{N}{n} P_{ac}^{n}(1-P_{ac})^{N-n} \tag{7.1}$$

where n is the number of nodes activated, P_{ac} the probability of node activation and N the total number of nodes in the layer.

The Binomial distribution has the mean $\mu = NP_{ac}$ so by tuning P_{ac} we can bias towards a particular number of nodes switching on. However there is no mechanism for determining which of the nodes are activated with ones on the periphery being just as likely as those in the central area.

The number of nodes activated at the top level has an important bearing on the number activated lower down. For nodes in the lower levels, apart from turning on as roots themselves with a similar small probability if a parent is too far away they will prefer not to connect and remain inactive. Subsequent levels will then have a similar difficulty. This can bias the prior in favour of turning on too few nodes and often the higher levels were unused and the hierarchical advantages of the model were curtailed.

Therefore rather than treating the activation of nodes at the top level as independent it would be better to take a more holistic approach and have some control on their position.

Choosing where to place the active nodes is not however straightforward. Perhaps a discrete Gaussian-like distribution centred about the middle node position on the top level could be used to decide which nodes to activate once we have decided how many will be turned on. This would then favour central nodes turning on over more distant ones which would desirably enforce a loose tree structure.

Perhaps this procedure could be used on lower levels with successive Gaussians having larger variances to give more and more uniform a spread. This would enable abstraction of the number of nodes activated from the issue of their connectivity, giving more control over the number of active nodes.

Such a scheme may well overcome the weakness of too few nodes in the *sparse dynamic tree* and would certainly be interesting to try.

7.1.3 More Complex Variational Approximations

Although mean field inference performed well it uses the assumption of a factorised distribution over $P(\mathbf{X})$ which is arguably a poor approximation to the true posterior.

Conveniently from the analysis the $\mathbf{Z}$ variables which describe the structure of the *dynamic tree* fall out as independent without need of further assumptions beyond the distribution being factorised between $\mathbf{Z}$ and $\mathbf{X}$. However the requirement for $\mathbf{X}$ to be factorised is incompatible with a hierarchical node arrangement.

An interesting extension to the standard mean field approach outlined in Chapter 4 might be to specify an alternative distribution over the x_i^k's that more realistically captures their relation. One possibility of using a tree structure to reflect their hierarchical dependence upon each other, is considered in (Storkey, 2000), but this is only one of a number of structured variational approximations which might be considered.

7.1.4 Other Datasets

There are other potential applications to which the *dynamic tree* model might be potentially profitably applied and it would be very interesting to try the model on some of them.

One such example might be the Landsat database of satellite images of the Earth used by Taylor and Henery (1994). There the class labels distinguish between soil types and various agricultural usages of different regions.

Appendix A

Calculating the Likelihood

For the sake of simplicity of notation we take a three level binary tree example to derive the likelihood before discussing how this is extended to more general tree structures.

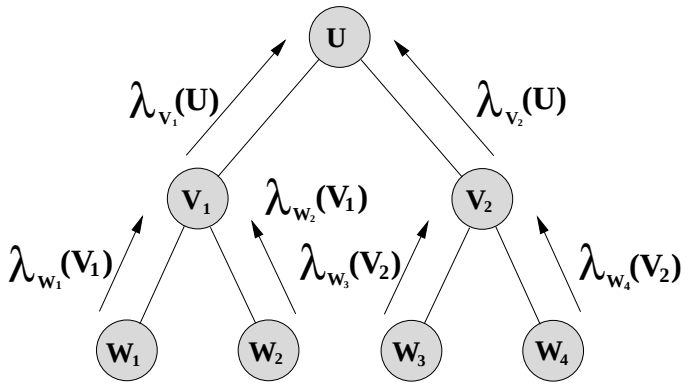


Figure A.1: 3-level binary tree.

Figure A.1 shows such a tree with a root node U and the leaf level is assumed to have the image vector $\mathbf{X}_v$ applied which sets the states of these nodes.

Pearl (1988b) defines the lambda vector $\boldsymbol{\lambda}(u)$ as follows

$$\boldsymbol{\lambda}(u) \quad = \quad P(\mathbf{e}^{-}|u) \tag{A.1}$$

where $\mathbf{e}^{-}$ is the evidence (instantiated nodes) in the subtree below u and shows that the

following two relations hold. Taking the node u in the Figure as an example, then

$$\boldsymbol{\lambda}(u) = \boldsymbol{\lambda}_{v_1}(u)\boldsymbol{\lambda}_{v_2}(u) \tag{A.2}$$

$$\boldsymbol{\lambda}_{v_1}(u) = \boldsymbol{\lambda}(v_1)\boldsymbol{\theta}^T_{v_1|u} \tag{A.3}$$

where $\boldsymbol{\theta}_{v_1|u}$ is the CPT for the link $u \rightarrow v_1$ and $\boldsymbol{\lambda}(u)$ is a row vector.

We examine the case where the leaf nodes $\mathbf{w}$ are instantiated with the evidence (image) $\mathbf{X}$ and derive an expression for the likelihood $P(\mathbf{X}|\mathbf{Z})$ with $\mathbf{Z} = \{u, \mathbf{v}, \mathbf{w}\}$.

$$\begin{aligned} P(\mathbf{X}|\mathbf{Z}) &= P(\mathbf{w}|\mathbf{Z}) \\ &= \int_{u,\mathbf{v}} P(u, \mathbf{v}, \mathbf{w}|\mathbf{Z}) du d\mathbf{v} \end{aligned} \tag{A.4}$$

Now the nodes are discrete so the integral becomes a sum, and using the product rule (A.4) can be rewritten

$$\begin{aligned} P(\mathbf{X}|\mathbf{Z}) &= \sum_u P(u) \sum_{\mathbf{v}} P(\mathbf{v}|u) P(\mathbf{w}|\mathbf{v}) \\ &= \sum_u P(u) \sum_{v_1} P(v_1|u) P(w_1|v_1) P(w_2|v_1) \\ &\quad . \sum_{v_2} P(v_2|u) P(w_3|v_2) P(w_4|v_2) \end{aligned} \tag{A.5}$$

$P(w_j|v_i)$ can be seen from the definition of Equation (A.1) to simply be $\boldsymbol{\lambda}_{w_j}(v_i)$ as w_j is at the leaf level of the network and is instantiated with the evidence x_j. Substituting in (A.5) gives

$$P(\mathbf{X}|\mathbf{Z}) = \sum_u P(u) \sum_{v_1} P(v_1|u) \boldsymbol{\lambda}_{w_1}(v_1) \boldsymbol{\lambda}_{w_2}(v_1) \sum_{v_2} P(v_2|u) \boldsymbol{\lambda}_{w_3}(v_2) \boldsymbol{\lambda}_{w_4}(v_2)$$

Using (A.2)

$$P(\mathbf{X}|\mathbf{Z}) = \sum_u P(u) \sum_{v_1} P(v_1|u) \boldsymbol{\lambda}(v_1) \sum_{v_2} P(v_2|u) \boldsymbol{\lambda}(v_2) \tag{A.6}$$

$P(v_1|u)$ is given by the conditional probability matrix $\boldsymbol{\theta}_{v_1|u}$, so

$$P(\mathbf{X}|\mathbf{Z}) = \sum_u P(u) \boldsymbol{\lambda}(v_1) \boldsymbol{\theta}^T_{v_1|u} \boldsymbol{\lambda}(v_2) \boldsymbol{\theta}^T_{v_2|u} \tag{A.7}$$

$$= \sum_u P(u) \boldsymbol{\lambda}_{v_1}(u) \boldsymbol{\lambda}_{v_2}(u) \tag{A.8}$$

$$= \sum_u P(u) \boldsymbol{\lambda}(u) \tag{A.9}$$

Expression (A.8) is as a result of substituting in $\boldsymbol{\lambda}$s from Equation (A.3), then by Equation (A.2) we get the final expression in (A.9).

The significance of this result is that the likelihood for any tree network can be obtained from the dot product of the $\boldsymbol{\lambda}$ vector in the root node with its prior probability. Since the algorithm for computing the $\boldsymbol{\lambda}$s is linear time this makes it quick to compute.

This result though shown for a 3-level binary tree generalises to any tree structure of any depth by virtue of (A.2) which fuses all the lambda messages passed from the children into a single one for the parent. Thus the only constraint on the architecture is that it does not contain any multiple paths (or loops).

Appendix B

Mean Field Derivation for Dynamic Trees

Begin by ordering the U nodes of the *dynamic tree* model sequentially from the top level root to the last leaf as shown in the Figure.

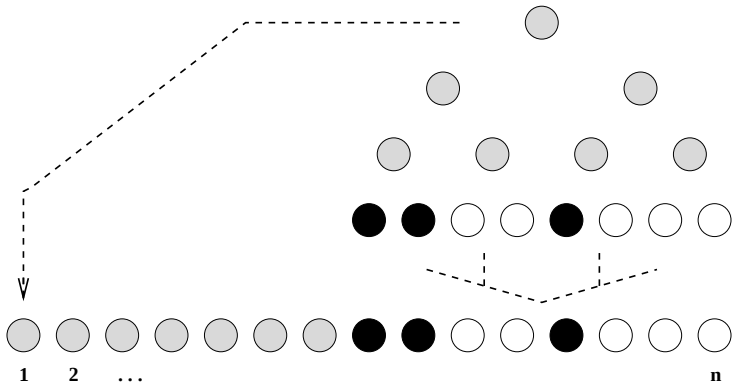


Figure B.1: Mean field node ordering.

Defining the indicator variable denoting the link from node i to its parent j as z_{ij} and π_{ij} as the prior probability of making that connection, then the prior $P(\mathbf{Z}|\boldsymbol{\phi})$ for the *dynamic tree* can be written as

$$P(\mathbf{Z}|\boldsymbol{\phi}) = \prod_{i=1,j=0}^{U} \pi_{ij}^{z_{ij}}, \tag{B.1}$$

where $\boldsymbol{\phi} = \{\pi_{ij}\}$ is the set of all prior probabilities. There are many ways to construct the prior from a set hyper-parameters some of which are discussed in Section 5.5. It is not of concern to us here how they were obtained.

The nodes in the DT model take on $1 \ldots C$ discrete states, and $\boldsymbol{\theta}$ defines the transition probabilities between connected nodes. Thus θ_{ij}^{kl} denotes the probability of node i being in state k given that its parent j is in state l. Analogously with the z indicator variables we can define indicators for node states where $x_i^k = 1$ when node i is in state k and zero otherwise. The full joint distribution for the *dynamic tree* can now be written as

$$P(\mathbf{Z}, \mathbf{X}|\boldsymbol{\theta}, \boldsymbol{\phi}) = \prod_{i=1}^{U} \prod_{j=0}^{U} \pi_{ij}^{z_{ij}} \prod_{kl} [\theta_{ij}^{kl}]^{x_i^k x_j^l z_{ij}}. \tag{B.2}$$

The nodes of the *dynamic tree* constitute two distinct sets. The first contains evidential or visible units $\mathbf{X}_v$ which are instantiated with the image data. The second are the hidden units $\mathbf{X}_h$ whose value has to be inferred. Conditioning on the training data (visible units) the posterior distribution of the *dynamic tree* takes the form

$$P(\mathbf{Z}, \mathbf{X}_h|\mathbf{X}_v) = \frac{P(\mathbf{Z}, \mathbf{X})}{P(\mathbf{X}_v)}. \tag{B.3}$$

In the mean field variational approach this posterior distribution is approximated by a factorising distribution of the form $Q(\mathbf{Z}|\mathbf{X}_v)Q(\mathbf{X}_h|\mathbf{X}_v)$ where $Q(\mathbf{Z}|\mathbf{X}_v)$ approximates over the $\mathbf{Z}$ distribution and $Q(\mathbf{X}_h|\mathbf{X}_v)$ the hidden units $\mathbf{X}_h$. The Kullback-Liebler (KL) divergence provides a convenient measure of the distance between two functional forms. Choosing good forms for the Q distribution is achieved by minimising the KL divergence between the approximating $Q(\mathbf{Z})Q(\mathbf{X}_h)$[1] and the true posterior distribution.

[1] the conditioning on $\mathbf{X}_v$ is dropped for clarity.

The KL divergence for this is given by

$$\begin{aligned} KL(Q||P) &= \sum_{\mathbf{Z},\mathbf{X}_h} Q(\mathbf{Z},\mathbf{X}_h) \log\left(\frac{Q(\mathbf{Z},\mathbf{X}_h)}{P(\mathbf{Z},\mathbf{X}_h|\mathbf{X}_v)}\right) \equiv \Delta \\ &= \sum_{\mathbf{Z},\mathbf{X}_h} Q(\mathbf{Z},\mathbf{X}_h) \log\left(\frac{Q(\mathbf{Z},\mathbf{X}_h)}{P(\mathbf{Z},\mathbf{X})/P(\mathbf{X}_v)}\right) \\ = \log P(\mathbf{X}_v) &- \sum_{\mathbf{Z},\mathbf{X}_h} Q(\mathbf{Z},\mathbf{X}_h)[\log P(\mathbf{Z},\mathbf{X}) - \log Q(\mathbf{Z},\mathbf{X}_h)]. \end{aligned} \tag{B.4}$$

The assumption of a factorising distribution for the Qs makes the states X independent of the the tree structure Z and KL divergence simplifies to

$$\begin{aligned} \Delta \;=\; & \log P(\mathbf{X}_v) - \sum_{\mathbf{Z}} Q(\mathbf{Z}) \sum_{\mathbf{X}_h} Q(\mathbf{X}_h)[\log P(\mathbf{Z},\mathbf{X}_h,\mathbf{X}_v) \\ & - \log Q(\mathbf{Z}) - \log Q(\mathbf{X}_h)]. \end{aligned} \tag{B.5}$$

This factorisation also allows separate treatment of the Xs and Zs.

B.1 Calculating $Q(\mathbf{Z})$

To optimise $Q(\mathbf{Z})$ the Kullback-Liebler divergence (Equation (B.4)) is differentiated with respect to Z

$$\Delta_{\mathbf{Z}} \;=\; \sum_{\mathbf{Z}} Q(\mathbf{Z}) \left[\sum_{\mathbf{X}_h} Q(\mathbf{X}_h).\log P(\mathbf{Z},\mathbf{X}_h,\mathbf{X}_v) - \log Q(\mathbf{Z})\right] \tag{B.6}$$

Taking logs of the joint *dynamic tree* distribution (Equation (B.2)) gives

$$\log P(\mathbf{Z},\mathbf{X}) = \sum_{i,j<i} z_{ij} \left[\log \pi_{ij} + \sum_{kl} x_i^k x_j^l \log \theta_{ij}^{kl}\right] \tag{B.7}$$

which can then be used to substitute for $\log P(\mathbf{Z},\mathbf{X})$ in Equation (B.6). Performing this

substitution and solving for the minimum gives

$$\begin{aligned}\log Q(\mathbf{Z}) &\propto \sum_{\mathbf{X}_h} Q(\mathbf{X}_h) P(\mathbf{Z}, \mathbf{X}_h, \mathbf{X}_v) \\ &\propto \sum_{\mathbf{X}_h} Q(\mathbf{X}_h) \left[\sum_{i,j<i} z_{ij} \left\{ \log \pi_{ij} + \sum_{kl} x_i^k x_j^l \log \theta_{ij}^{kl} \right\} \right] \\ &\propto \sum_{i,j<i} z_{ij} \left\{ \log \pi_{ij} + \sum_{kl} \langle x_i^k x_j^l \rangle_{Q(X)} \log \theta_{ij}^{kl} \right\} \end{aligned} \tag{B.8}$$

Now define

$$\nu_{ij} = \log \pi_{ij} + \sum_{kl} \langle x_i^k x_j^l \rangle_{Q(X)} \log \theta_{ij}^{kl}. \tag{B.9}$$

Substituting in the above and taking the exponential produces

$$Q(\mathbf{Z}) \propto \prod_i \exp\left(\sum_{j<i} z_{ij} \nu_{ij} \right). \tag{B.10}$$

The final step uses Lagrange multipliers to constrain the Zs to be valid probabilities such that $\sum_j Q(z_{ij}) = 1$, and gives rise to the following mean field update equation for the *dynamic tree* structure

$$Q(\mathbf{Z}) = \prod_{ij} \frac{\exp(z_{ij} \nu_{ij})}{\sum_s \exp(\nu_{is})} \tag{B.11}$$

The nice thing about this derivation is that without any further assumptions other than that $\mathbf{X}_h$ and $\mathbf{Z}$ are independent we arrive at a fully factorised distribution for $Q(\mathbf{Z})$.

B.2 Calculating $Q(\mathbf{X}_h)$

The optimisation of $Q(\mathbf{X}_h)$ proceeds along similar lines, by firstly differentiating Equation (B.4) wrt to $\mathbf{X}_h$

$$\Delta_{\mathbf{X}} = \sum_{\mathbf{X}_h} Q(\mathbf{X}_h) \left[\sum_{\mathbf{Z}} Q(\mathbf{Z}) \log P(\mathbf{Z}, \mathbf{X}_h, \mathbf{X}_v) - \log Q(\mathbf{X}_h) \right]. \tag{B.12}$$

As well as taking the log of the joint *dynamic tree* probability distribution giving in Equation (B.2) we also average over $Q(\mathbf{Z})$ as follows

$$\sum_{\mathbf{Z}} Q(\mathbf{Z}) \log P(\mathbf{Z},\mathbf{X}) = \sum_{i,j<i} \langle z_{ij} \rangle_{Q(Z)} \left[\log \pi_{ij} + \sum_{kl} x_i^k x_j^l \log \theta_{ij}^{kl} \right]. \tag{B.13}$$

Define $\mu_{ij} = \langle z_{ij} \rangle_{Q(Z)}$, and substitute the above into Equation (B.12) to obtain

$$\begin{aligned}
\Delta_{\mathbf{X}} &= -\sum_{\mathbf{X}_h} Q(\mathbf{X}_h) \log Q(\mathbf{X}_h) + \\
&\quad \sum_{\mathbf{X}_h} Q(\mathbf{X}_h) \left[\sum_{i,j<i} \mu_{ij} \left\{ \log \pi_{ij} + \sum_{kl} x_i^k x_j^l \log \theta_{ij}^{kl} \right\} \right] \\
&= -\sum_{\mathbf{X}_h} Q(\mathbf{X}_h) \log Q(\mathbf{X}_h) + \left[\sum_{i,j<i} \mu_{ij} \left\{ \log \pi_{ij} + \langle \sum_{kl} x_i^k x_j^l \log \theta_{ij}^{kl} \rangle_{Q(X)} \right\} \right] \\
&= -\sum_{\mathbf{X}_h} Q(\mathbf{X}_h) \log Q(\mathbf{X}_h) + \left[\sum_{i,j<i} \mu_{ij} \left\{ \log \pi_{ij} + \sum_{kl} \langle x_i^k x_j^l \rangle_{Q(X)} \log \theta_{ij}^{kl} \right\} \right]
\end{aligned} \tag{B.14}$$

where the term involving μ_{ij} has been averaged over $Q(\mathbf{X})$, noting that $\langle \sum_{kl} x_i^k x_j^l \log \theta_{ij}^{kl} \rangle_{Q(X)}$ $= \sum_{kl} \langle x_i^k x_j^l \rangle_{Q(X)} \log \theta_{ij}^{kl}$.

Computation of $\langle x_i^k x_j^l \rangle_{Q(X)}$ raises the same tractability issues as for the true posterior and to proceed any further we need to make a second assumption that all of the xs are independent. So we define

$$\begin{aligned}
Q(\mathbf{X}_h) &= \prod_i Q(x_i) \\
&= \prod_{i,k} (m_i^k)^{x_i^k}
\end{aligned} \tag{B.15}$$

such that m_i^k is the mean probability that node x_i is in state k. In order to ensure that the ms sum to unity over states it is necessary to introduce a Lagrange multiplier term $\sum_\alpha \rho_\alpha \left(\sum_\beta m_\alpha^\beta - 1 \right)$. With these substitutions Equation (B.14) becomes

$$\begin{aligned}
\Delta_{\mathbf{X}} &= -\sum_{i,k} m_i^k \log m_i^k + \sum_{i,j<i} \mu_{ij} \left[\log \pi_{ij} + \sum_{kl} m_i^k m_j^l \log \theta_{ij}^{kl} \right] \\
&\quad -\sum_\alpha \rho_\alpha \left(m_\alpha^\beta - 1 \right).
\end{aligned} \tag{B.16}$$

Optimising the above with respect to the means m_i^k produces

$$\begin{aligned}\frac{\partial \Delta_{\mathbf{X}}}{\partial m_s^r} &= -1 - \log m_s^r + \sum_{i,j<i} \mu_{ij} \left[\sum_{kl} \left\{ m_i^k \delta_{sj} \delta_{rl} + m_j^l \delta_{si} \delta_{rk} \right\} \log \theta_{ij}^{kl} \right] - \rho_s \\ &= -1 - \log m_s^r - \rho_s + \sum_{j<i} \sum_l \mu_{sj} m_j^l \log \theta_{sj}^{rl} + \sum_i \sum_k \mu_{is} m_i^k \log \theta_{is}^{kr} \end{aligned} \tag{B.17}$$

where δ_{ab} is the Kronecker delta. Defining $\gamma_s^r = \sum_{j<i} \sum_l \mu_{sj} m_j^l \log \theta_{sj}^{rl} + \sum_i \sum_k \mu_{is} m_i^k \log \theta_{is}^{kr}$ produces a similar form for the update equation of the means as was obtained for $Q(\mathbf{Z})$. Normalising in the same way gives the means update rule

$$m_s^r = \frac{\exp \gamma_s^r}{\sum_{r'} \exp \gamma_s^{r'}} \tag{B.18}$$

However since the right hand side of Equation (B.18) contains terms involving m we have a set of coupled linear equations which need to be solved iteratively.

B.3 The Full Mean Field Algorithm

The full set of mean field Equations for the *dynamic tree* are summarised below.

$$Q(Z) = \prod_{ij} \frac{\exp(z_{ij} \nu_{ij})}{\sum_s \exp(\nu_{is})} \tag{B.19}$$

$$m_s^r = \frac{\exp(\gamma_s^r)}{\sum_r \exp(\gamma_s^r)} \tag{B.20}$$

where

$$\begin{aligned}\nu_{ij} &= \log \pi_{ij} + \sum_{kl} \langle x_i^k x_j^l \rangle_{Q(X_h)} \log \theta_{ij}^{kl} \\ \gamma_s^r &= \sum_j \sum_l \mu_{sj} m_j^l \log \theta_{sj}^{rl} + \mu_{js} m_j^l \log \theta_{js}^{rl} \\ \mu_{ij} &= \langle z_{ij} \rangle_{Q(Z)} \end{aligned}$$

The update algorithm is of necessity iterative requiring a number of cycles through each mean in turn, before $Q(\mathbf{Z})$ can be updated in a single step. To reach an equilibrium this whole process usually needs repeating.

Specific details and full discussion of this are given in Chapter 4.

Bibliography

Adams, N. J. and Williams, C. K. I. (1999). SDTs: Sparse Dynamic Trees. In *Proceedings of 9th International Conference on Artificial Neural Networks*, pages 527–532. IEE.

Adams, N. J., Storkey, A. J., Ghahramani, Z., and Williams, C. K. I. (2000). MFDTs: Mean Field Dynamic Trees. In A. Sanfeliu, J. J. Villanueva, A. Vanrell, R. Alquézar, T. Huang, and J. Serra, editors, *Proceedings of 15th International Conference Pattern Recognition*, volume 3, *Image speech and Signal Processing*, pages 151–154. IEEE Computer Society.

Adams, N. J., Williams, C. K. I., and Storkey, A. J. (2001). Comparing Mean Field and Exact EM in Tree Structured Belief Networks. In *Fourth International ICSC Symposium on Soft Computing and Intelligent Systems for Industry*. ICSC-NAISO Adademic Press.

Baldi, P. and Pineda, F. (Winter 1991). Contrastive Learning and Neural Oscillations. *Neural Computation*, **1199**(3(4)), 526–545.

Bienenstock, E. and Doursat, R. (1990). Issues of Representation in Neural Networks. In *European Conference on Visual Perception*.

Bienenstock, E. and Doursat, R. (1994). A Shape-Recognition Model Using Dynamical Links. In *Network: Computation in Neural Systems*, volume 5, pages 241–258.

Bienenstock, E., Geman, S., and Potter, D. (1997). Compositionality, MDL Priors, and

Object Recognition. Division of Applied Mathematics, Brown University, Providence, RI 02912 USA.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*, pages 65–73. Oxford University Press, Oxford, UK, 1st edition.

Bouman, C. A. and Shapiro, M. (1994). A Multliscale Random Field Model for Bayesian Image Segmentation. *IEEE Transactions on Image Processing*, **3(2)**, 162–177.

Bridle, J. S. (1990). Probabilistic Interpretation of Feedforward Classification Network Outputs, With Relationships to Statistical Pattern Recognition. In F. Fogelman Soulié and J. Hérault, editors, *Neurocomputing: Algorithms, Architectures and Applications*, pages 227–236. Springer-Verlag New York inc.

Burt, P. J. and Adelson, E. H. (1987). The Lapacian Pyramid as a Compact Image Code. In M. A. Fischler and O. Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles and Pardigms*, pages 671–679. Morgan Kaufman Publishers Inc., California, USA.

Castillo, E., Gutiéz, J. M., and Hadi, A. S. (1997). *Expert Systems and Probabilistic Network Models*. Springer-Verlag New York inc., New York, USA.

Chellappa, R. and Chatterie, S. (1985). Classification of Textures using Guassian Markov Random Fields. In *IEEE Trans. Accoust., Speech and Signal Processing*, volume 33, pages 959–963.

Chellappa, R. and Jain, A. (1993). *Markov Random Fields - Theory and Applications*. Academic Press Ltd, London, UK.

Chou, P. A. (1989). Recgonition of Equations Using a Two-Dimensional Stochastic Context-Free Grammar. *Visual Communications and Image Processing IV*, **1199**, 852–863.

Cowles, M. K. and Carlin, B. P. (1996). Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. In *Journal of the American Statistical Association*, volume 91, no. 434, pages 883–904.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz Machine. *Neural Computation*, **7(5)**.

De Bonet, J. S. and Viola, P. A. (1998). A Non-Parametric Multi-Scale Statistical Model for Natural Images. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*. MIT Press, Cambridge, MA.

De Bonet, J. S., Viola, P., and Fisher, J. (1998). Flexible Histograms: A Multiresolution Target Discrimination Model. In *Proceedings of SPIE*.

Feng, X. and Williams, C. K. I. (1998). Training Bayesian Networks for Image Segmentation. In *Proceedings of SPIE*, volume 3457.

Feng, X., Williams, C. K. I., and Felderhof, S. N. (2001). Combining Belief Networks and Neural Networks for Scene Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Accepted for publication.

Fieguth, P. W., Willsky, A. S., and Karl, W. C. (1994). Multiresolution Stochastic Imaging of Satellite Oceanographic Altimetric data. *IEEE International Conference on Image Processing*, **2**, 1–5.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability - A Guide to the Theory of NP–Completeness*. W.H. Freeman and Co., New York, USA.

Geiger, D. and Heckerman, D. (1996). Knowledge Representation and Inference in Similarity Networks and Bayesian Multinets. *Artificial Intelligence*, **82**, 45–74.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian Data Analysis*. Chapman and Hall, London, UK.

Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 6, no. 6, pages 721–741.

Geman, S. and Manbeck, K. (1994). Experiments in Syntactic Recognition. Technical Report CICS-P-411, Division of Applied Mathematics, Brown University, Providence, RI 02912 USA.

Ghahramani, Z. (1997). On Structured Variational Approximations. Technical Report CRG-TR-97-1, Department of Computer Science, University of Toronto, 6 King's College Road, Toronto, Canada, M5S 1A4.

Ghahramani, Z. and Hinton, G. E. (1996). The EM Algorithm for Mixtures of Factor Analyzers. Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto, 6 King's College Road, Toronto, Canada M5S 1A4.

Ghahramani, Z. and Hinton, G. E. (1998). Switching State-Space Models. ftp://ftp.cs.toronto.edu/pub/zoubin/switch-ftp.ps.gz.

Gibbons, A. (1988). *Algorithmic Graph Theory*. Cambridge University Press, Cambridge, UK, 3rd edition.

Godsill, S. J. (1997). Some New Relationships Between MCMC Model Uncertainty Methods. Technical Report CUED/F-INFENG/TR.305, Signal Processing and Communications Laboratory, Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK.

Hinton, G., Ghahramani, Z., and Teh, Y. W. (2000). Learning to Parse Images. In S. A. Solla, T. K. Leen, and K. R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 463–469. MIT Press.

Hinton, G. E., Sallans, B., and Ghahramani, Z. (1998). A Hierarchical Community of Experts. In C. M. Bishop, editor, *Neural Networks and Machine Learning*. Springer-Verlag New York inc.

Hummel, J. E. and Biedermann, I. (1992). Dynamic Binding in a Neural Network for Shape Recognition. *Psychological Review*, **99**(3), 480–517.

Irving, W. W., Fieguth, P. W., and Willsky, A. S. (1997). An Overlapping Tree Approach to Multiscale Stochastic Modeling and Estimation. *IEEE Transactions on Image Processing*, **6**(11), 1517–1529.

Istas, J. (1995). Nonparametric Supervised Image Segmentation by Energy Minimisation Using Wavelets. In A. Antoniadis and G. Oppenheim, editors, *Lecture Notes in Statistics 103, Wavelets and Statistics*, pages 169–192. Springer-Verlag New York inc., New York, USA.

Jain, A. K. and Nadabar, S. G. (1993). Range Image Segmentation Using MRF Models. In R. Chepella and A. Jain, editors, *Markov Random Fields – Theory and Applications*, pages 543–572. Academic Press Ltd, London, UK.

Jordan, M. and Saul, L. (1998). A Mean Field Learning Algorithm for Unsupervised Neural Networks. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 541–554. Kluwer Academic Publishers.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1998). An Introduction to Variational Methods For Graphical Models. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 105–161. Kluwer Academic Publishers.

Lades, M., Vorbrüggen, J. C., Buhmann, J., Lange, J., von der Malsburg, C., Würtz, R. P., and Konen, W. (1993). Distortion Invariant Object Recognition in the Dynamic Link Architecture. In *IEEE Transactions on Computers*, volume 42, no. 3, pages 300–311.

Laferté, J. M., Pérez, P., and Heitz, F. (2000). Discrete Markov Image Modelling and Inference on the Quadtree. *IEEE Transactions on Image Processing*, **9**(3), 390–404.

Little, R. J. A. and Rubin, D. B. (1987). *Statistical Analysis with Missing Data*. John Wiley, New York, USA.

Luettgen, M. R. and Willsky, A. S. (1995). Likelihood Calculation for a Class of Multiscale Stochastic Models, with Application to Texture Discrimination. *IEEE Transactions on Image Processing*, **4(2)**, 194–207.

MacKay, D. J. C. (1995). Developments in Probabilistic Modelling with Neural Networks–Ensemble Learning. In *Neural Networks: Artificial Intelligence and Industrial Applications. Proceedings of the 3rd Annual Symposium on Neural Net-*

works, Nijmegen, Netherlands, 14-15 September 1995, pages 191–198, Berlin. Springer.

Meer, P. and Connelly, S. (1989). A Fast Parallel Method for Synthesis of Random Patterns. In *Pattern Recognition*, volume 22, no. 2, pages 189–204.

Montanvert, A., Meer, P., and Rosenfeld, A. (1991). Hierarchical Image Analysis Using Irregular Tessellations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **13(4)**, 307–316.

Neal, R. and Hinton, G. E. (1998). A View of the EM Algorithm That Justifies Incremental, Sparse, and Other Variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers.

Neal, R. M. (1993). Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical Report CRG-TR-93-1, Connectionist Research Group, Department of Computer Science, University of Toronto, Toronto, Ontario, M5S 1A4.

Neapolitan, R. (1990). *Probabilistic Reasoning in Expert Systems – Theory and Algorithms*. John Wiley and Sons inc.

Olshausen, B. A. and Field, D. J. (1996). Emergence of Simple-cell Receptive Field Properties by Learning a Sparse Code for Natural Images. *Nature*, **381**, 607–609.

Pearl, J. (1988a). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman Publishers Inc., San Francisco, USA, 2nd edition.

Pearl, J. (1988b). *Probabilistic Reasoning in Intelligent Systems*, pages 150–174. In Pearl (1988a), 2nd edition.

Pentland, A. (1989). Part Segmentation for Object Recognition. *Neural Computation*, **1**(1), 82–91.

Rees, S. and Ball, R. C. (1987). Criteria for an Optimum Simulated Annealing Schedule for Problems of the Travelling Salesman Type. *Phys A: Math. Gen*, **20**, 1239–1249.

Sallans, B., Hinton, G. E., and Ghahramani, Z. (1998). A Hierarchical Community of Experts. In C. M. Bishop, editor, *Neural Networks and Machine Learning*. Springer.

Sampson, G. (1996). *Communications in Artificial Intelligence: Evolutionary Language Understanding*. Cassel, London, UK.

Saul, L. K. and Jordan, M. I. (1996). Exploiting Tractable Substructures in Intractable Networks. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*. MIT Press.

Spence, C. D. and Parra, L. (2000). Hierarchical Image Probability (HIP) Models. In *Advances in Neural Information Processing Systems 12*. MIT Press.

Stander, J. and Silverman, B. W. (1994). Temperature Schedules for Simulated Annealing. *Statistics and Computing*, **4**, 21–32.

Storkey, A. J. (2000). Dynamic Trees: A structured Variational Approach Giving Efficient Propagation Rules. In *Uncertainty in Artificial Intelligence (UAI2000)*.

Taylor, C. C. and Henery, R. J. (1994). Comparative Trials in Classification of Image Data. *Journal of Applied Statistics, Supplement on Advances in Applied Statistics: Statistics and Images: 2*, **21**(1), 77–91.

Utans, J. (1994). Learning in Compositional Hierarchies: Inducing the Structure of Objects from Data. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*. Morgan Kaufman Publishers Inc.

Utans, J. and Gindi, G. (1993). Improving Convergence in Hierarchical Matching Networks for Object Recognition. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*. Morgan Kaufman Publishers Inc.

von der Malsburg, C. (1988). Pattern Recognition by Labelled Graph Matching. In *Neural Networks*, volume 1, pages 141–148.

Williams, C. K. I. and Adams, N. J. (1999). DTs: Dynamic Trees. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 634–640. MIT Press.

Williams, C. K. I. and Feng, X. (1998). Combining Neural Networks and Belief Networks for Image Segmentation. In *Proceedings of IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing*.

Williams, C. K. I. and Feng, X. (1999). Tree-Structured Belief Networks as Models of Images. In *Proceedings of 9th International Conference on Artificial Neural Networks*, pages 31–36. IEE.