Aspects of the Interface Between Statistics and Neural Networks

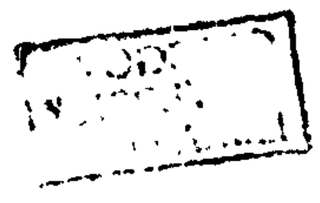
UNIVERSITY
of
GLASGOW

Matt Whiley

A Dissertation Submitted to the
University of Glasgow
for the degree of
Doctor of Philosophy

Department of Statistics

December 1999



Abstract

In recent years the cross-fertilisation of ideas between the statistics and machine learning communities has become increasingly important. This exchange of ideas resulted from a recognition that the two communities often have to tackle similar problems and has resulted in an exchange which has enriched both disciplines. There is much to be gained in considering the two literatures in tandem, and the aim of this thesis is to build on some of the research currently taking place at the interface between these two disciplines.

Specifically we will be considering a class of models called Bayesian belief networks. These are models which are closely related to neural networks, a type of model often used in machine learning but largely eschewed by statisticians due to their 'black box' approach. Neural networks, while useful tools, lack transparency; by their nature it is difficult to interpret the method in which neural network models process their inputs in order to produce the observed output. Bayesian networks provide an attractive alternative, allowing us to take some of the best elements of neural network methodology and apply it to a system in which we have clear understanding of cause and effect between variables. This greater interpretability can lead to new insight and Bayesian network methodology has recently become a popular tool for giving machines a degree of 'intelligence'.

We begin in Chapter 1 by introducing the reader to Bayesian belief networks and outlining some of the issues that must be dealt with when working with these models.

Chapter 2 then details various methods that can be used to estimate the parameters of these models when we have incomplete data sets. We compare the

EM algorithm to various alternatives that have been suggested both to improve on the convergence rate of the EM algorithm and to try to find a better stationary point than we might find with the EM algorithm alone. In this chapter we also detail a new parameter estimation algorithm.

In Chapter 3 we consider issues of model identifiability, review the current literature relating to model identifiability problems, and offer a proof that naive Bayesian networks with a binary root node and more than 2 binary observable variables will be identifiable.

We then consider model selection in Chapter 4. We review a number of model selection criteria that are currently available and compare their performance on artificial datasets and on a dataset concerned with the survival of patients admitted to an adult intensive care unit.

Finally we discuss the application of Green's reversible jump Markov chain Monte Carlo algorithm to naive Bayesian networks in Chapter 5. This method allows us to tackle issues of parameter estimation and model selection simultaneously. While the simultaneous solving of these twin problems is an attractive proposition the postprocessing of the output generated by this method raises issues of interpretation which we attempt to tackle.

Acknowledgements

- I wish to thank my supervisor Professor Mike Titterington for his invaluable guidance and for the encouragement he has given me. Without his help the completion of this thesis would not have been possible.

- I would also like to thank all of the staff and students, past and present, of the statistics department in Glasgow for making my time here happy and even, dare I say it, productive.

- A big thank you to my family for their support.

- And finally, cheers darlin' for all the good times and gin!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Imagine we have some domain $\boldsymbol{X}$ which contains a number of variables; probabilistic relationships exist between these variables, the current state of each one influencing some or all of the other variables under consideration. We might be interested in making some statements about this domain; perhaps having observed some variables we wish to make some inference about those we have not seen. Depending upon the size of the domain, and how complicated the dependencies within it are, making such statements can be difficult. What is needed is some way of breaking the problem down into smaller, more manageable chunks. A Bayesian network is a useful tool for tackling such problems; it provides a convenient graphical summary of our variables and their relationships and helps us to clarify our understanding of the domain being studied. Figure 1.1 shows a simple Bayesian network. Here the domain of interest is starting a car, and this simple diagram immediately gives us some understanding of the structure

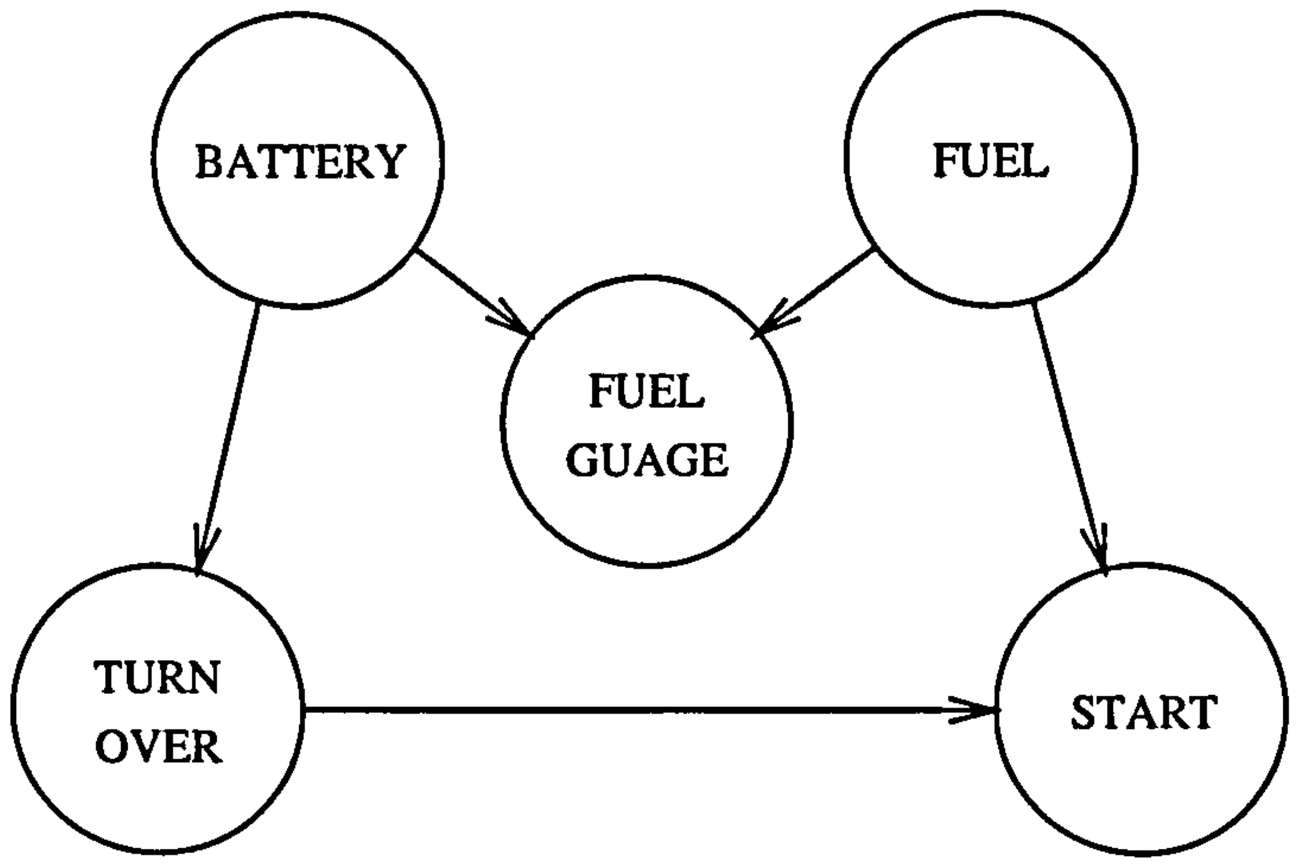


Figure 1.1: After Heckerman [32]. A simple Bayesian network; the domain of interest is starting a car. The network clearly shows the variables involved and how they influence each other.

of the problem by displaying cause and effect within the domain. Aside from displaying dependencies between our variables, a Bayesian network also contains a set of local probability distributions, one for each variable in the domain. These local distributions describe the probabilities of all possible outcomes of the variables given all the influences they may be subject to. For example in Figure 1.1 the variable 'Start' has associated with it a table containing the probability of successfully starting the car given all possible states of the two variables 'Fuel' and 'Turn Over'. We can then use these local probability distributions and our knowledge of the relationships between the variables to answer questions about the system we are looking at.

Formally, a Bayesian network for a domain $\boldsymbol{X}$ is a way of encoding the joint probability distribution for $\boldsymbol{X}$. It consists of a set of local probability distributions, one associated with each variable in $\boldsymbol{X}$, and a set of assertions of conditional independence for these variables that allow the construction of the

joint distribution from the local distributions.

The first stage in constructing a Bayesian network is to identify the variables within the domain, and to collect our data set, $D = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, which we will use to learn about our domain. Typically the variables in the domain will be categorical (or if continuous will be discretised), and in the discussion that follows we will assume we are working with categorical variables unless otherwise stated. Once this is completed the second stage is to encode our assertions of conditional independence using a directed acyclic graph (DAG), the nodes of which correspond to our variables. Given a domain, $\boldsymbol{X} = \{X_1, \ldots, X_n\}$, and an ordering on the variables, $(X_1, \ldots, X_n)$, we can use the chain rule of probability to write the joint probability distribution for $\boldsymbol{X}$ as

$$P(x_1, \ldots, x_n) = P(x_1) \prod_{i=2}^{n} P(x_i \mid x_1, \ldots, x_{i-1}). \tag{1.1}$$

Furthermore, for each X_i $(i > 1)$ there will be some subset $\boldsymbol{Pa}_i \subseteq \{X_1, \ldots, X_{i-1}\}$ such that X_i is independent of $\{X_1, \ldots, X_{i-1}\} \setminus \boldsymbol{Pa}_i$ given $\boldsymbol{Pa}_i$, and $\boldsymbol{Pa}_i$ is known as the set of *parents* of the variable X_i. It follows that, using $\boldsymbol{pa}_i$ to denote the states of $\boldsymbol{Pa}_i$ and noting that $\boldsymbol{Pa}_1$ will be the empty set, we can write

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid \boldsymbol{pa}_i). \tag{1.2}$$

Once our conditional independencies have been clarified they are represented in the DAG by drawing arcs to each node in the graph from its parents, in effect linking cause to immediate effect. Unfortunately this definition of a Bayesian network is quite unwieldy, and the resulting network depends upon the initial

ordering of the variables. A poorly chosen ordering could well fail to reveal the conditional independencies present in the domain.

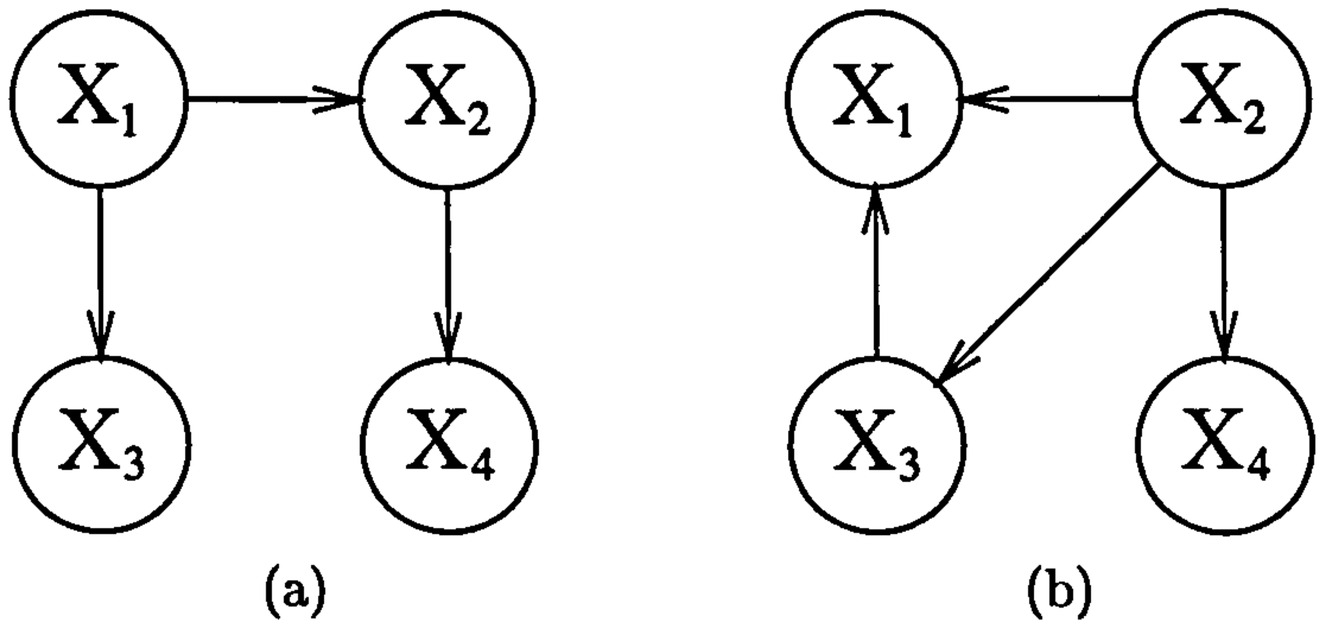


Figure 1.2: The network structure resulting from a well chosen ordering of the variables (a) and a poorly chosen ordering (b) within the domain.

For example consider the domain $\boldsymbol{X} = \{X_1, X_2, X_3, X_4\}$ for which we know the two conditional independencies

$$P(x_3 \mid x_1, x_2) = P(x_3 \mid x_1)$$

$$P(x_4 \mid x_1, x_2, x_3) = P(x_4 \mid x_2)$$

Using the ordering (X_1, X_2, X_3, X_4) gives

$$P(x_1, x_2, x_3, x_4) = P(x_1)\, P(x_2 \mid x_1)\, P(x_3 \mid x_1)\, P(x_4 \mid x_2)$$

which gives us the network structure shown in Figure 1.2(a). If however we had chosen the ordering (X_2, X_3, X_4, X_1) we would have

$$P(x_1, x_2, x_3, x_4) = P(x_2)\, P(x_3 \mid x_2)\, P(x_4 \mid x_2)\, P(x_1 \mid x_2, x_3)$$

which gives us the network structure shown in Figure 1.2(b). Clearly the first of these two structures better encodes the conditional independencies present in our domain, and this better understanding of these conditional independencies will in turn allow us to construct a more parsimonious model.

It is perhaps easier to construct the network by considering cause and effect between the variables of the domain. If these causal relationships are easily determined then deciding upon the appropriate network structure is a simple enough task. An example in which causal relationships are well known is shown in Figure 1.3, which shows a simple Bayesian network for the diagnosis of a patient with jaundice. When working in a medical context we generally have a great deal of prior knowledge of the relationships between cause, disease and symptoms which means that ascertaining the structure of our network is a fairly straightforward matter.

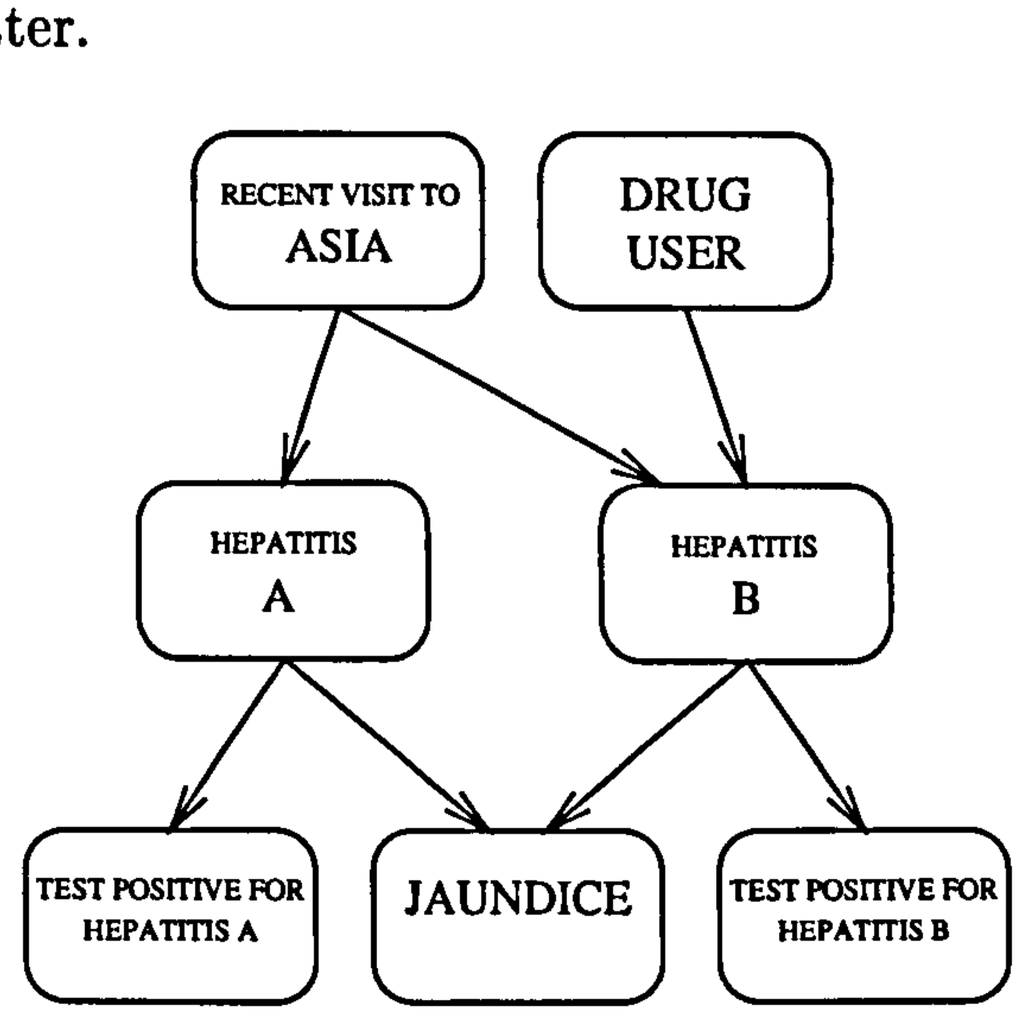


Figure 1.3: A simple Bayesian network representing the relationship between cause, disease and symptoms for two different strains of the hepatitis virus.

In other contexts the structure of the network may also be in doubt and we

are then faced with the more difficult task of selecting the best network from a (possibly large) number of alternatives. Such model selection is often performed using a scoring function which assigns a score to each network structure that reflects how well that model fits our data and any prior beliefs we might have. One possible model selection criterion is to choose the model structure that maximises

$$P(\boldsymbol{m} \mid D) \propto P(\boldsymbol{m})\, P(D \mid \boldsymbol{m}), \tag{1.3}$$

where we have defined a discrete variable $\boldsymbol{M}$ whose states $\boldsymbol{m}$ correspond to the set of model structures under consideration. In practice the quantity

$$P(D \mid \boldsymbol{m}) = \int P(D \mid \boldsymbol{\theta}_{\boldsymbol{m}}, \boldsymbol{m})\, P(\boldsymbol{\theta}_{\boldsymbol{m}} \mid \boldsymbol{m})\, d\boldsymbol{\theta}_{\boldsymbol{m}}$$

cannot easily be computed; in general this integral is intractable. However a number of approximations to this integral are available which have been suggested as suitable scoring functions for model selection. The use of such scoring functions and other potential model selection criteria is reviewed in Chapter 4.

For simple domains it is possible to look at all possible network structures and select the most suitable. However, for more complicated domains it becomes impractical to consider all possible models, and methods which search in the space of model structures become necessary. These methods often begin with a network structure constructed from any known information which the search algorithm then modifies using information gained from the data. These methods cannot claim to identify the best structure for the network, but will be able to produce a structure that has a high score.

In determining our model structure one important issue that must be considered is that of model identifiability. If a model is not identifiable it means that the model is in some sense over-parameterised. The likelihood surfaces for these parameter-redundant models possess completely flat ridges and hence we are not able to give unique estimates for some subset of our model parameters. When problems with model identifiability arise it is usually possible to re-parameterise the model in such a way that the parameters of the model become uniquely identifiable. These issues are considered in greater depth in Chapter 3.

Once our network structure is known, or perhaps as part of the selection process if more than one candidate model is being considered, we must find estimates for the local probability distributions. The parametric structure of these local probability models will have been determined either entirely from prior knowledge, or if there was some uncertainty about their structure this will have been settled in the network structure selection stage of the model construction. We define Θ_m, a continuous vector-valued variable, whose configurations $\boldsymbol{\theta}_m$ correspond to the possible true parameters for model m. $\boldsymbol{\theta}_m$ consists of $\{\theta_{ijk}\}$ where

$$\theta_{ijk} = p\left(X_i = x_i^k \mid \boldsymbol{Pa}_i = \boldsymbol{pa}_i^j, \boldsymbol{\theta}_m, m\right) \tag{1.4}$$

and

$$\sum_{k=1}^{r_i} \theta_{ijk} = 1 \quad \forall i, j \ ,$$

x_i^k denoting the k^{th} possible state of X_i and $\boldsymbol{pa}_i^j$ denoting the j^{th} possible state of $\boldsymbol{Pa}_i$. We also use r_i and q_i to denote the number of possible states of X_i and $\boldsymbol{Pa}_i$ respectively.

There are a number of possible criteria for determining the 'best' parameter

estimates for our model. In a frequentist framework we would use the Maximum Likelihood (ML) principle and choose our estimates to be the parameter values $\boldsymbol{\theta_m} \in \boldsymbol{\Theta_m}$ that maximise $P(D \mid \boldsymbol{\theta_m}, \boldsymbol{m})$. Alternatively we can work in a Bayesian framework in which case we must determine the Maximum *A Posteriori* (MAP) estimate which is $\boldsymbol{\theta_m} \in \boldsymbol{\Theta_m}$ such that $P(\boldsymbol{\theta_m} \mid D, \boldsymbol{m}) \propto P(D \mid \boldsymbol{\theta_m}, \boldsymbol{m}) P(\boldsymbol{\theta_m} \mid \boldsymbol{m})$ is maximised. If our prior $P(\boldsymbol{\theta_m} \mid \boldsymbol{m})$ is chosen to be uniform then our MAP and ML parameter estimates will coincide. Of course although the Bayesian approach is more flexible it means we also have to deal with the problem of selecting an appropriate prior distribution; this raises many issues not all of which can be satisfactorily resolved. If our data set D is complete then whichever criterion we use to select our parameter estimates the required maximisation is quite simple, but more often than not we will be working with an incomplete data set. If some of our data are missing the problem becomes more difficult, and techniques such as the EM algorithm must be used to estimate our network parameters. This and other methods for dealing with missing data problems are discussed in Chapter 2.

From a Bayesian viewpoint separate inference on the twin problems of model selection and parameter estimation is less than satisfactory. The two problems are inextricably linked, and ideally we would wish to tackle them simultaneously. This is a formidable task, if not theoretically then computationally, and it is only recently that tools suitable for tackling such problems have become available. In Chapter 5 we consider one such technique, Reversible Jump MCMC, which allows us to move between parameter spaces of different dimensionality as easily as we do within them.

1.2 The Naive Bayesian Network

Throughout much of this thesis we will concentrate our attention on one particular type of Bayesian network, the *naive* Bayesian network, an example of which is shown in Figure 1.4. In a naive network we have n fully observable

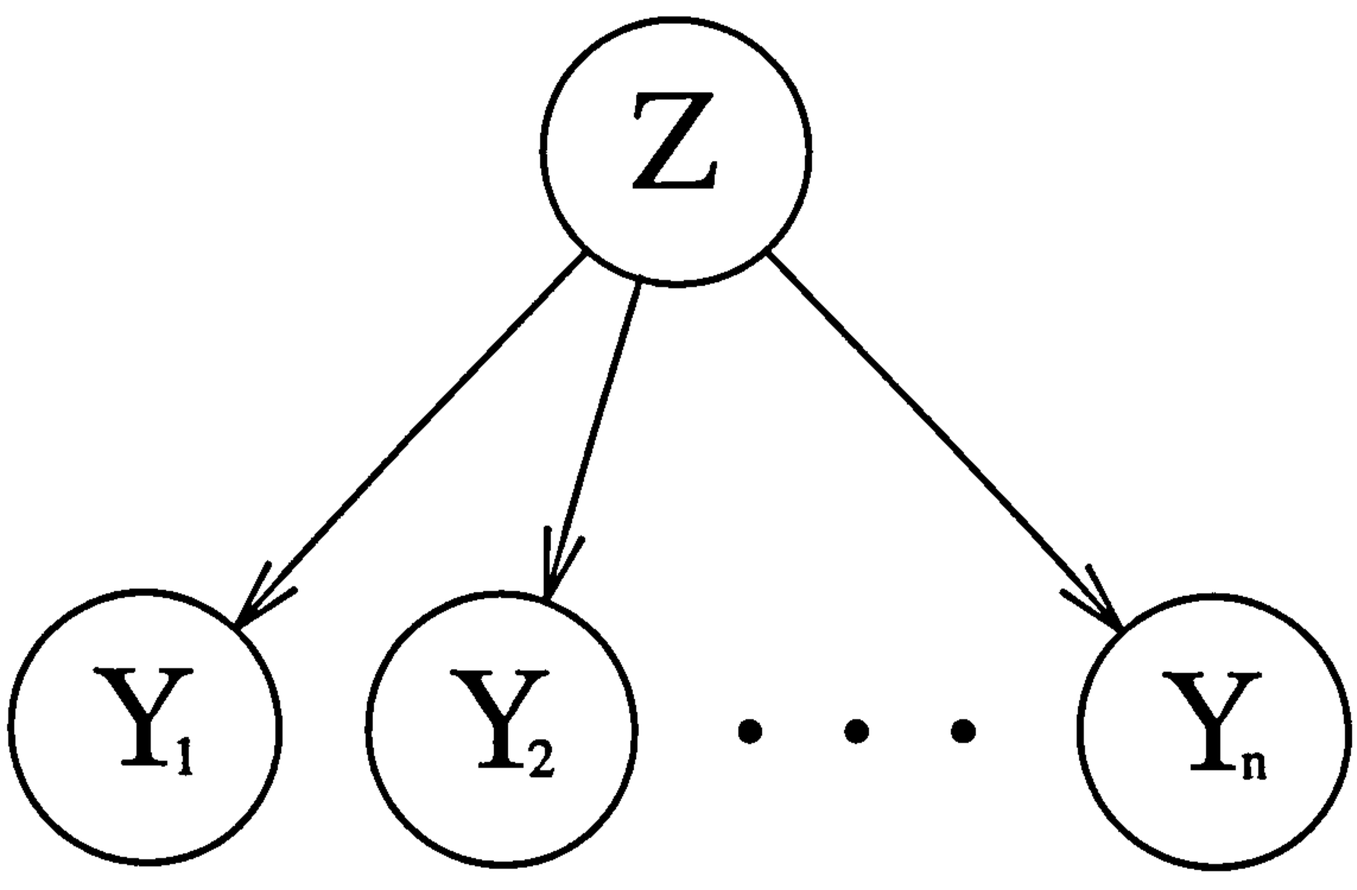


Figure 1.4: A naive Bayesian network. The nodes $Y_1, \ldots, Y_n$ are fully observable while the root node, Z, is unobserved.

nodes, $Y_1, \ldots, Y_n$, and a single unobserved node Z. We use the notation r_z to denote the number of states of the variable Z, and r_i to denote the number of states of the variable Y_i. Our dataset D is then given by $D = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$, and we will use y_{mi} to denote the observed value of Y_i in the m^{th} observation. Similarly z_m will denote the unobserved value of Z indicating the class membership of observation m. The parameters of the model are then given by $\boldsymbol{\theta_m} = \{\boldsymbol{p}, \boldsymbol{\theta}\}$ where $\boldsymbol{p} = \{p_1, \ldots, p_{r_z}\}$ is known as the *proportion vector* and consists of $p_j = P(Z = j)$, and $\boldsymbol{\theta} = \{\theta_{ijk} : i = 1, \ldots, n; j = 1, \ldots, r_z, k = 1, \ldots, r_i\}$ with $\theta_{ijk} = P(Y_i = k \mid Z = j)$. When working with naive models for the sake of simplicity we will often assume $r_i = 2$ for all i, though of course more generally this need not be the case. In these circumstances we will simplify our notation slightly by

using $\boldsymbol{\theta} = \{\theta_{i|j} : i = 1, \ldots, n; j = 1, \ldots, r_z\}$ with $\theta_{i|j} = P(Y_i = 1 \mid Z = j)$.

A naive Bayesian network is a 'latent-state' model, and represents the situation in which we believe that our observed population consists of a number of distinct sub-populations, and that within each sub-population the variables we have observed follow a different distribution. This is of course an example of a wider class of models known as mixture distributions, and it is perhaps easier to understand what a naive Bayesian network represents if we consider it in this framework. A mixture density, $f(\boldsymbol{x})$ is of the form

$$f(\boldsymbol{x}) = \sum_{j=1}^{c} w_j g_j(\boldsymbol{x}),$$

where $\sum w_j = 1$, and w_j are the weights of the components of the mixture, so that

$$P(\text{ observation comes from component } j\ ) = w_j,$$

and $g_j(\boldsymbol{x})$ is the distribution of $\boldsymbol{x}$ given that it comes from component j. In general we will take the $g_j(\cdot)$ all to come from the same parametric family of densities, though this need not necessarily be the case. Although, as said earlier, we will be working with mixtures of discrete distributions they could equally well be continuous. Figure 1.5 shows a simple example of a mixture of three one dimensional normal distributions, the three black lines represent the three distributions making up the mixture, and the shaded region shows the mixture distribution. It is this mixture rather than its individual components that we would actually observe. We would then fit a mixture model to our observed data in an attempt to unravel the confounded information we have observed, perhaps

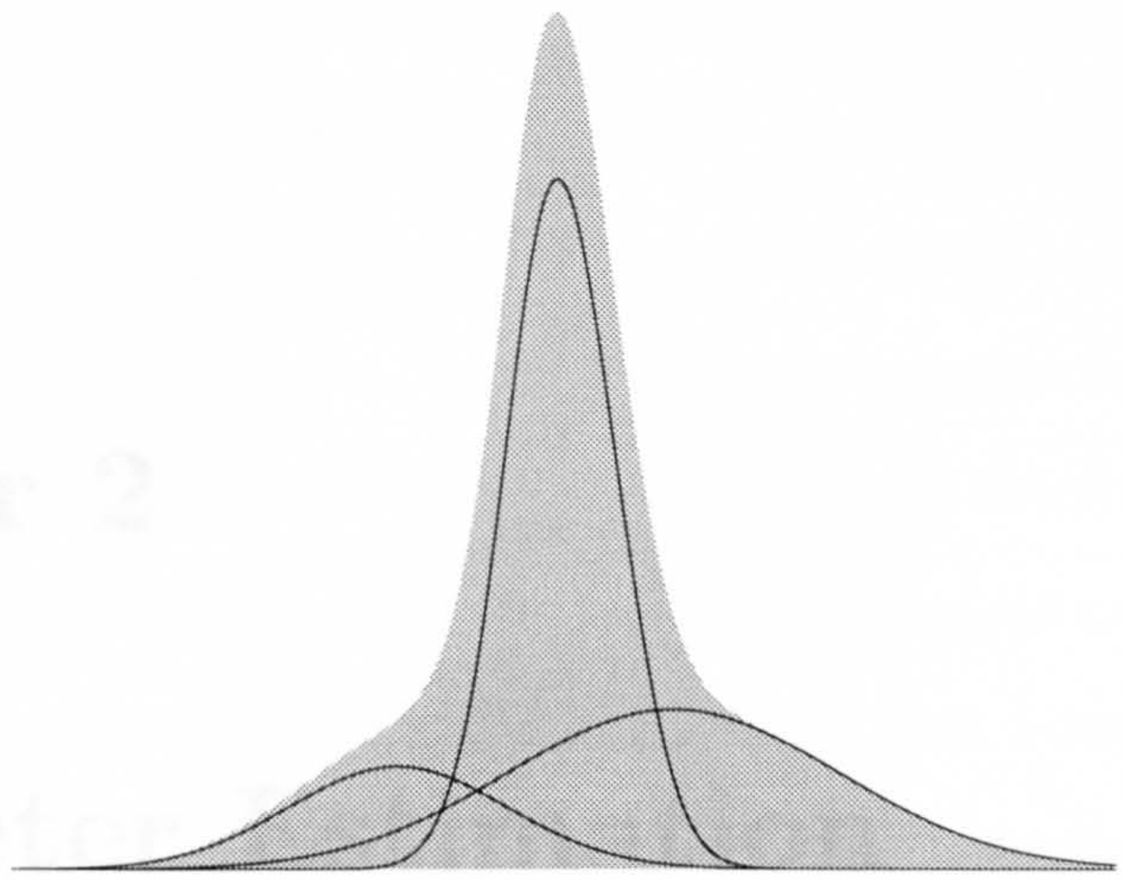


Figure 1.5: A simple example of a mixture of three one dimensional normal distributions. The black curves show the distributions of the three mixture components multiplied by their respective mixture weights. The shaded region represents the mixture, which is what we would actually see if we were to observe this population.

to find out how our sub-populations differ, to find out their relative size, or even to determine how many sub-populations are present. A naive Bayesian network does effectively the same thing, but with discrete data.

Chapter 2

Parameter Estimation

2.1 Introduction

Once the structure of our Bayesian network has been established it only remains to determine estimates for the parameters of the local probability distributions. Depending upon whether we take a Frequentist or a Bayesian viewpoint we would generally use either the Maximum Likelihood (ML) or the Maximum *a Posteriori* (MAP) method to determine our parameter estimates. In the former case we must find the value $\hat{\boldsymbol{\theta}}_m$ of $\boldsymbol{\theta}_m$ that maximises the likelihood of the data, $P(D \mid \boldsymbol{\theta}_m, m)$, or equivalently solves

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta}_m} = 0, \tag{2.1}$$

where $\mathcal{L}(\boldsymbol{\theta}_m) = \log P(D \mid \boldsymbol{\theta}_m, m)$ is known as the log-likelihood. In the Bayesian framework we would look for the value $\tilde{\boldsymbol{\theta}}_m$ of $\boldsymbol{\theta}_m$ that maximises

$$P(\boldsymbol{\theta}_m \mid D, m) \propto P(D \mid \boldsymbol{\theta}_m, m)\, P(\boldsymbol{\theta}_m \mid m), \tag{2.2}$$

where $P(D \mid \boldsymbol{\theta}_m, \boldsymbol{m})$ is the likelihood of our observed data, and $P(\boldsymbol{\theta}_m \mid \boldsymbol{m})$ and $P(\boldsymbol{\theta}_m \mid D, \boldsymbol{m})$ are respectively the prior and posterior densities of the unknown parameters. The Bayesian approach offers more flexibility allowing us to introduce our prior beliefs into the parameter estimation process, although suitable priors will not always be obvious. Of course we could always choose our priors to be uniform, in which case our MAP and ML parameter estimates will coincide. Unless otherwise stated, in the discussion that follows we will assume that we are trying to determine Maximum Likelihood parameter estimates, but all of the techniques described here can easily be extended to work in a Bayesian framework.

If our dataset is complete, that is to say we have observed every instance of each of our variables $X_1, \ldots, X_n$, then the value of $\boldsymbol{\theta}_m$ which maximises $\mathcal{L}(\boldsymbol{\theta}_m)$ is easily found, and

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}, \tag{2.3}$$

where N_{ijk} is the number of instances in which $X_i = x_i^k$ and $\boldsymbol{Pa}_i = \boldsymbol{pa}_i^j$, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. However, if the dataset is incomplete finding the maximum likelihood parameter estimates becomes more difficult. Fortunately there exist a number of techniques which can be used to calculate these maximum likelihood estimates even when we have missing data and there follows a summary of a number of these methods. This summary is by no means exhaustive; it offers only an introduction to some of the more popular techniques in an ever growing catalogue.

2.2 The EM Algorithm

The EM algorithm is a useful and well known tool for estimating model parameters when working with an incomplete dataset. We use $\boldsymbol{\theta}$ to denote the parameters of the model, $\boldsymbol{y}$ and $\boldsymbol{z}$ to denote respectively the observed and unobserved portions of our dataset, and $\boldsymbol{x}$ to denote our complete dataset, so that $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$. This gives us two sample spaces, the completed sample space, $\mathcal{X}$, and the observed sample space, $\mathcal{Y}$, and, defining $\mathcal{X}(\boldsymbol{y})$ to be the set of completions of $\boldsymbol{y}$, that is $\mathcal{X}(\boldsymbol{y}) = \{\boldsymbol{x} = (\boldsymbol{y}', \boldsymbol{z}) : \boldsymbol{y}' = \boldsymbol{y}\}$, we can say that

$$P(\boldsymbol{y} \mid \boldsymbol{\theta}) = \int_{\mathcal{X}(\boldsymbol{y})} P(\boldsymbol{x} \mid \boldsymbol{\theta}).$$

The EM algorithm tackles the problem of solving the incomplete data likelihood equation (2.1) by considering the complete data log-likelihood,

$$\mathcal{L}_c(\boldsymbol{\theta}) = \log P(\boldsymbol{x} \mid \boldsymbol{\theta}). \tag{2.4}$$

This cannot be directly observed since $\boldsymbol{z}$ is unknown, so the EM algorithm adopts a two-stage iterative procedure. The E step consists of determining the expected value of $\log P(\boldsymbol{x} \mid \boldsymbol{\theta})$ given both the observed data $\boldsymbol{y}$ and the current parameter estimates $\boldsymbol{\theta}$. The M step then involves the maximisation of this expected log-likelihood over $\boldsymbol{\theta}$. Effectively we alternate between updating our 'nuisance' parameters, the $\boldsymbol{z}$'s, and the parameters of interest, $\boldsymbol{\theta}$.

For our general Bayesian network consisting of n multivariate nodes $X_1, \ldots, X_n$, the sufficient statistics for calculating the ML estimate for the network parameters is the collection of counts N_{ijk}. If we take

our initial guess at the values of the network parameters to be $\theta_m^0 = \left\{\theta_{ijk}^0 i = 1, \ldots, n; j = 1, \ldots, q_i; k = 1, \ldots, r_i\right\}$ then the EM algorithm can be expressed very simply as

$$\textbf{E Step} \quad \mathbb{E}\left(N_{ijk}^{t+1}\right) = \sum_{l=1}^{N} P\left(x_i^k, pa_i^j \mid y_l, \theta_m^t, m\right)$$

$$\textbf{M Step} \quad \theta_{ijk}^{t+1} = \mathbb{E}\left(N_{ijk}^{t+1}\right) \Big/ \sum_k \mathbb{E}\left(N_{ijk}^{t+1}\right).$$

The EM algorithm has existed in one form or another since at least 1926, but it was the 1977 paper by Dempster, Laird and Rubin [19] which both gave the algorithm its name and demonstrated its full generality. They provided a wide range of examples of the EM algorithm in use, and were able to prove a number of results about its behaviour. They showed that the framework of the algorithm is conceptually simple and perhaps most importantly that each iteration of the algorithm increases the likelihood function, effectively guaranteeing convergence to a stationary point of the likelihood surface.

Since this paper was published much work has been done on the uses and properties of the EM algorithm. Much of this work has centred around the convergence properties of the EM algorithm. Dempster *et al.* were able to show that the rate of convergence of the EM algorithm is linear with rate dependent on the proportion of information about θ in $P(\theta \mid y)$ that is observed. This implies that convergence can be quite slow, particularly if a large proportion of the dataset is missing. In much of the remainder of this chapter we will consider some of the many algorithms that have been proposed which aim to improve on the EM's rate of convergence.

A second problem with the EM algorithm is that, although we are guaranteed to find a stationary point of the likelihood function, we can not be sure that this will be the global maximum. This problem has long been recognised, but for the most part the only solution to this that has enjoyed much success has been the 'multiple re-start' approach, whereby the EM algorithm is run many times from different starting points and the best of the set of end points is assumed to be the global maximum; this approach can be computationally expensive. More recently the Deterministic Annealing EM (DAEM) algorithm [72] has been proposed. This algorithm is a novel adaptation of the EM algorithm using ideas from simulated annealing. It aims to find, if not the global maximum, then at least a better stationary point than might otherwise be found. We consider its application to naive Bayesian networks in the final section of this chapter.

2.3 The 'Incremental' EM Algorithm

In Dempster *et al.*'s 1977 paper on the EM algorithm they referred to variants of this algorithm which they describe as generalised EM (GEM) algorithms. In these algorithms a partial M step is taken in which the new parameter estimates improve the quantity calculated in the E step, but do not necessarily maximise it. Often in missing data problems it also makes sense to implement a partial E step. Generally the unobserved variables will be independent and it can seem sensible to re-estimate the parameters of the model immediately on re-calculating the distribution of just one of these variables. Such variants of the EM algorithm have been investigated for some time, but were only formally justified by Neal and Hinton [52] in their 1998 paper. They view the EM algorithm as maximising

a joint function of the parameters and of the distribution over the unobserved variables. In this way the E and M steps of the algorithm can be seen as maximisations of the function over respectively the unobserved variables and the parameters. Using this viewpoint they are able to argue that any approach which leads to an increase of the likelihood in the E step without necessarily maximising it must be valid. This leads to a number of new EM like algorithms. Neal and Hinton [52] suggest the use of 'sparse' EM algorithms, in which only that part of the distributions for an unobserved variable pertaining to its most likely values is updated in most iterations, or 'winner takes all' EM algorithms, in which, at least initially, the distributions over unobserved variables are restricted to those in which a single value has probability one.

One very simple adaptation of the EM algorithm which follows from this partial E step approach is the 'incremental' EM (IEM) algorithm. This is an adaptation of the EM algorithm in which the distribution of only one of the unobserved variables is updated in each E step. Though this increases the amount of time required for calculation the hope is that convergence will be speeded up as the distributions found for the unobserved variables in each partial E step are utilised immediately to update the network parameter estimates.

Initially a single step of the EM algorithm is carried out to determine the quantities

$$s^0_{l,ijk} = P\left(x_i^k, \boldsymbol{pa}_i^j \mid \boldsymbol{y}_l, \boldsymbol{\theta}^0_m, m\right) \qquad l = 1, \ldots, N \qquad \forall i, j, k$$

where $\boldsymbol{\theta}^0_m$ is our initial guess at the network parameters; after this we iterate between the following two steps until convergence is reached.

E Step Choose an observation, $\boldsymbol{y_l}$

Set $s^{t+1}_{m,ijk} = s^t_{m,ijk} \qquad \forall i,j,k \quad \text{and} \quad m \neq l$

Set $s^{t+1}_{l,ijk} = P\left(x^k_i, \boldsymbol{pa}^j_i \mid \boldsymbol{y_l}, \boldsymbol{\theta}^t_{\boldsymbol{m}}, \boldsymbol{m}\right) \qquad \forall i,j,k$

Calculate $\mathbb{E}\left(N^{t+1}_{ijk}\right) = \sum_l s^{t+1}_{l,ijk}$.

M Step $\theta^{t+1}_{ijk} = \mathbb{E}\left(N^{t+1}_{ijk}\right) \Big/ \sum_k \mathbb{E}\left(N^{t+1}_{ijk}\right)$.

When updating the distributions of the unobserved variables we would normally take each one in turn, but Neal and Hinton also suggest that the performance of the IEM could be further improved by preferentially updating the distributions for the unobserved variables which have yet to stabilise.

When carrying out the E step we can decrease the amount of calculation required by using the identity

$$\mathbb{E}\left(N^{t+1}_{ijk}\right) = \mathbb{E}\left(N^t_{ijk}\right) - s^t_{l,ijk} + s^{t+1}_{l,ijk}. \tag{2.5}$$

However, this means there will be some loss of accuracy due to rounding errors. This error is likely to be inconsequential for smaller models in which convergence takes relatively few iterations, but could be considerable for larger models. This loss of accuracy can be averted by re-calculating the expected sufficient statistics at each iteration of the algorithm, but this would greatly slow down the algorithm. A more sensible approach would be to balance the requirements of speed and accuracy by only carrying out a complete re-calculation periodically.

In Neal and Hinton's paper another similar algorithm first investigated by

Nowlan [53] is mentioned. Nowlan's algorithm differs from the IEM algorithm in that it does not keep strictly accurate sufficient statistics, but instead uses statistics computed as an exponentially decaying average of recently visited data points. Applying this algorithm to the problem in hand gives the following procedure.

Initially a single step of the EM algorithm is carried out to determine starting values for the expected sufficient statistics, $\mathbb{E}\left(N_{ijk}^0\right)$. Then we iterate the following steps:

E Step Choose an observation, y_l

Calculate $s_{l,ijk}^{t+1} = P\left(x_i^k, pa_i^j \mid y_l, \theta_m^t, m\right)$

Estimate $\mathbb{E}\left(N_{ijk}^{t+1}\right)$ by

$\tilde{\mathbb{E}}\left(N_{ijk}^{t+1}\right) = \lambda \cdot \tilde{\mathbb{E}}\left(N_{ijk}^{t}\right) + s_{l,ijk}^{t+1}$,

M Step Set $\theta_{ijk}^{t+1} = \tilde{\mathbb{E}}\left(N_{ijk}^{t+1}\right) \Big/ \sum_k \tilde{\mathbb{E}}\left(N_{ijk}^{t+1}\right)$,

where λ is a decay constant which determines how quickly the algorithm 'forgets' recently visited points.

This algorithm does not converge to the correct answer, as it does not calculate exact sufficient statistics, but Neal and Hinton claim that Nowlan has shown it to provide an effective quick and dirty method for learning in neural networks. This algorithm also has the advantage that unlike the IEM algorithm it does not need to remember the previous distributions of the unobserved parameters. This reduces storage requirements and might make the algorithm more suitable for larger networks.

2.4 Conjugate Gradients

The use of the conjugate gradients (CG) method for function maximisation was first proposed by Fletcher and Reeves [23] in 1964. It is one of a class of maximisation methods in which we start at an initial point $\boldsymbol{\theta}_m^0$ and proceed to the maximum $\hat{\boldsymbol{\theta}}_m$ by a succession of steps. At the t^{th} step we find the next point $\boldsymbol{\theta}_m^{t+1}$ in the series using the relationship

$$\boldsymbol{\theta}_m^{t+1} = \boldsymbol{\theta}_m^t + \alpha^t \boldsymbol{d}^t, \tag{2.6}$$

where $\boldsymbol{d}^t$ is known as the search direction, and α^t is the step length at iteration t. The step length is found by using a line search algorithm to determine the value of $\alpha = \alpha^t$ which maximises

$$\mathcal{L}\left(\boldsymbol{\theta}_m^t + \alpha \boldsymbol{d}^t\right). \tag{2.7}$$

Perhaps the simplest of these methods is the gradient ascent algorithm, illustrated in Figure 2.1, in which we take $\boldsymbol{d}^t = \boldsymbol{g}^t$ where $\boldsymbol{g}^t$ is the gradient of the log-likelihood at the point $\boldsymbol{\theta}_m^t$. This method can be proved to converge to a local maximum of the log-likelihood, but convergence can be very slow.

One way in which convergence can be speeded up is to instead select the $\boldsymbol{d}^t$ in such a way that they are conjugate. This conjugacy means that when we maximise $\mathcal{L}(\boldsymbol{\theta}_m)$ with respect to $\boldsymbol{d}^t$ we are also ensuring that $\mathcal{L}(\boldsymbol{\theta}_m)$ is still maximised with respect to $\boldsymbol{d}^0, \ldots, \boldsymbol{d}^{t-1}$. We can ensure the conjugacy of our search directions by using a predetermined set of $\delta = \dim(\Theta_m)$ search directions.

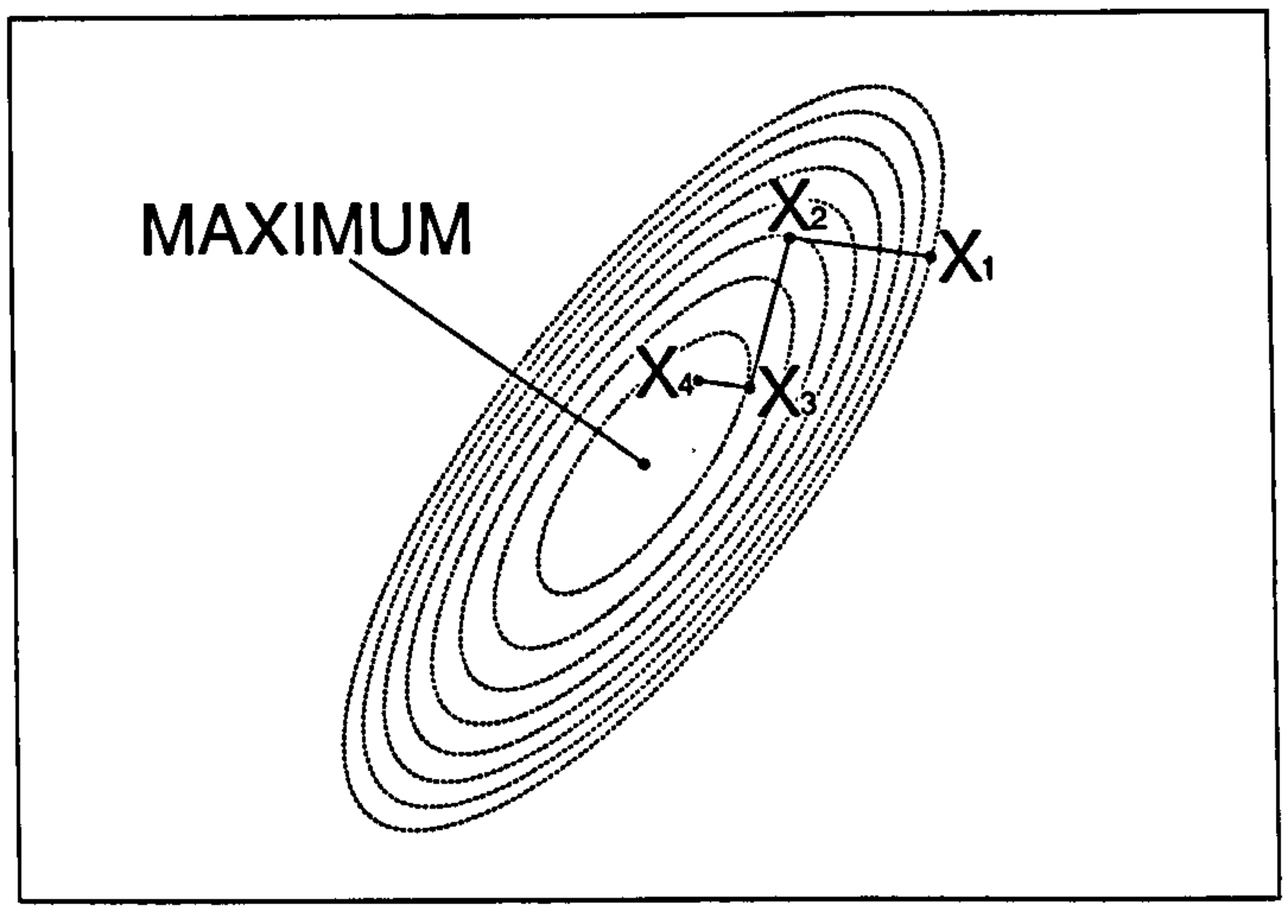


Figure 2.1: Gradient ascent in a simple two dimensional maximisation problem. From the initial point we calculate the gradient and search in that direction for a maximum, we then recalculate the gradient and begin to search in the new direction. This continues until we reach a point at which the gradient is zero, a local maximum.

One very simple example of this would be

$$
\begin{array}{ccccccccccc}
(& 1 & , & 0 & , & 0 & , & \ldots & , & 0 &) \\
(& 0 & , & 1 & , & 0 & , & \ldots & , & 0 &) \\
(& 0 & , & 0 & , & 1 & , & \ldots & , & 0 &) \\
 & \vdots & & \vdots & & \vdots & & & & \vdots & \\
(& 0 & , & 0 & , & 0 & , & \ldots & , & 1 &)
\end{array}
$$

We maximise $\mathcal{L}(\theta_m)$ along the directions d^0 up to $d^{\delta-1}$ successively, and then restart the algorithm until we have convergence. This method may be improved upon further by using the Conjugate Gradient algorithm in which, rather than

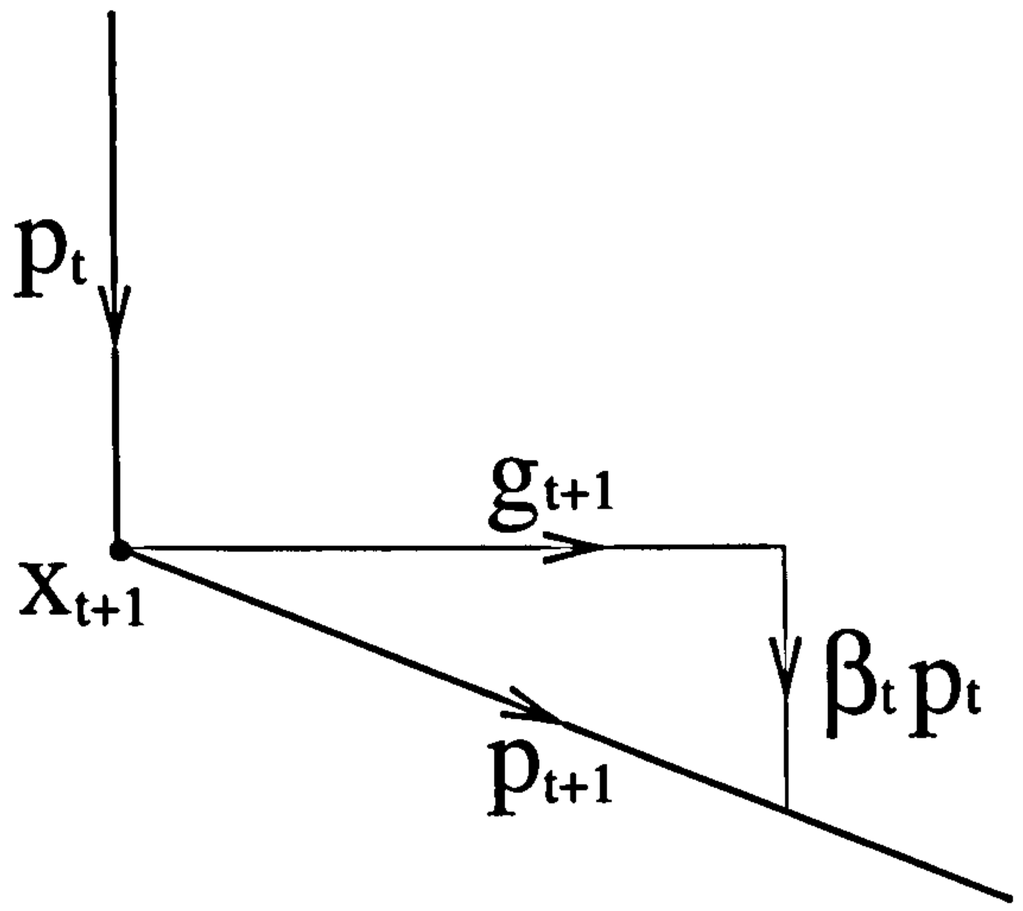


Figure 2.2: Selecting a new search direction in the conjugate gradients algorithm. Each search direction is made up of the gradient at the current point plus a contribution from all previous search directions.

using a predetermined set of search directions, we select a new search direction at each stage based upon the gradient of $\mathcal{L}(\boldsymbol{\theta}_m)$ evaluated at the current parameter estimates, and our previous search directions. So our search direction at iteration $t+1$ is given by

$$\boldsymbol{d}^{t+1} = \boldsymbol{g}^{t+1} + \beta^t \boldsymbol{\theta}^t,$$

where β^t can be defined in a number of ways, including

$$\begin{aligned}
\text{(i)} \quad \beta^t &= -\left(\boldsymbol{g}^{t+1^T}\boldsymbol{g}^{t+1}\right) / \left(\boldsymbol{g}^{t^T}\boldsymbol{g}^t\right), \\
\text{(ii)} \quad \beta^t &= \left(\boldsymbol{g}^{t+1^T}\boldsymbol{g}^{t+1}\right) / \left(\boldsymbol{d}^{t^T}\boldsymbol{g}^t\right),
\end{aligned}$$

or

$$\text{(iii)} \quad \beta^t = \boldsymbol{g}^{t+1^T}\left(\boldsymbol{g}^t - \boldsymbol{g}^{t+1}\right) / \left(\boldsymbol{g}^{t^T}\left(\boldsymbol{g}^t - \boldsymbol{g}^{t+1}\right)\right) \tag{2.8}$$

where (i) is the value for β originally suggested by Fletcher. To see that the set of

search directions produced by these β's are orthogonal consider that, if we begin with the search direction $\boldsymbol{d}^0 = \boldsymbol{g}^0$, it then follows that

$$\boldsymbol{g}^{1T}\boldsymbol{d}^0 = 0 = \boldsymbol{g}^{1T}\boldsymbol{g}^0$$

so $\boldsymbol{g}^0$ and $\boldsymbol{g}^1$ are orthogonal. We then prove the conjugacy of all of the $\boldsymbol{g}$'s by induction. If we suppose that $\boldsymbol{g}^0, \ldots, \boldsymbol{g}^t$ are conjugate and that $\boldsymbol{d}^0, \ldots, \boldsymbol{d}^t$ are each a function of them then the $\boldsymbol{d}$'s must also be conjugate. Since $\boldsymbol{g}^{t+1}$ is to be determined at a point which is a maximum in the direction $\boldsymbol{d}^t$ it must follow that $\boldsymbol{g}^{t+1}$ is orthogonal to $\boldsymbol{d}^t$ and hence to all previous $\boldsymbol{g}$'s.

The most important advantage of deriving our search directions as the algorithm progresses rather than using a predetermined set of search directions is that this method will generally make good uniform progress towards the maximum, whereas when using predetermined search directions progress could well be slight until the final few steps.

Using Fletcher's value for β, the CG algorithm proceeds as follows. Starting from an initial point $\boldsymbol{\theta}_m^0 \in \Theta_m$ we take $\boldsymbol{d}^0 = \boldsymbol{g}^0$ and iterate between the following two steps:

Step 1 Find α^t, the value of α that maximises

$$\mathcal{L}\left(\boldsymbol{\theta}_m^t + \alpha \boldsymbol{d}^t\right)$$

Set $\boldsymbol{\theta}_m^{t+1} = \boldsymbol{\theta}_m^t + \alpha^t \boldsymbol{d}^t$;

Step 2 set $\boldsymbol{d}^{t+1} = \boldsymbol{g}^{t+1} + \beta^t \boldsymbol{d}^t$,

where $\beta^t = \left(\boldsymbol{g}^{t+1T}\boldsymbol{g}^{t+1}\right) \Big/ \left(\boldsymbol{d}^{tT}\boldsymbol{g}^t\right)$.

This algorithm should be restarted every δ steps.

The conjugate gradient algorithm is only guaranteed to converge to a local maximum of $\mathcal{L}(\theta_m)$ if started sufficiently close to that maximum. As such it is common practice to begin with a few steps of the EM algorithm and then as the neighbourhood of the local maximum is approached switch to the faster conjugate gradients algorithm.

2.5 Generalised Conjugate Gradients

The performance of algorithms that utilise gradient information can often be improved by the use of 'generalised gradients'. Theisson [68] outlines the application of a generalised conjugate gradient (GCG) method first described by Jamshidian and Jennrich [37] to Bayesian networks.

If we consider the generalised norm

$$\|\theta\| = (\theta^T W \theta)^{\frac{1}{2}},$$

defined by a positive definite matrix W, then the generalised gradient of $\mathcal{L}(\theta_m)$, $\tilde{g}$ is given by

$$\tilde{g} = W^{-1} g. \tag{2.9}$$

The generalised conjugate gradient algorithm proceeds as follows:

Starting from an initial point $\theta_m^0 \in \Theta_m$ we take $d^0 = \tilde{g}^0$, and iterate between the following two steps:

Step 1 Find α^t, the value of α

that maximises $\mathcal{L}\left(\boldsymbol{\theta}_m^t + \alpha \boldsymbol{d}^t\right)$

Set $\boldsymbol{\theta}_m^{t+1} = \boldsymbol{\theta}_m^t + \alpha^t \boldsymbol{d}^t$

Step 2 set $\boldsymbol{d}^{t+1} = \tilde{\boldsymbol{g}}^{t+1} + \beta^t \boldsymbol{d}^t$

where $\beta^t = \tilde{\boldsymbol{g}}^T \left(\boldsymbol{g}^{t+1} - \boldsymbol{g}^t\right) \Big/ \boldsymbol{g}^{tT} \left(\boldsymbol{g}^{t+1} - \boldsymbol{g}^t\right)$.

As with the method of conjugate gradients it is necessary to restart this algorithm every δ steps, and convergence is only guaranteed to a local maximum if started in the neighbourhood of that maximum. Jamshidian and Jennrich suggest that a good point to switch from the EM to the GCG algorithm is when

$$2\mathcal{L}\left(\boldsymbol{\theta}_m^{t+1}\right) - 2\mathcal{L}\left(\boldsymbol{\theta}_m^t\right) < 1; \tag{2.10}$$

that is, to switch to the GCG algorithm once the χ^2 statistic for testing the equality of two successive parameter estimates falls below 1.

It is important that the matrix W which defines the generalised norm is well chosen. Jamshidian and Jennrich show that a good choice for W, that leads to a particularly simple approximation to the value of $\tilde{\boldsymbol{g}}$, is

$$W = -\ddot{Q}\left(\hat{\boldsymbol{\theta}}_m, \hat{\boldsymbol{\theta}}_m\right) \tag{2.11}$$

where $Q\left(\boldsymbol{\theta}'_m, \boldsymbol{\theta}^t_m,\right) = \mathbb{E}\left[\mathcal{L}_c\left(\boldsymbol{\theta}'_m\right) \mid \boldsymbol{y}, \boldsymbol{\theta}^t_m\right]$ may be viewed as a local approximation to $\mathcal{L}\left(\boldsymbol{\theta}'_m\right)$ in the neighbourhood of $\boldsymbol{\theta}^t_m$ (see Dempster *et al* [19]) and $\ddot{Q}\left(\hat{\boldsymbol{\theta}}_m, \hat{\boldsymbol{\theta}}_m\right)$ is the Hessian of $Q\left(\boldsymbol{\theta}'_m, \boldsymbol{\theta}^t_m\right)$ viewed as a function of $\boldsymbol{\theta}'_m$ and

evaluated at $\left(\theta'_m, \theta^t_m\right) = \left(\hat{\theta}_m, \hat{\theta}_m\right)$. Then, if $\hat{\theta}_m$ is an interior point of Θ_m, it follows that

$$\tilde{\theta}_m - \theta^t_m = -\left(\ddot{Q}\left(\hat{\theta}_m, \hat{\theta}_m\right)\right)^{-1} g^t + o\left(\theta^t_m - \hat{\theta}_m\right), \tag{2.12}$$

where $\tilde{\theta}_m$ is the value of θ'_m that maximises $Q\left(\theta'_m, \theta^t_m\right)$. Hence using equations 2.9, 2.11 and 2.12 we can make the approximation

$$\tilde{g} \approx \tilde{\theta}_m - \theta^t_m. \tag{2.13}$$

This means that a good approximation to the generalised gradient can be easily computed by simply carrying out an EM step.

2.6 Helmbold's Methods

2.6.1 Introduction

Helmbold *et al.* [34] adapted a framework developed for supervised learning in neural networks to produce a number of iterative algorithms for finding the proportion vector which maximises the likelihood of a given sample for a mixture of given densities. It is a fairly straightforward matter to adapt their procedure to suit other models, such as the one considered here.

At step t of the algorithm we consider a Taylor expansion of $\mathcal{L}\left(\theta_m\right)$ about our current parameter estimates, θ^t_m, which is

$$\mathcal{L}\left(\theta_m\right) \approx \mathcal{L}\left(\theta^t_m\right) + \Delta\mathcal{L}\left(\theta^t_m\right)^T \left(\theta_m - \theta^t_m\right). \tag{2.14}$$

We could select $\theta_m = \theta_m^{t+1}$ to maximise this approximation, but the approximation degrades as we move further from θ_m^t. To compensate for this we introduce a penalty term, $d\left(\theta_m, \theta_m^t\right)$, where d is a non-negative function measuring the distance between θ_m and θ_m^t. Hence we actually choose θ_m^{t+1} to be the value of θ_m that maximises

$$F\left(\theta_m\right) = \eta\left(\mathcal{L}\left(\theta_m^t\right) + \Delta\mathcal{L}\left(\theta_m^t\right)^T\left(\theta_m - \theta_m^t\right)\right) - d\left(\theta_m, \theta_m^t\right), \qquad (2.15)$$

where η is a positive parameter called the 'learning rate' and governs the relative importance of our penalty term to our approximation of the log-likelihood. It takes into account our need to stay near our current estimate in order that we remain within the region in which our approximation to the likelihood surface is relatively good, and our need to move quickly away from our current estimate in order that a reasonable rate of convergence is achieved.

In order to maximise this function subject to the constraints, Lagrange multipliers are introduced, and it is noted that $\mathcal{L}\left(\theta_m^t\right) + \Delta\mathcal{L}\left(\theta_m^t\right)^T\theta_m^t$ is independent of θ_m, so maximising F subject to the constraints is equivalent to maximising

$$F'\left(\theta_m, \gamma\right) = \eta\Delta\mathcal{L}\left(\theta_m^t\right)^T\theta_m - d\left(\theta_m, \theta_m^t\right) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}\gamma_{ij}\left(\sum_{k=1}^{r_i}\theta_{ijk} - 1\right), \quad (2.16)$$

where the γ_{ij} are Lagrange multipliers. We carry out this maximisation by setting the $\sum_{i=1}^{n} q_i\left(r_i - 1\right)$ partial derivatives to zero, and enforcing the additional constraints

$$\sum_{k=1}^{r_i}\theta_{ijk}^{t+1} = 1 \quad \forall i, j$$

The algorithm updates are then found by solving this system of equations and plugging in different distance metrics. The three considered here are the Euclidean, relative entropy and chi-squared metrics

$$
\begin{aligned}
d_{EUC}(\boldsymbol{u} \parallel \boldsymbol{v}) &= \frac{1}{2} \parallel \boldsymbol{u} - \boldsymbol{v} \parallel_2^2 = \frac{1}{2} \sum_{i=1}^{N} (u_i - v_i)^2 , \\
d_{RE}(\boldsymbol{u} \parallel \boldsymbol{v}) &= \sum_{i=1}^{N} u_i \ln \frac{u_i}{v_i}, \\
d_{\chi^2}(\boldsymbol{u} \parallel \boldsymbol{v}) &= \frac{1}{2} \sum_{i=1}^{N} \frac{(u_i - v_i)^2}{v_i}.
\end{aligned}
$$

This results in three update rules

- The GP_η update is

$$\theta_{ijk}^{t+1} = \theta_{ijk}^{t} + \eta \left(\frac{\partial \mathcal{L}(\theta_m^t)}{\partial \theta_{ijk}^t} - \frac{1}{r_i} \sum_{l=1}^{r_i} \frac{\partial \mathcal{L}(\theta_m^t)}{\partial \theta_{ijl}^t} \right) \tag{2.17}$$

- The EG_η update is

$$\theta_{ijk}^{t+1} = \theta_{ij1}^{t} \exp\left(\eta \frac{\partial \mathcal{L}(\theta_m^t)}{\partial \theta_{ijk}^t} \right) \Big/ \sum_{l=1}^{r_i} \theta_{ijl}^{t} \exp\left(\eta \frac{\partial \mathcal{L}(\theta_m^t)}{\partial \theta_{ijl}^t} \right) \tag{2.18}$$

- The EM_η update is

$$\theta_{ijk}^{t+1} = \theta_{ijk} \left(\eta \left(\frac{\partial \mathcal{L}(\theta_m^t)}{\partial \theta_{ijk}^t} - \sum_{l=1}^{r_i} \theta_{ijl}^{t} \frac{\partial \mathcal{L}(\theta_m^t)}{\partial \theta_{ijl}^t} \right) + 1 \right) \tag{2.19}$$

Helmbold *et al.* observe that for the problem they considered, namely determining the distribution of the latent classes for a mixture of known distributions, the EM algorithm is a special case of the EM_η algorithm. We

can see from the formula given above for the EM_η update that this will only hold true if

$$\sum_{l=1}^{R_I} \theta_{ijl}^t \frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial \theta_{ijl}^t} = 1, \tag{2.20}$$

and this will only be the case if $\boldsymbol{Pa_i} = \emptyset$, which is true when we are only estimating the proportion vector of a naive network, but which will not be true for more general problems.

The EM_η algorithm in the form derived by Helmbold *et al.* is identical to the EM acceleration method suggested by Peters and Walker [54] in 1978, although Peters and Walker derive their algorithm in a different fashion. They note that when applying the EM algorithm to the problem of estimating the proportion vector for a mixture of known distributions the point to which the EM algorithm converges satisfies

$$\theta_{ijk}^{t+1} = \theta_{ijk}^t \frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial \theta_{ijk}^t}. \tag{2.21}$$

We can equivalently write

$$\theta_{ijk}^{t+1} = (1-\epsilon)\,\theta_{ijk}^t + \epsilon \theta_{ijk}^t \frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial \theta_{ijk}^t}, \tag{2.22}$$

where (2.22) equals (2.21) for $\epsilon = 1$. We can rearrange (2.22) in order to obtain 2.19, with ϵ substituting for η. Peters and Walker prove a number of results about this algorithm. They prove that convergence is guaranteed for $0 < \epsilon < 2$, and they further show that the optimal choice for ϵ, that is the choice leading to the fastest convergence, must be greater than 1 and can be greater than 2 even though they are unable to guarantee convergence for $\epsilon \geq 2$.

Helmbold et al's algorithms also bear some relation to Green's One Step Late

EM (OSLEM) Algorithm [30]. Green's OSLEM algorithm comes from his work on maximum penalised likelihood estimation, in which the model parameters, $\boldsymbol{\theta}$, are estimated by $\tilde{\boldsymbol{\theta}}$ which is chosen to maximise

$$\mathcal{L}(\boldsymbol{\theta}) - \lambda J(\boldsymbol{\theta}).$$

Depending upon our point of view, we can either regard $\exp\{-\lambda J(\boldsymbol{\theta})\}$ as being proportional to a prior for $\boldsymbol{\theta}$, or regard $J(\boldsymbol{\theta})$ as a roughness function and λ as a smoothing parameter. In order to make the comparison we need to interpret Helmbold *et al*'s methods as attempting to find $\hat{\boldsymbol{\theta}}$ to maximise

$$F_1(\boldsymbol{\theta}) + \eta F_2(\boldsymbol{\theta}, \boldsymbol{\theta}),$$

where

$$\begin{aligned} F_1(\boldsymbol{\theta}) &= \mathcal{L}(\boldsymbol{\theta}) \\ F_2(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2) &= \text{cross-entropy measure with maximum} \\ &\quad \text{value zero, when } \boldsymbol{\theta}^1 = \boldsymbol{\theta}^2. \end{aligned}$$

In each iterative step we are then looking for $\boldsymbol{\theta}^{t+1}$ to maximise

$$F_3(\boldsymbol{\theta}^{t+1}) = F_1(\boldsymbol{\theta}^{t+1}) + \eta F_2(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t)$$

However, rather than finding our updated parameter estimates by solving

$$\frac{\partial F_3(\boldsymbol{\theta}^{t+1})}{\partial \boldsymbol{\theta}^{t+1}} = 0$$

for $\boldsymbol{\theta}^{t+1}$, we instead consider the easier problem

$$\frac{\partial F_1\left(\boldsymbol{\theta}^t\right)}{\partial \boldsymbol{\theta}^t}+\eta \frac{\partial F_2\left(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t\right)}{\partial \boldsymbol{\theta}^{t+1}}=0, \tag{2.23}$$

arguing that $\frac{\partial \mathcal{L}\left(\boldsymbol{\theta}^t\right)}{\partial \boldsymbol{\theta}^t}$ will be a good approximation for $\frac{\partial \mathcal{L}\left(\boldsymbol{\theta}^{t+1}\right)}{\partial \boldsymbol{\theta}^{t+1}}$. Similarly, in OSLEM we are looking for $\boldsymbol{\theta}^{t+1}$ to maximise

$$Q\left(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t\right)-\lambda J\left(\boldsymbol{\theta}^{t+1}\right),$$

where $Q\left(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t\right)=\mathbb{E}\left[\mathcal{L}\left(\boldsymbol{\theta}^{t+1}\right) \mid \boldsymbol{y}, \boldsymbol{\theta}^t\right]$ (see Dempster *et al.* [19]). This is equivalent to finding $\boldsymbol{\theta}^{t+1}$ to maximise

$$F_3\left(\boldsymbol{\theta}^{t+1}\right)=F_1\left(\boldsymbol{\theta}^{t+1}\right)+\eta F_2\left(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t\right),$$

where

$$\begin{aligned} F_1\left(\boldsymbol{\theta}^{t+1}\right) &= J\left(\boldsymbol{\theta}^{t+1}\right) \\ F_2\left(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t\right) &= Q\left(\boldsymbol{\theta}^{t+1}, \boldsymbol{\theta}^t\right) \\ \eta &= \frac{1}{\lambda}. \end{aligned}$$

However, as before, rather than solving $\frac{\partial F_3\left(\boldsymbol{\theta}^{t+1}\right)}{\partial \boldsymbol{\theta}^{t+1}}=0$ we use the same argument to justify solving the easier expression given in (2.23). Although viewing these two approaches in this framework demonstrates considerable similarities between them, it is interesting to note that the log-likelihood $\mathcal{L}(\boldsymbol{\theta})$ influences 'F_1' in Helmbold *et al.*'s approach and 'F_2' in OSLEM.

2.7 Approximate Conjugate Gradients (ACG)

The ACG algorithm is a simple iterative algorithm based upon the method of conjugate gradients. It uses ideas from Helmbold *et al.*'s paper to replace the line search to find the optimum step length at each iteration with a simple calculation which approximates this quantity.

Considering the conjugate gradient algorithm, the first step is to select the search direction. Denoting the search direction at iteration t by $\boldsymbol{d}^t$ we use the scheme

$$\begin{aligned} \boldsymbol{d}^0 &= \boldsymbol{g}^0 \\ \boldsymbol{d}^t &= \boldsymbol{g}^t + \beta^{t-1}\boldsymbol{d}^{t-1} \qquad t > 0, \end{aligned}$$

where

$$\boldsymbol{g}^t = \frac{\partial}{\partial \theta_{ijk}} \mathcal{L}\left(\boldsymbol{\theta}_m\right) \qquad \text{and} \qquad \beta^{t-1} = \left(\boldsymbol{g}^{tT}\boldsymbol{g}^t\right) \Big/ \left(\boldsymbol{d}^{t-1T}\boldsymbol{g}^{t-1}\right).$$

We therefore update our parameter estimates $\boldsymbol{\theta}_m^t$ according to

$$\boldsymbol{\theta}_m^{t+1} = \boldsymbol{\theta}_m^t + \alpha^t \boldsymbol{d}^t,$$

where α^t is our step length at iteration t. In the conjugate gradients algorithm we would perform a line search to determine α^t, the value of α maximising $\mathcal{L}\left(\boldsymbol{\theta}_m^t + \alpha \boldsymbol{d}^t\right)$. In the new algorithm we use Helmbold *et al.*'s idea of maximising the Taylor expansion of the log-likelihood subject to a penalty term, the hope

being that this will give a reasonable approximation to the optimal step length for considerably less computational effort. Hence in the second step of our algorithm we must chose $\alpha = \alpha^t$ to maximise the quantity

$$\eta g^{tT} \theta_m^{t+1} - d\left(\theta_m^{t+1}, \theta_m^t\right),$$

where η is the 'learning parameter' and is used to balance the relative importance of the need to increase the log-likelihood and the need not to stray too far from the current parameter estimates. So in effect we need to choose $\alpha = \alpha^t$ to maximise

$$\eta g^{tT} \theta_m^t + \eta \alpha g^{tT} d^t - d\left(\theta_m^t + \alpha^t d^t, \theta_m^t\right). \tag{2.24}$$

Depending on which metric we choose to use for our penalty term d we get a different update rule[1]:

$$\begin{aligned} \text{Euclidean Metric} &\Rightarrow \alpha^t = \eta \left(g^{tT} d^t\right) / \left(d^{tT} d^t\right) \\ \chi^2 \text{ Metric} &\Rightarrow \alpha^t = \eta \left(g^{tT} d^t\right) / \left(\textstyle\sum_i \frac{\left(d_i^t\right)^2}{\theta_i^t}\right). \end{aligned}$$

The problem with additive algorithms such as this is that it is all too easy to stray outside the parameter space; in this case we avoid that possibility by keeping the value of η small, though of course this is likely to affect the rate of convergence adversely. One possible way of avoiding this problem is to re-parameterise our model. If we use the logit transformation

$$\phi_{ijk} = \log\left(\frac{\theta_{ijk}}{\theta_{ij1}}\right), \qquad \text{for } k = 2, \ldots, r_i,$$

[1]The Relative Entropy metric is not used here as it does not result in a simple closed form for α^t.

then in the new parameter space denoted by ϕ_m no restriction need be imposed on the ϕ_{ijk} in order to ensure that the point to which the algorithm converges satisfies the necessary requirements for the corresponding θ_{ijk} to be probabilities. This means that we are able to carry out an unconstrained maximisation of the transformed log-likelihood, denoted by $\mathcal{L}_\phi$, which in turn implies that we may be able to make use of larger values of η. One problem with this parameterisation is that, when using the update rule derived from the χ^2 metric, we might find that the value assigned to α^t corresponds to a minimum of the approximation to the likelihood surface rather than a maximum. This can be seen by differentiating (2.24) twice with respect to α; the result is $\sum_i \frac{d_i^{t\,2}}{\phi_i^t}$, which can be positive or negative depending upon the current value of ϕ^t. We use ACG_θ and ACG_θ^χ to denote the update rules derived using respectively the Euclidean and χ^2 metrics in our untransformed parameter space, and ACG_ϕ to denote the update derived after applying the logit transformation when using the Euclidean metric.

Hence ACG_θ is the simple iterative algorithm

Step 1 $\boldsymbol{d}^0 = \boldsymbol{g}^0$

$$\boldsymbol{d}^t = \boldsymbol{g}^t + \left(\left(\boldsymbol{g}^{tT}\boldsymbol{g}^t\right) / \left(\boldsymbol{d}^{t-1T}\boldsymbol{g}^{t-1}\right)\right)^T \boldsymbol{d}^{t-1} \qquad \forall t > 0$$

Step 2 $\boldsymbol{\theta}_m^{t+1} = \boldsymbol{\theta}_m^t + \alpha^t \boldsymbol{d}^t$

where

$$\alpha^t = \eta \left(\boldsymbol{g}^{tT}\boldsymbol{d}^t\right) / \left(\boldsymbol{d}^{tT}\boldsymbol{d}^t\right)$$

As with the standard conjugate gradients algorithm, this algorithm should be reinitialised every δ steps. The ACG_θ^χ and ACG_ϕ algorithms are similarly simple.

2.7.1 Convergence of the ACG Algorithm

We were unable to produce a complete proof of the convergence of the ACG_θ algorithm, instead we offer here a partial proof of the algorithm's convergence, when applied to maximising a quadratic problem, which will illustrate some of the issues that must be dealt with in order to produce a definitive proof of convergence.

Quadratic Problem

Consider the function

$$f(\theta) = -\frac{1}{2}\theta^T Q\theta + \theta^T b \tag{2.25}$$

where Q is a positive definite and symmetric $n \times n$ matrix. This function, $f(\theta)$, is quadratic with a single maximum, θ^*, satisfying $Q\theta^* = b$. If we apply the iterative ACG_θ algorithm to this problem from a starting point θ^0 then at iterations t and $t+1$ we will find ourselves at points θ^t and θ^{t+1} respectively. Substituting $\theta = \theta^{t+1}$ into Equation 2.25 and using the fact that in the ACG_θ algorithm $\theta^{t+1} = \theta^t + \alpha^t d^t$ and also that for this particular function $g^t = \frac{\partial f(\theta)}{\partial \theta} = -Q\theta + b$, we can see that

$$\begin{aligned}
f(\theta^{t+1}) &= -\frac{1}{2}\theta^{t+1^T} Q\theta^{t+1} + \theta^{t+1^T} b \\
&= -\frac{1}{2}\left(\theta^t + \alpha^t d^t\right)^T Q\left(\theta^t + \alpha^t d^t\right) + \left(\theta^t + \alpha^t d^t\right)^T b \\
&= -\frac{1}{2}\theta^{t^T} Q\theta^t - \frac{1}{2}\alpha^t d^{t^T} Q\theta^t - \frac{1}{2}\alpha^t \theta^{t^T} Qd^t - \frac{1}{2}\alpha^{t^2} d^{t^T} Qd^t + \theta^{t^T} b + \alpha^t d^{t^T} b \\
&= f(\theta^t) - \frac{1}{2}\alpha^t d^{t^T} Q\theta^t - \frac{1}{2}\alpha^t \theta^{t^T} Qd^t - \frac{1}{2}\alpha^{t^2} d^{t^T} Qd^t + \alpha^t d^{t^T} b \\
&= f(\theta^t) - \alpha^t d^{t^T} Q\theta^t - \frac{1}{2}\alpha^{t^2} d^{t^T} Qd^t + \alpha^t d^{t^T} b \\
&= f(\theta^t) + \alpha^t d^{t^T}\left(-Q\theta^t + b\right) - \frac{1}{2}\alpha^{t^2} d^{t^T} Qd^t
\end{aligned}$$

$$= f\left(\boldsymbol{\theta}^t\right) + \alpha^t \boldsymbol{d}^{tT} \boldsymbol{g}^t - \frac{1}{2}\alpha^{t2} \boldsymbol{d}^{tT} Q \boldsymbol{d}^t$$

The implication of this is that, if the step length at iteration t, α^t, satisfies

$$\alpha^t \boldsymbol{d}^{tT} \boldsymbol{g}^t - \frac{1}{2}\alpha^{t2} \boldsymbol{d}^{tT} Q \boldsymbol{d}^t \geq 0,$$

then it must follow that $f\left(\boldsymbol{\theta}^{t+1}\right) \geq f\left(\boldsymbol{\theta}^t\right)$ and hence we can expect to converge to the maximum point of $f(\boldsymbol{\theta})$. In the $\text{ACG}_{\boldsymbol{\theta}}$ algorithm we use

$$\alpha^t = \eta \left(\boldsymbol{g}^{tT} \boldsymbol{d}^t\right) / \left(\boldsymbol{d}^{tT} \boldsymbol{d}^t\right),$$

and hence to demonstrate convergence we must show that

$$\eta \left(\boldsymbol{g}^{tT} \boldsymbol{d}^t\right) / \left(\boldsymbol{d}^{tT} \boldsymbol{d}^t\right) \boldsymbol{d}^{tT} \boldsymbol{g}^t - \frac{1}{2}\eta^2 \left(\left(\boldsymbol{g}^{tT} \boldsymbol{d}^t\right) / \left(\boldsymbol{d}^{tT} \boldsymbol{d}^t\right)\right)^2 \boldsymbol{d}^{tT} Q \boldsymbol{d}^t \geq 0$$

for some value of the learning parameter, η. Since $\left(\boldsymbol{d}^{tT} \boldsymbol{d}^t\right)$ and $\left(\boldsymbol{g}^{tT} \boldsymbol{d}^t\right)^2$ will both be positive we can simplify this expression to give

$$\eta \boldsymbol{d}^{tT} \boldsymbol{d}^t - \frac{1}{2}\eta^2 \boldsymbol{d}^{tT} Q \boldsymbol{d}^t \geq 0.$$

Assuming η is positive we are then able to state that convergence will occur provided

$$\eta \leq 2 \frac{\boldsymbol{d}^{tT} \boldsymbol{d}^t}{\boldsymbol{d}^{tT} Q \boldsymbol{d}^t} \qquad \forall t.$$

Unfortunately the fact that $\boldsymbol{d}^t$ depends upon $\left\{\boldsymbol{d}^{t-1}, \ldots, \boldsymbol{d}^0\right\}$ makes it difficult to use this to place an upper bound on the value of η that would lead to convergence, but the implication here is that we can cause the algorithm to converge by making

η sufficiently small.

2.8 A Comparison of Parameter Estimation Algorithms

We will consider the application of the algorithms previously described to parameter estimation in naive Bayesian networks, such as that shown in Figure 2.3. More complete details of the implementation of these algorithms can be found in Appendix A. When comparing the performance of these algorithms a number

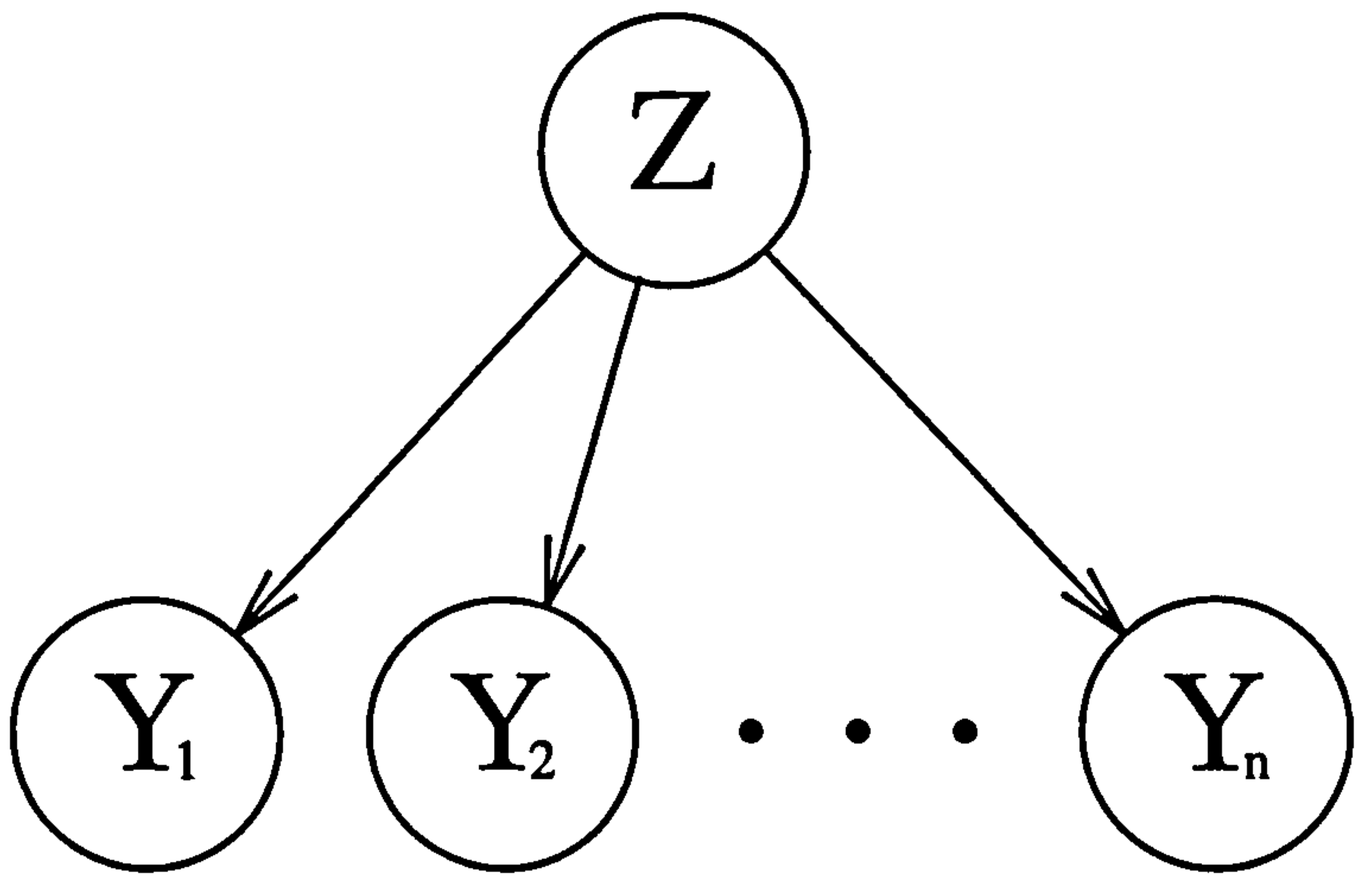


Figure 2.3: A naive Bayesian network. The nodes $Y_1, \ldots, Y_n$ are fully observable while the root node, Z, is unobserved.

of factors need to be considered. Obviously we wish to compare these algorithms on a 'per iteration' basis, but this is only part of the story. Other factors such as how long each iteration of the algorithm takes, and the computing resources that different algorithms will require, must also be considered. Finally, we must also pay attention to simplicity of implementation. This is difficult to quantify but nevertheless important; we do not wish to spend a great deal of time coding and

testing a more sophisticated algorithm if the pay back, in terms of the improved performance of the algorithm, does not warrant our investment.

Mixture distribution problems generally fall into one of two categories, those in which we know the distribution of the components of the mixture and need only to find the proportion vector in order to fully define the mixture distribution, and those in which the distribution of each component is also unknown. We will consider these two cases separately. Initially we will simply compare the algorithms on a per iteration basis; later we will attempt to bring to bear some of the other factors under consideration.

2.8.1 Mixtures of Known Components: The Proportion Vector Problem

A generative naive Bayesian network in which the root variable Z had $r_z = 4$ states and with $n = 5$ observable variables, $Y_1, \ldots, Y_5$, each having $r_i = 2$ possible outcomes was set up, the parameters of this network being drawn from a uniform distribution. A dataset of size 100 was then generated from this network. To simulate data from such a network we first draw the state of the variable Z from its distribution then draw values of each of the Y_i according to their distribution given the value of Z. We then discard all observations of Z to give a dataset in which we have no information about latent class membership. We then fitted a naive Bayesian network to the artificial dataset using each of the algorithms previously described. In fitting the model we assumed that the number of states of the variable Z was known and that the true parameters of each component of the mixture were also known, so that all that remained to be estimated was the

proportion vector. This procedure was carried out 20 times and the performance of each algorithm in estimating the proportion vector was compared to that of the EM algorithm, the results being shown in Figures 2.4 to 2.13. It should be noted that both the CG and GCG algorithms start with a few iterations of the EM algorithm, the change over from the EM algorithm taking place once the criterion suggested by Jamshidian and Jennrich [37] given in Equation 2.10 is satisfied. In order to produce the plots detailing the comparison of the algorithms the following procedure was adopted. We denote the value of the log-likelihood for the baseline algorithm at iteration t by $\mathcal{L}_1^t$ and of the competing algorithm by $\mathcal{L}_2^t$, and take $\mathcal{L}_1^0 = \mathcal{L}_2^0$ to be the log-likelihood at the starting point for each algorithm and $\mathcal{L}_1^\infty = \mathcal{L}_2^\infty$ to be the log-likelihood at the point to which both of the algorithms converge (in these examples the algorithms all converged to the same point, more generally this may not be the case). Then we begin by rescaling them so that

$$\mathcal{L}_{*1}^t = \frac{\mathcal{L}_1^t - \mathcal{L}_1^0}{\mathcal{L}_1^\infty - \mathcal{L}_1^0} \qquad \mathcal{L}_{*2}^t = \frac{\mathcal{L}_2^t - \mathcal{L}_2^0}{\mathcal{L}_2^\infty - \mathcal{L}_2^0},$$

so now $\mathcal{L}_{*i}^0 = 0$, $\mathcal{L}_{*i}^\infty = 1$ and $\mathcal{L}_{*i}^t \in [0,1]$ for all t, $i = 1,2$. The point plotted is then $\mathcal{L}_{*2}^t - \mathcal{L}_{*1}^t$, so it is positive if the competing algorithm is closer to convergence than our baseline algorithm and negative if the opposite is true. In each case we plotted all 20 cases in dotted lines and an average of the 20 cases is shown by the heavy solid line.

For the EM_η, EG_η, GP_η, ACG_ϕ , ACG_θ and ACG_θ^χ algorithms we must pay some attention to how the learning parameter will be selected. Unfortunately there is no easy answer to this and the only way to select the value of the learning parameter was to spend some time experimenting for each example to find a value

that gave good results. Obviously this can be quite time consuming. Table 2.1 gives summary statistics for the 20 different learning parameters used for each of these algorithms.

Algorithm	Learning Parameter			
	Mean	Minimum	Maximum	Standard Error
EM_η	1.88	1.4	2.2	0.24
EG_η	2.48	1.5	7.0	1.15
GP_η	0.44	0.002	1.3	0.28
ACG_ϕ	0.10	0.06	0.23	0.040
ACG_θ	0.0026	0.00005	0.008	0.0019
ACG_θ^χ	0.0027	0.00001	0.008	0.0019

Table 2.1: Learning parameters for algorithms for the proportion vector problem.

Figures 2.4, 2.5, 2.6, 2.7 and 2.8 show that IEM, GCG, CG, EM_η and EG_η all give a clear improvement over the EM algorithm. For each of these algorithms convergence is faster than the EM algorithm for all 20 test runs. The ACG_ϕ algorithm leads to an improvement on average, but in 4 cases the EM algorithm converges faster and in a further 2 cases, although the ACG_ϕ algorithm eventually overtakes the EM algorithm, the EM algorithm initially shows better convergence. The GP_η, ACG_θ and ACG_θ^χ algorithm all perform poorly. These algorithms have particular problems when the stationary point to which they are converging lies on the boundary of the parameter space. This occurred twice in the 20 trials and these two occasions can be clearly seen in Figures 2.9, 2.11 and 2.12; they are represented by the two dotted lines that lie towards the bottom of the plot. If we ignore these two cases for the moment and look at the average performance for these three algorithms on the remaining 18 cases, shown by the bold dashed lines, we see that only for the ACG_θ^χ algorithm is the EM algorithm still better

on average.

It is quite difficult to use these plots to make a direct comparison between the CG and GCG algorithms and the others as both of these algorithms are preceded by a few iterations of the EM algorithm in order to ensure that we are close to a stationary point. To compare their performance to the other algorithms Figure 2.13 directly compares GCG, the better of the two conjugate gradient algorithms, and EG_η, which, out of the remaining algorithms, gives the best average performance. It shows that, although initially the EG_η algorithm displays faster convergence, once we change from EM to the GCG algorithm the GCG algorithm swiftly overtakes the EG_η algorithm.

Thus on a per iteration basis the conjugate gradient algorithms offer the greatest average improvement over the EM algorithm, followed by EG_η, EM_η, ACG_ϕ and IEM. The remaining algorithms, GP_η, ACG_θ and ACG_θ^x , do not perform as well as the EM algorithm.

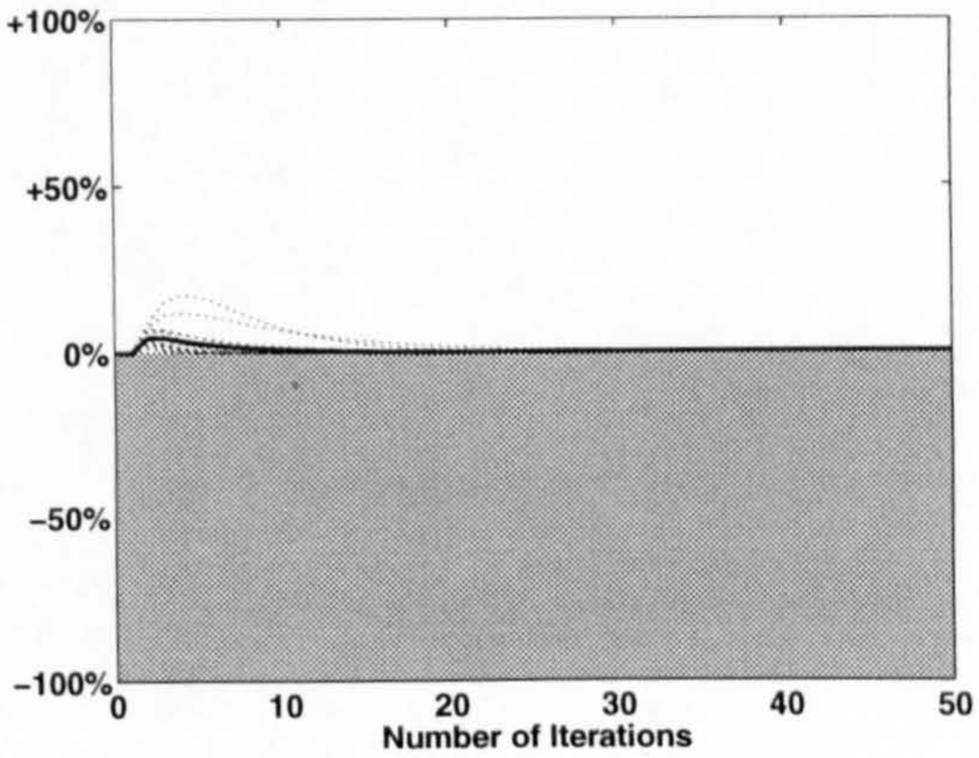


Figure 2.4: Comparison of IEM algorithm to EM algorithm. Lines plotted show the performance of the IEM algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the IEM being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

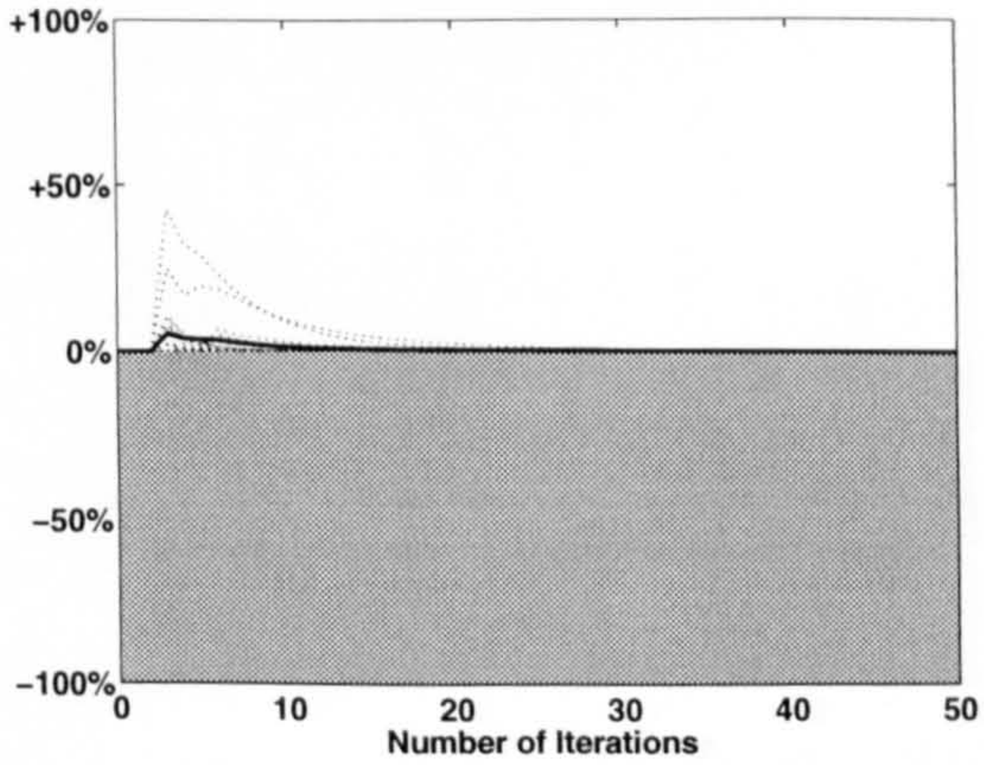


Figure 2.5: Comparison of GCG algorithm to EM algorithm. Lines plotted show the performance of the GCG algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the GCG being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

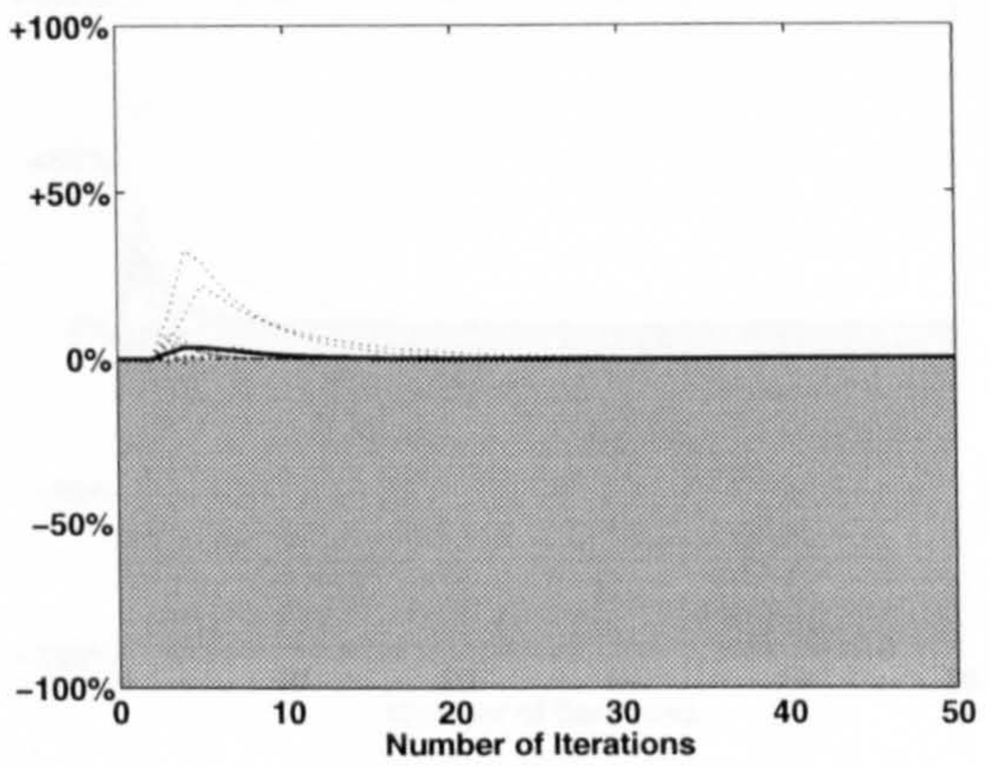


Figure 2.6: Comparison of CG algorithm to EM algorithm. Lines plotted show the performance of the CG algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the CG being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

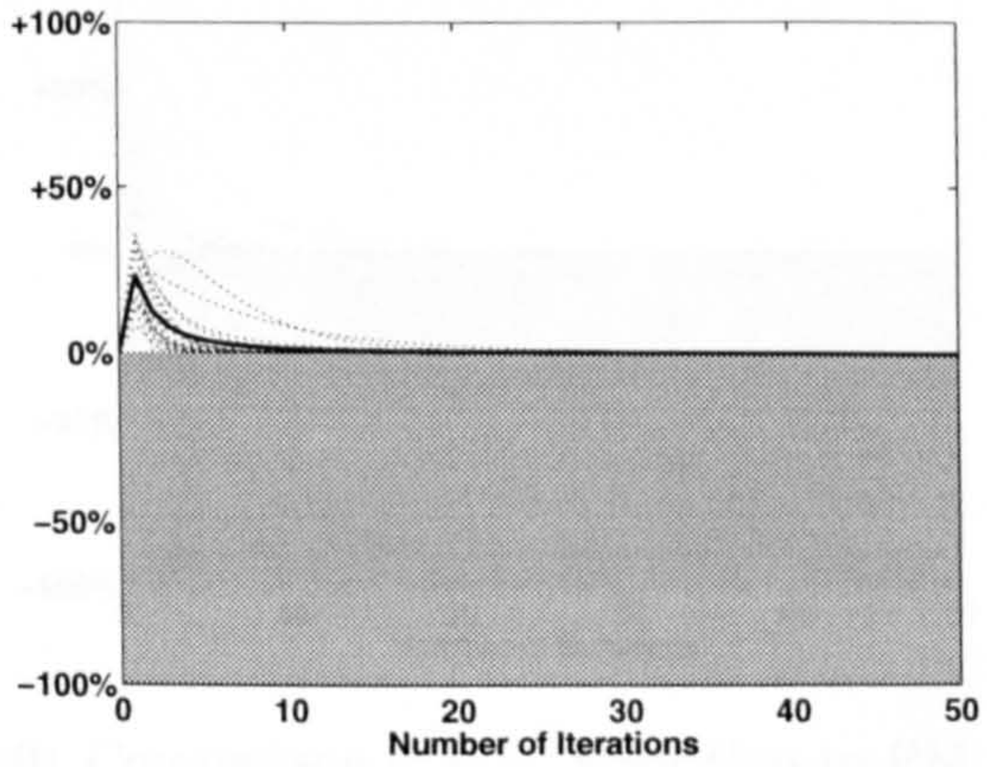


Figure 2.7: Comparison of EM_η algorithm to EM algorithm. Lines plotted show the performance of the EM_η algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the EM_η being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

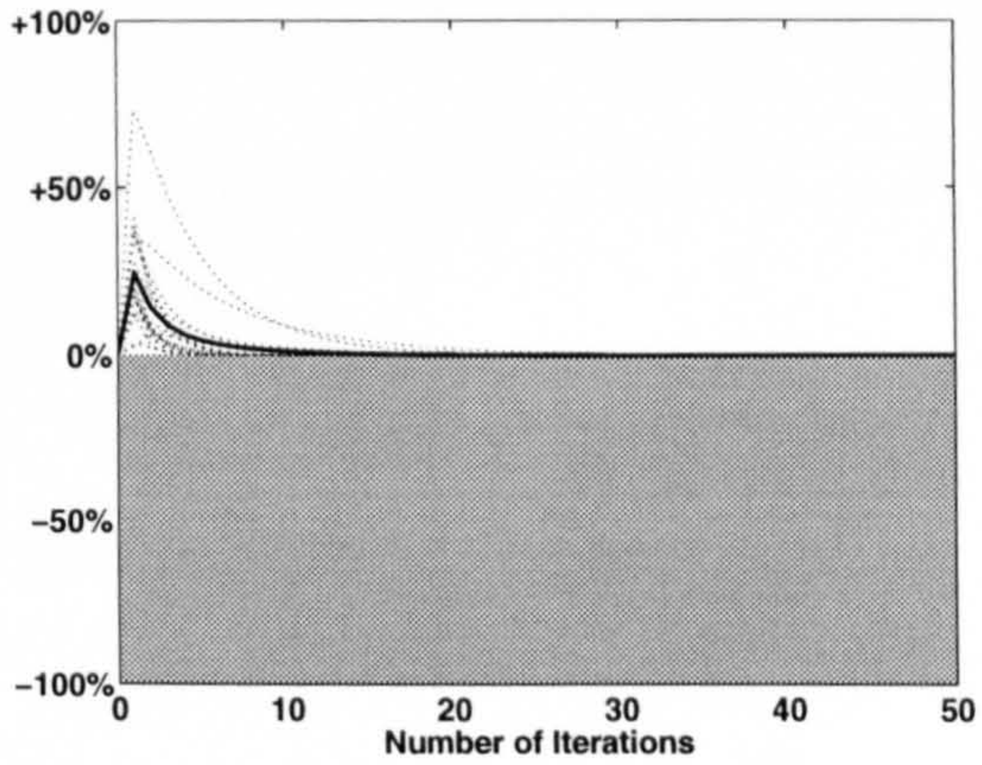


Figure 2.8: Comparison of EG_η algorithm to EM algorithm. Lines plotted show the performance of the EG_η algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the EG_η being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

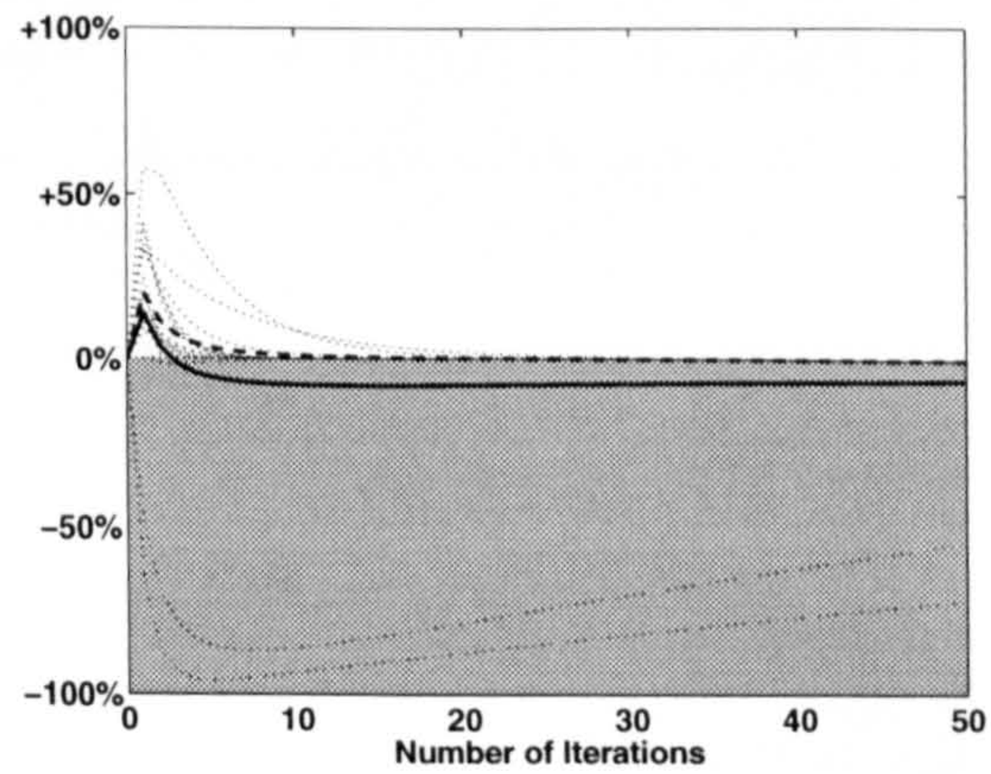


Figure 2.9: Comparison of GP_η algorithm to EM algorithm. Lines plotted show the performance of the GP_η algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the GP_η being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons, and the solid dashed line shows the average after ignoring the two cases which lie on the edge of the parameter space.

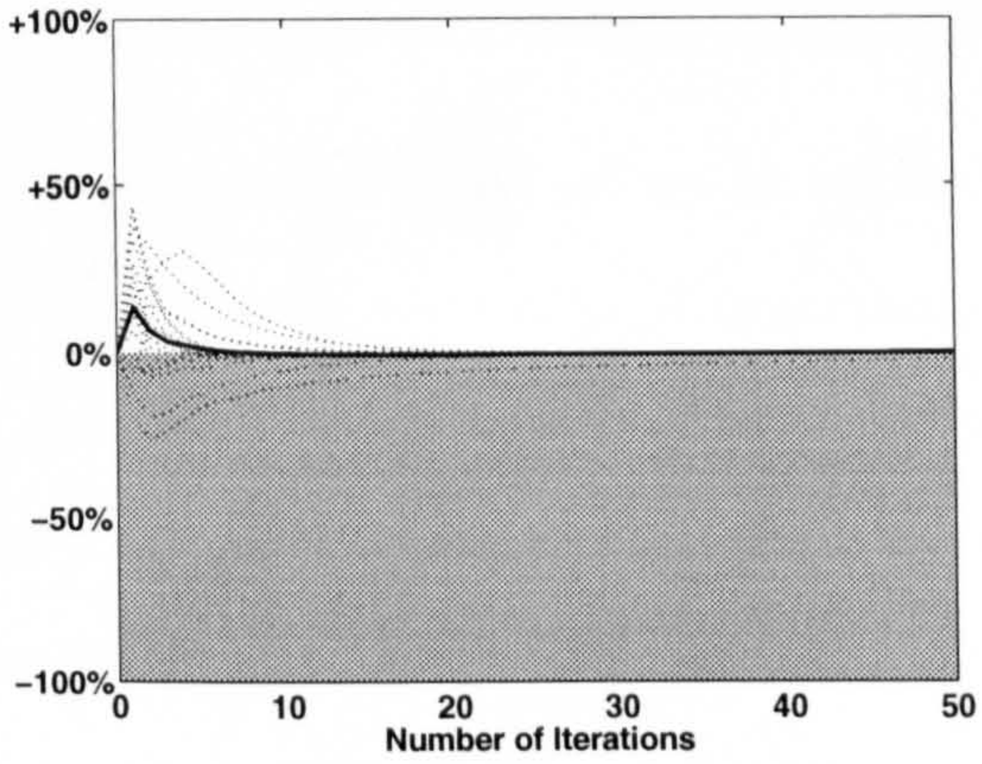


Figure 2.10: Comparison of ACG_ϕ algorithm to EM algorithm. Lines plotted show the performance of the ACG_ϕ algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the ACG_ϕ being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

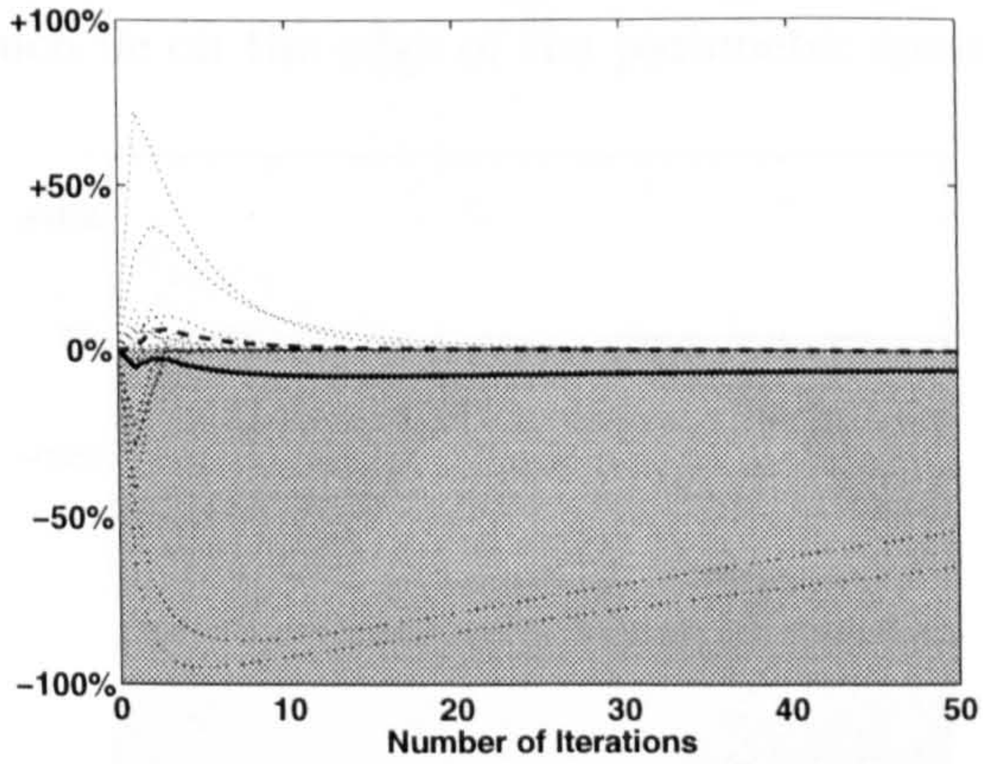


Figure 2.11: Comparison of ACG_θ algorithm to EM algorithm. Lines plotted show the performance of the ACG_θ algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the ACG_θ being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons, the solid line shows the average of these 20 comparisons, and the solid dashed line shows the average after ignoring the two cases which lie on the edge of the parameter space.

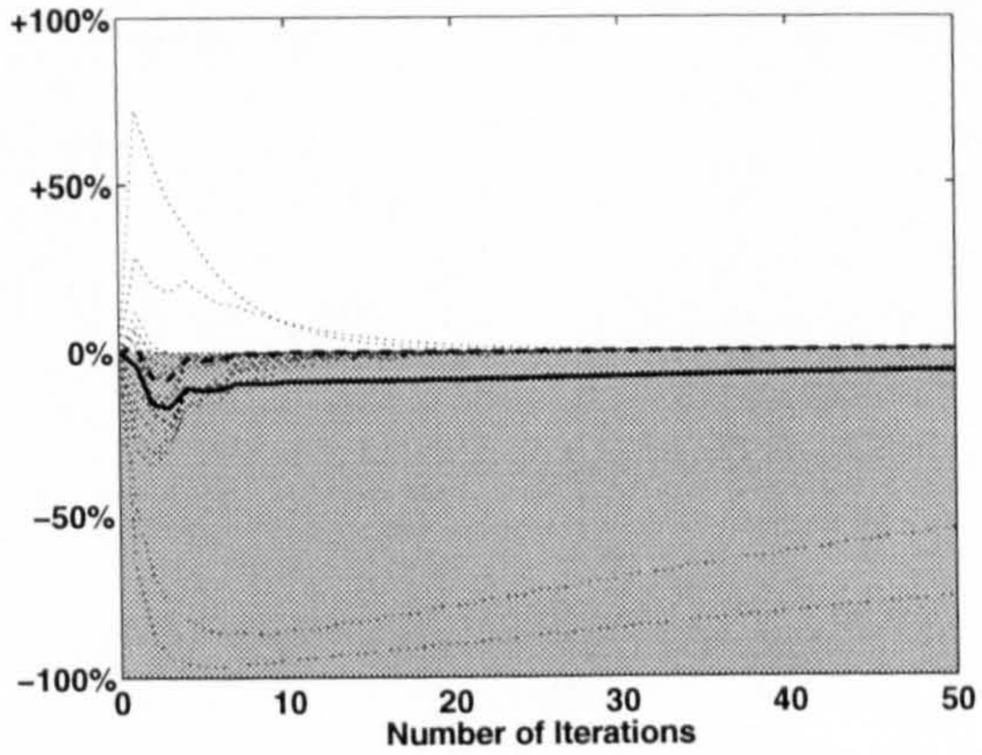


Figure 2.12: Comparison of ACG_θ^χ algorithm to EM algorithm. Lines plotted show the performance of the ACG_θ^χ algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the ACG_θ^χ being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons, the solid line shows the average of these 20 comparisons, and the solid dashed line shows the average after ignoring the two cases which lie on the edge of the parameter space.

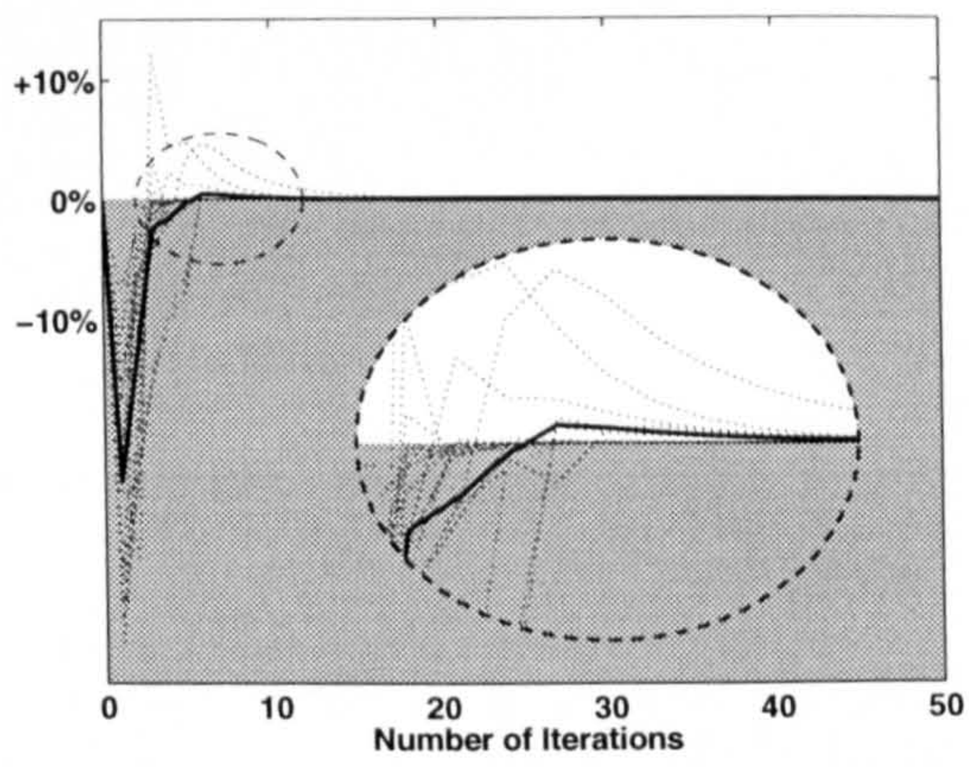


Figure 2.13: Comparison of GCG algorithm to EG_η algorithm. Lines plotted show the performance of the GCG algorithm by comparison to the EG_η algorithm; points above the line $y = 0$ correspond to the GCG being closer to convergence than the EG_η algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EG_η algorithm. The dotted lines show each of 20 comparisons; the solid line shows their average.

2.8.2 Mixtures of Unknown Components

A generative naive Bayesian network was set up in which the root node Z had $r_z = 2$ possible states and there were $n = 4$ observable variables each with $r_i = 2$ possible states, the parameters of this network being selected from uniform distributions. A dataset was generated using the method outlined earlier from which all observations of Z were discarded. A Bayesian network with the same structure but in which all parameters were assumed to be unknown was then fitted to the resulting dataset using each of the algorithms under consideration. Once again this was repeated 20 times and the same method as previously was used to display the results which are shown in Figures 2.14 - 2.23. As before, learning parameters for the EM_η, EG_η, GP_η, ACG_ϕ , ACG_θ and ACG_θ^χ algorithms were selected by hand and summary statistics for these parameters are given in Table 2.2.

Algorithm	Learning Parameter			
	Mean	Minimum	Maximum	Standard Error
EM_η	1.91	1.7	2.2	0.17
EG_η	1.85	0.3	2.7	0.49
GP_η	0.24	0.005	0.7	0.23
ACG_ϕ	0.12	0.09	0.15	0.017
ACG_θ	0.0013	0.00003	0.003	0.0012
ACG_θ^χ	0.0018	0.0001	0.006	0.0017

Table 2.2: Learning parameters for algorithms for the mixtures of unknown components problem.

Of the methods looked at only three, GCG, CG and IEM, improved on the EM algorithm. Of the other algorithms ACG_ϕ , EM_η and EG_η would seem to offer a similar rate of convergence to the EM algorithm, whilst GP_η, ACG_θ

and ACG_θ^χ all show much slower convergence than the EM algorithm. Again, in order to facilitate direct comparison of the conjugate gradient methods to the other algorithms Figure 2.23 shows a comparison between the IEM and GCG algorithms; once again the conjugate gradient algorithms give the greatest improvement over the EM algorithm.

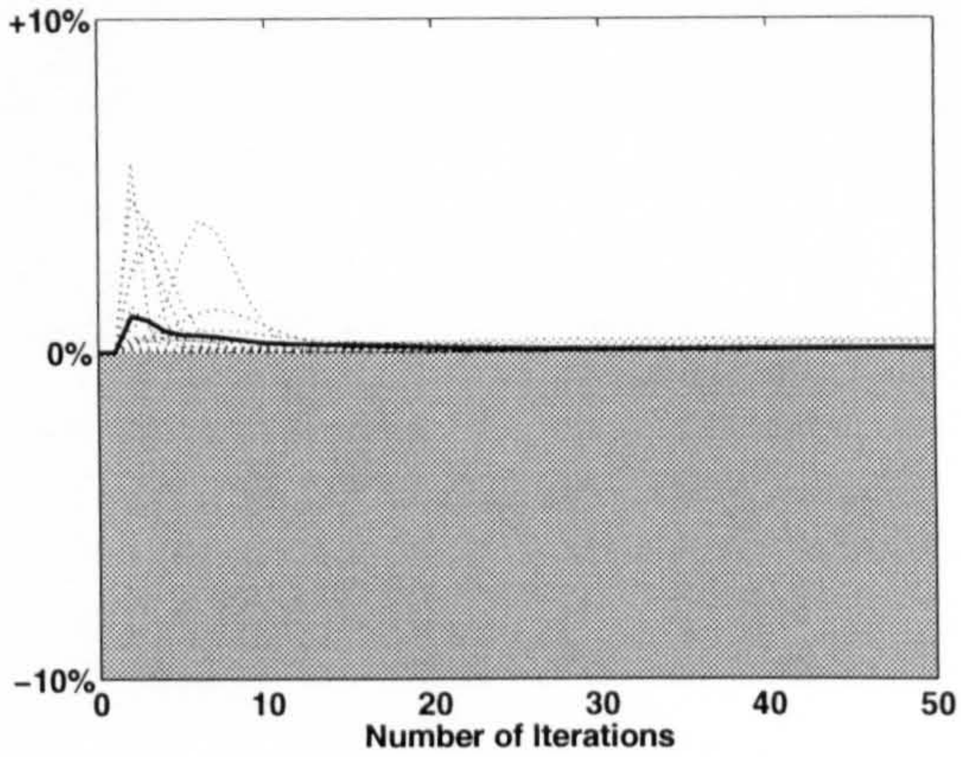


Figure 2.14: Comparison of IEM algorithm to EM algorithm. Lines plotted show the performance of the IEM algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the IEM being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

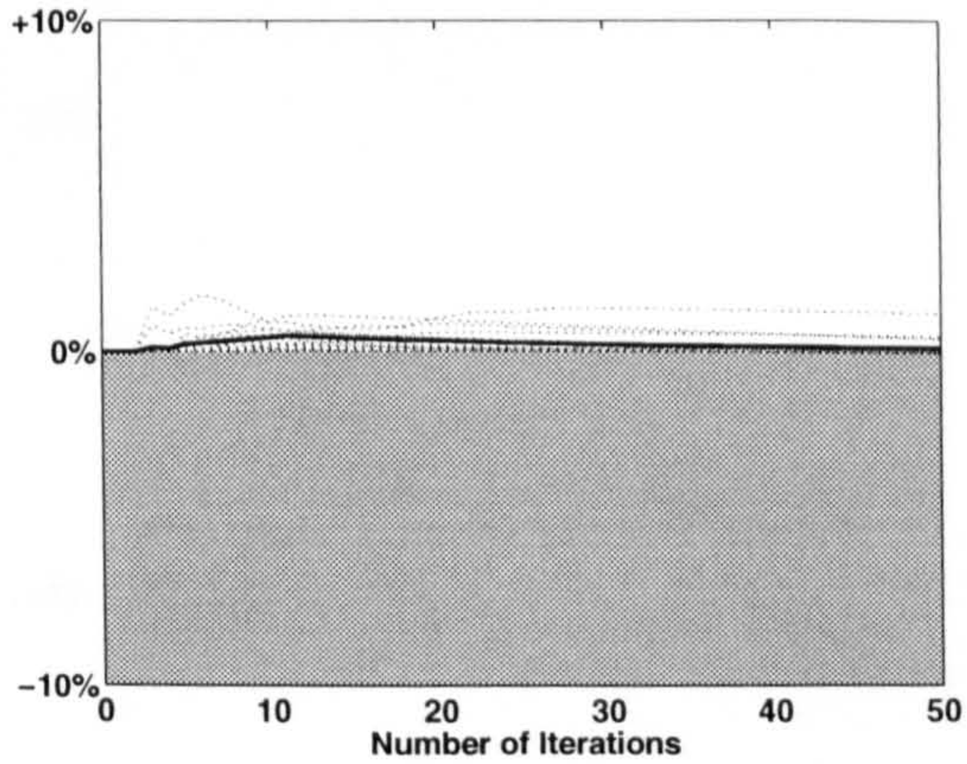


Figure 2.15: Comparison of GCG algorithm to EM algorithm. Lines plotted show the performance of the GCG algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the GCG being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

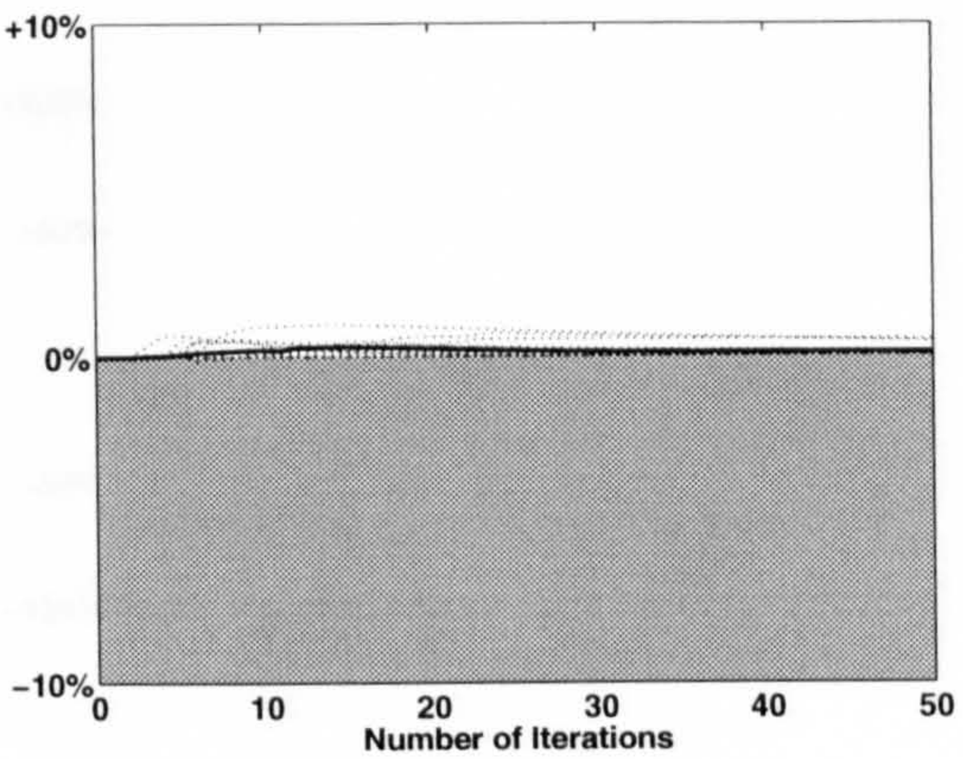


Figure 2.16: Comparison of CG algorithm to EM algorithm. Lines plotted show the performance of the CG algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the CG being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

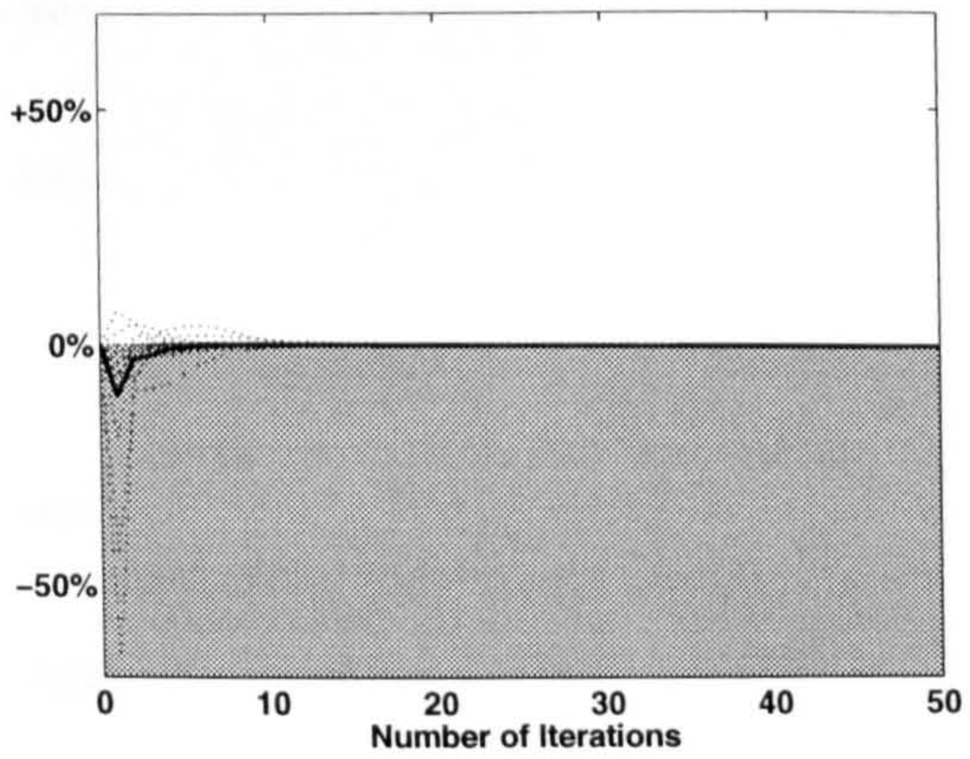


Figure 2.17: Comparison of EM_η algorithm to EM algorithm. Lines plotted show the performance of the EM_η algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the EM_η being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

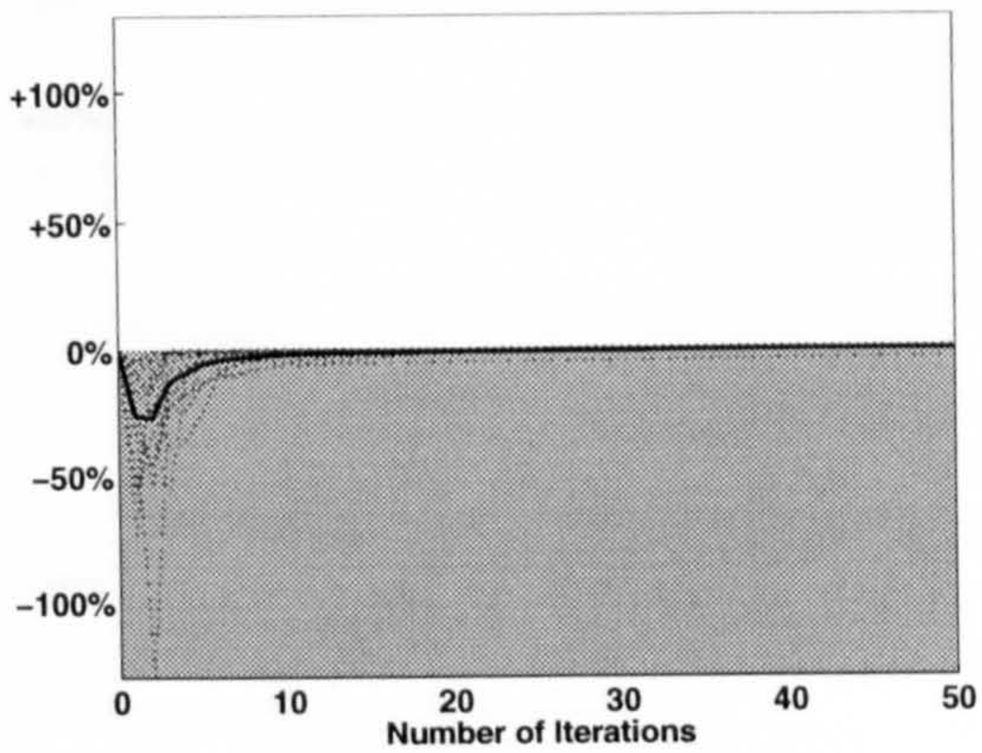


Figure 2.18: Comparison of EG_η algorithm to EM algorithm. Lines plotted show the performance of the EG_η algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the EG_η being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

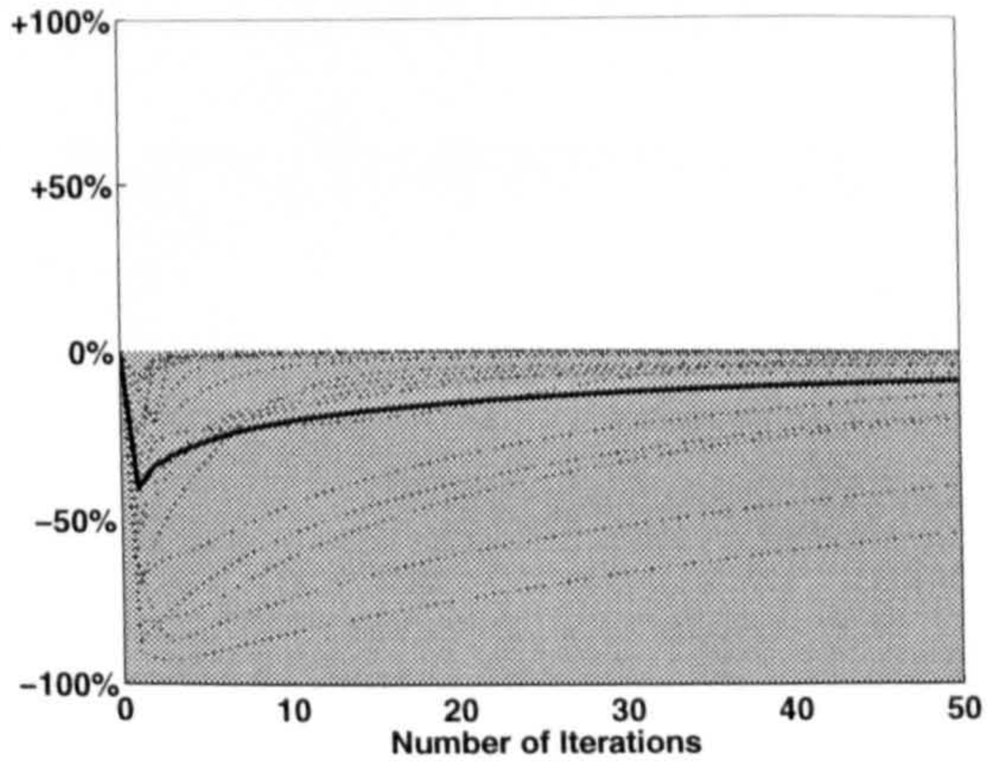


Figure 2.19: Comparison of GP_η algorithm to EM algorithm. Lines plotted show the performance of the GP_η algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the GP_η being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

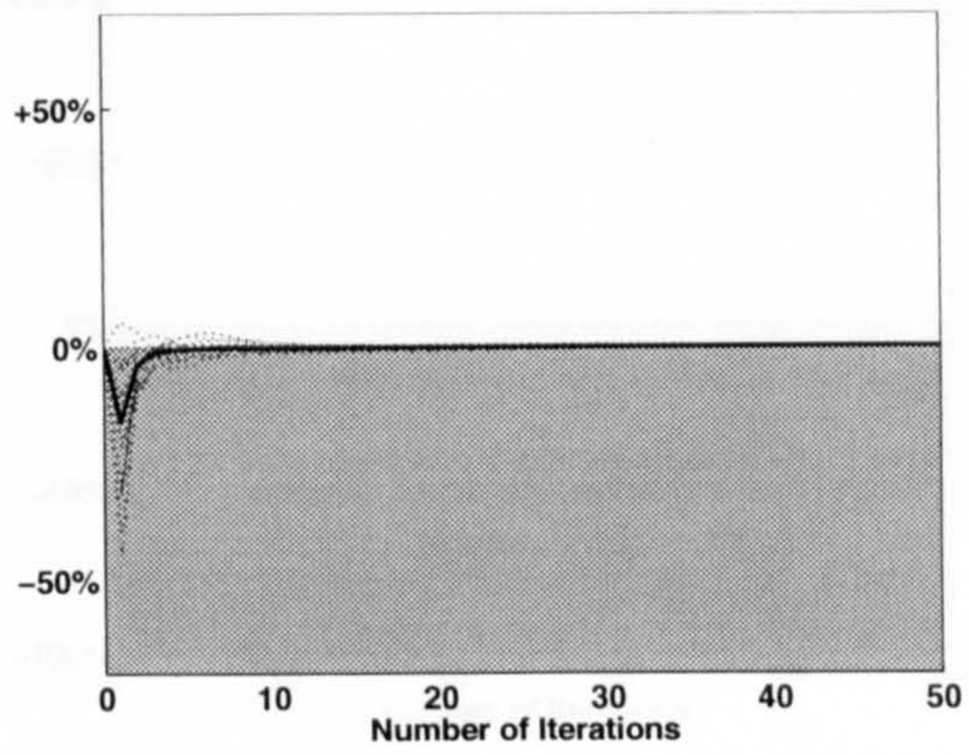


Figure 2.20: Comparison of ACG_ϕ algorithm to EM algorithm. Lines plotted show the performance of the ACG_ϕ algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the ACG_ϕ being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

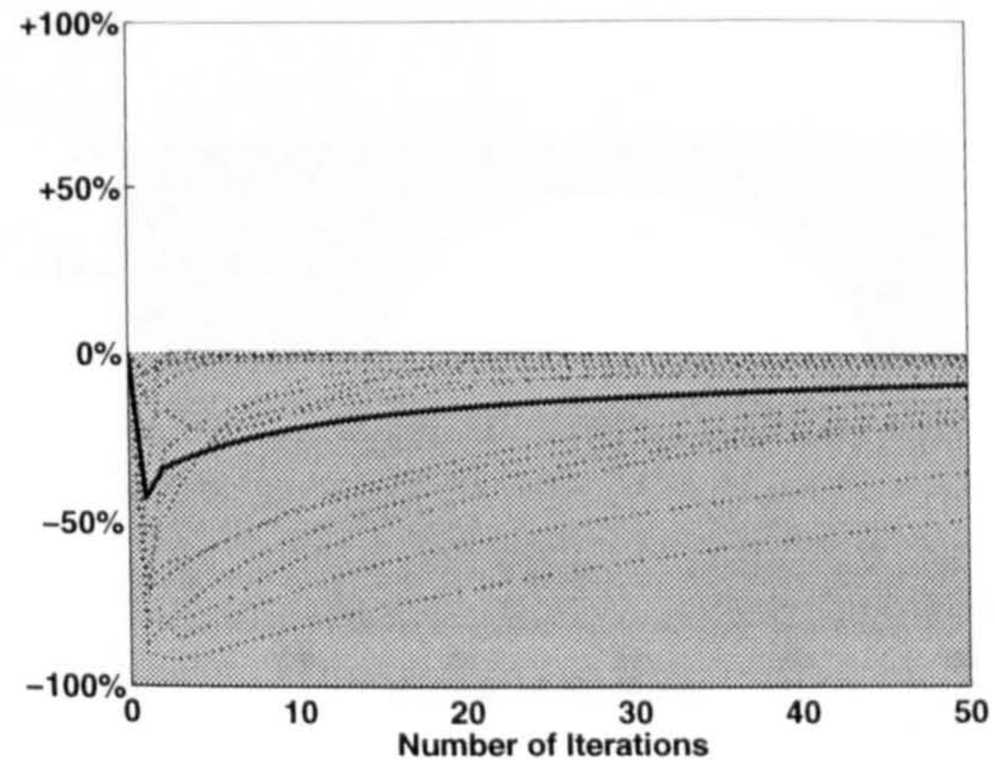


Figure 2.21: Comparison of ACG_θ algorithm to EM algorithm. Lines plotted show the performance of the ACG_θ algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the ACG_θ being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

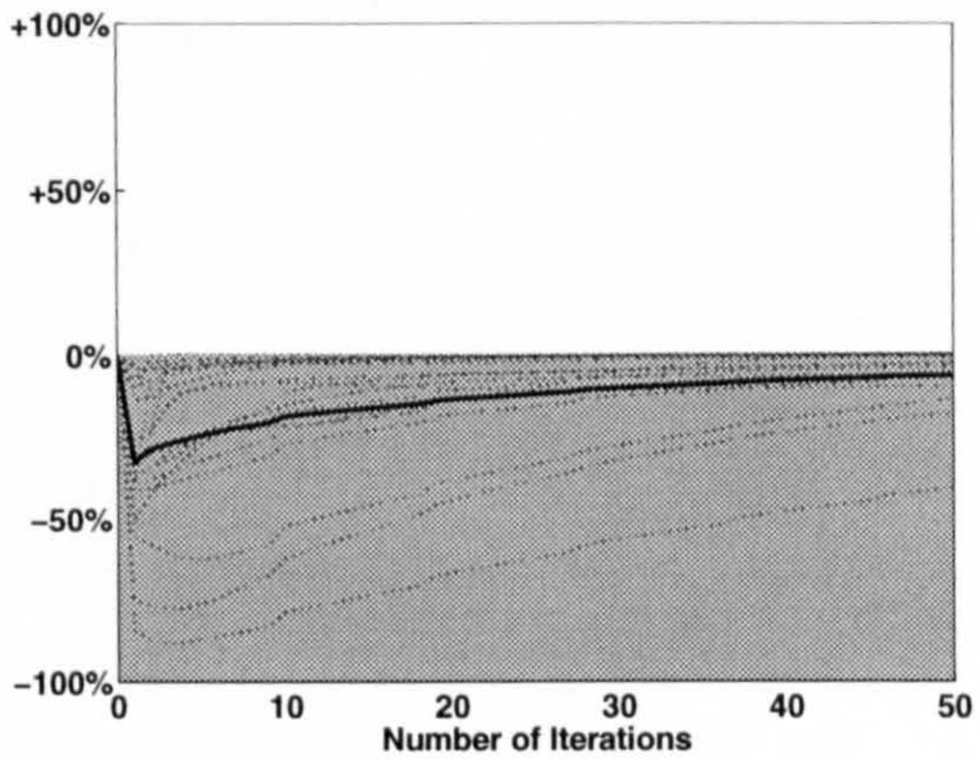


Figure 2.22: Comparison of ACG_θ^χ algorithm to EM algorithm. Lines plotted show the performance of the ACG_θ^χ algorithm by comparison to the EM algorithm; points above the line $y = 0$ correspond to the ACG_θ^χ being closer to convergence than the EM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the EM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

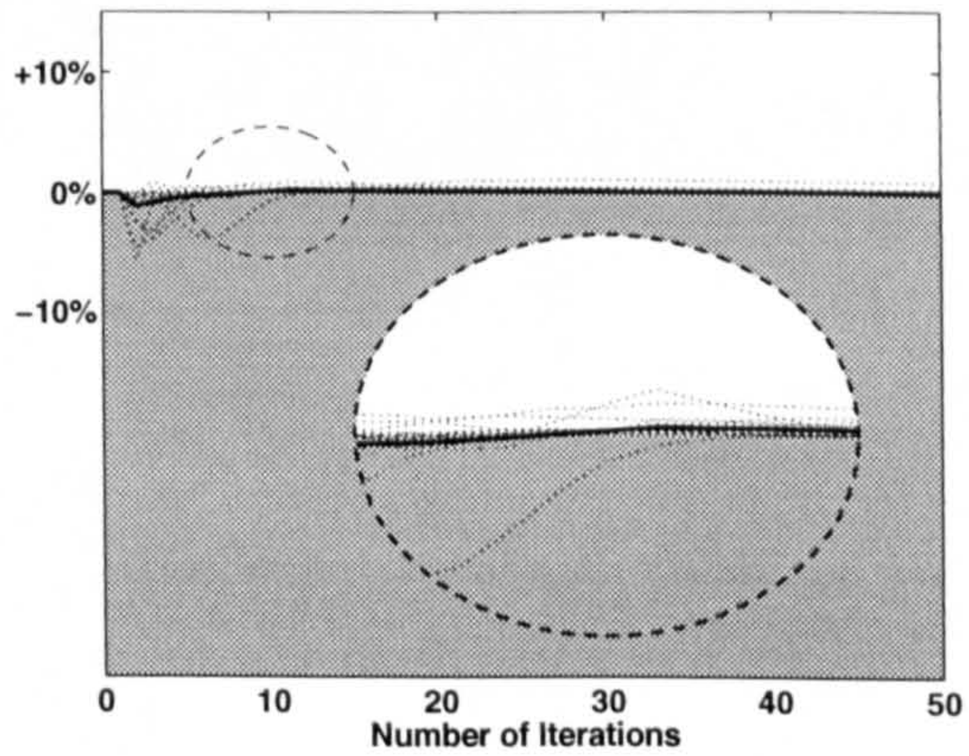


Figure 2.23: Comparison of GCG algorithm to IEM algorithm. Lines plotted show the performance of the CGG algorithm by comparison to the IEM algorithm; points above the line $y = 0$ correspond to the GCG being closer to convergence than the IEM algorithm and conversely points below $y = 0$ correspond to being further from convergence than the IEM algorithm. The dotted lines show each of 20 comparisons; the solid line shows the average of these 20 comparisons.

2.9 Computational Costs of Algorithms

When considering the relative merits of these algorithms, as well as comparing their performance on a 'per iteration' basis, we also have to consider the computational cost of each step of these iterations. An algorithm that may seem to give excellent performance when compared to the EM algorithm per iteration may in fact be slow and cumbersome. Some balance must be struck between the need for faster per iteration convergence and the amount of computation each step takes.

We also have to consider the rather intangible factor of ease of implementation. Clearly we do not want to invest a great deal of extra effort in coding up and testing routines which are not going to offer a reasonable improvement in the rate of convergence.

We begin by considering computational cost. We can compare the speed of our algorithms by considering the number of floating point operations (FLOPS) required by each iteration of the algorithm, if we take each of the operations addition, subtraction, multiplication, division and taking logs and exponents to be a single FLOP, and also assume that the computational costs of all other operations such as moving data or comparing values has negligible computational cost then we can calculate the number of FLOPS required by each iteration of our algorithms and hence compare the amount of computation required by each. Table 2.3 shows the FLOP count when applying these algorithms to the proportion vector problem, and Table 2.4 gives the FLOP counts for the problem in which our mixture components are also unknown. Each iteration of the algorithms has been broken down into 4 stages, calculating the gradient,

determining the search direction, calculating the step length and performing the final update. Not all of these 4 stages are applicable to all of the algorithms. The FLOP counts for choosing the search direction assume that we are not at the first iteration of the algorithm; in this case all of the algorithms apart from GCG take the gradient as the initial search direction and no additional computational cost is incurred.

It is quite difficult to interpret these raw formulae for computational requirement other than to say that moving down each of the tables there is a tendency for computational expense to increase. To see more clearly what these formulae mean Figures 2.24-2.28 show how some of these algorithms compare to the EM algorithm when we judge convergence against computation time as opposed to number of iterations for a single example. The examples shown here are all taken from the the earlier comparison of the algorithms in estimating the proportion vector for a mixture of known distributions. One very noticeable feature of these graphs is the much greater computational requirements of the CG and GCG algorithms. The effect this increase in computational requirement has on the relative speed of convergence is particularly obvious in this somewhat simple example in which the EM algorithm converges quite quickly anyway. For the other algorithms computational requirements are of a similar order to those for the EM algorithm in this example so their performance is similar whether compared on a per iteration basis or in terms of computational requirement. However the figures in Tables 2.3 and 2.4 indicate that this difference in computational cost will become more apparent as the complexity of our model increases.

A final point of consideration is ease of implementation. If the EM algorithm

Algorithm	Calculating Gradient	Choose Search Direction	Step Length	Update	Total FLOPS per Iteration
EM				$Nc(3+n)+c$	$Nc(3+n)+c$
IEM				$Nc(6+n)$	$Nc(6+n)$
EM_η	$Nc(2n+3)$			$5c$	$Nc(2n+3)+5c$
EG_η	$Nc(2n+3)$			$6c$	$Nc(2n+3)+6c$
GP_η	$Nc(2n+3)$			$5c+1$	$Nc(2n+3)+5c+1$
ACG_ϕ	$N\left(nc^2+3c^2-2\right)$	$6(c-1)+1$	$4(c-1)+2$	$3(c-1)$	$N\left(nc^2+3c^2-2\right)+13(c-1)+3$
ACG_θ	$N(c(n+1)$ $+(c-1)(2n+3))$	$6(c-1)+1$	$4(c-1)+2$	$2(c-1)$	$N(c(n+1)+(c-1)(2n+3))+12(c-1)+3$
ACG_θ^χ	$N(c(n+1)$ $+(c-1)(2n+3))$	$6(c-1)+1$	$5(c-1)+2$	$2(c-1)$	$N(c(n+1)+(c-1)(2n+3))+13(c-1)+3$
CG	$N((n+1)(2c-1)$ $+(n+2)(c-1))$	$6(c-1)+1$	$4(3(c-1)+N(c(n+1)+1))$ $+x[2+3(c-1)+N(c(n+1)+1)]$	$2(c-1)$	$N((n+1)(2c-1)+(n+2)(c-1))+8(c-1)+1$ $+4(3(c-1)N(c(n+1)+1))$ $+x[2+3(c-1)N(c(n+1)+1)]$
GCG	$N((n+1)(2c-1)$ $+(n+2)(c-1))$	$Nc(3+n)+c$ $+8(c-1)+1$	$4(3(c-1)+N(c(n+1)+1))$ $+x[2+3(c-1)+N(c(n+1)+1)]$	$2(c-1)$	$N((n+1)(2c-1)+(n+2)(c-1))$ $+Nc(3+n)+c+10(c-1)+1$ $+4(3(c-1)+N(c(n+1)+1))$ $+x[2+3(c-1)N(c(n+1)+1)]$

Table 2.3: The number of floating point operations (FLOPS) required per iteration of each of the algorithms shown when determining Maximum Likelihood parameter estimates for a naive Bayesian network in which only the proportion vector is unknown. The Bayesian network has n binary observable variables, a root node with c states, and our dataset is of size N. The variable x in the calculation of the number of FLOPS per iteration for the CG and GCG algorithms represents the fact that the number of function evaluations required by the line search algorithms used to determine step length for these algorithms can not be predetermined.

Algorithm	Calculating Gradient	Choose Search Direction	Step Length	Update	Total FLOPS per Iteration
EM				$Nc\,(2n+3)+c\,(n+1)$	$N\,(2nc+3c)+c\,(n+1)$
IEM				$Nc\,(4n+5)$	$Nc\,(4n+5)$
EM_η	$Nc\,(n+1)\,(n+3)$			$c\,(11n+5)$	$Nc\,(n+1)\,(n+3)+c\,(11n+5)$
EG_η	$Nc\,(n+1)\,(n+3)$			$c\,(13n+6)$	$Nc\,(n+1)\,(n+3)+c\,(13n+6)$
GP_η	$Nc\,(n+1)\,(n+3)$			$c\,(8n+5)$	$Nc\,(n+1)\,(n+3)+c\,(8n+5)$
ACG_ϕ	$N\,(c\,(n+1)$ $+(c-1)\,(c+1)$ $+nc\,(5+n))$	$6\,(nc+c-1)+1$	$4\,(n+c-1)+2$	$2\,(nc+c-1)$	$N\,(c\,(n+1)+(c-1)\,(c+1)+nc\,(5+n))$ $+12\,(nc+c-1)$
ACG_θ	$N\,(c\,(n+1)$ $(2c-2+nc)\,(n+2))$	$6\,(nc+c-1)+1$	$4\,(n+c-1)+2$	$2\,(nc+c-1)$	$N\,(c\,(n+1)\,(2c-2+nc)\,(n+2))$ $+12\,(nc+c-1)$
$\mathrm{ACG}_\theta^{\mathrm{X}}$	$N\,(c\,(n+1)$ $(2c-2+nc)\,(n+2))$	$6\,(nc+c-1)+1$	$5\,(n+c-1)+2$	$2\,(nc+c-1)$	$N\,(c\,(n+1)\,(2c-2+nc)\,(n+2))$ $+13\,(nc+c-1)$
CG	$N\,(c\,(n+1)$ $(2c-2+nc)\,(n+2))$	$6\,(nc+c-1)+1$	$4\,(2nc+3\,(c-1)$ $+N\,(c\,(n+1)+1))$ $+x\,[2+2nc+3\,(c-1)$ $+N\,(c\,(n+1)+1)]$	$2\,(nc+c-1)$	$N\,(c\,(n+1)\,(2c-2+nc)\,(n+2))$ $+8\,(nc+c-1)+1$ $+4\,(2nc+3\,(c-1)+N\,(c\,(n+1)+1))$ $+x\,[2+2nc+3\,(c-1)+N\,(c\,(n+1)+1)]$
GCG	$N\,(c\,(n+1)$ $(2c-2+nc)\,(n+2))$	$Nc\,(2n+3)$ $+c\,(n+1)$ $+8\,(nc+c-1)+1$	$4\,(2nc+3\,(c-1)$ $+N\,(c\,(n+1)+1))$ $+x\,[2+2nc+3\,(c-1)$ $+N\,(c\,(n+1)+1)]$	$2\,(nc+c-1)$	$N\,(c\,(n+1)\,(2c-2+nc)\,(n+2))$ $+Nc\,(2n+3)+c\,(n+1)$ $+10\,(nc+c-1)+1$ $+4\,(2nc+3\,(c-1)+N\,(c\,(n+1)+1))$ $+x\,[2+2nc+3\,(c-1)+N\,(c\,(n+1)+1)]$

Table 2.4: The number of floating point operations (FLOPS) required per iteration of each of the algorithms shown when determining Maximum Likelihood parameter estimates gor a naive Bayesian network in which both the proportion vector and the parameters of the mixture components are unknown. The Bayesian network has n binary observable variables, a root node with c states, and our dataset is of size N. The variable x in the calculation of the number of FLOPS per iteration for the CG and GCG algorithms represents the fact that the number of function evaluations required by the line search algorithms used to determine step length for these algorithms can not be predetermined.

is already implemented then the IEM algorithm will be by far the easiest of these algorithms to implement as it requires only a fairly minor alteration of the code for the EM algorithm. Otherwise, if starting from scratch, all the algorithms apart from CG and GCG required a similar amount of work to code up. However, for the EM_η, EG_η, GP_η, ACG_ϕ , ACG_θ and ACG_θ^χ there is an additional implementation cost, that of determining the learning parameters. Tables 2.1 and 2.2 show how these learning parameters can vary even when we are fitting the same model structure to different datasets. It turns out for all of these algorithms that a learning parameter that gives good convergence when fitting the model to one dataset may be too small to give good convergence when applied to another dataset, or alternatively may be too large and the algorithm breaks down and fails to give a sensible answer. The CG and GCG algorithms were by far the most difficult to implement as they are considerably more sophisticated than the other techniques considered here. One particularly difficult issue here is that of the line search algorithm used. A huge number of such algorithms exist and it is important to strike up the right balance between selecting a method which is quick yet sufficiently accurate.

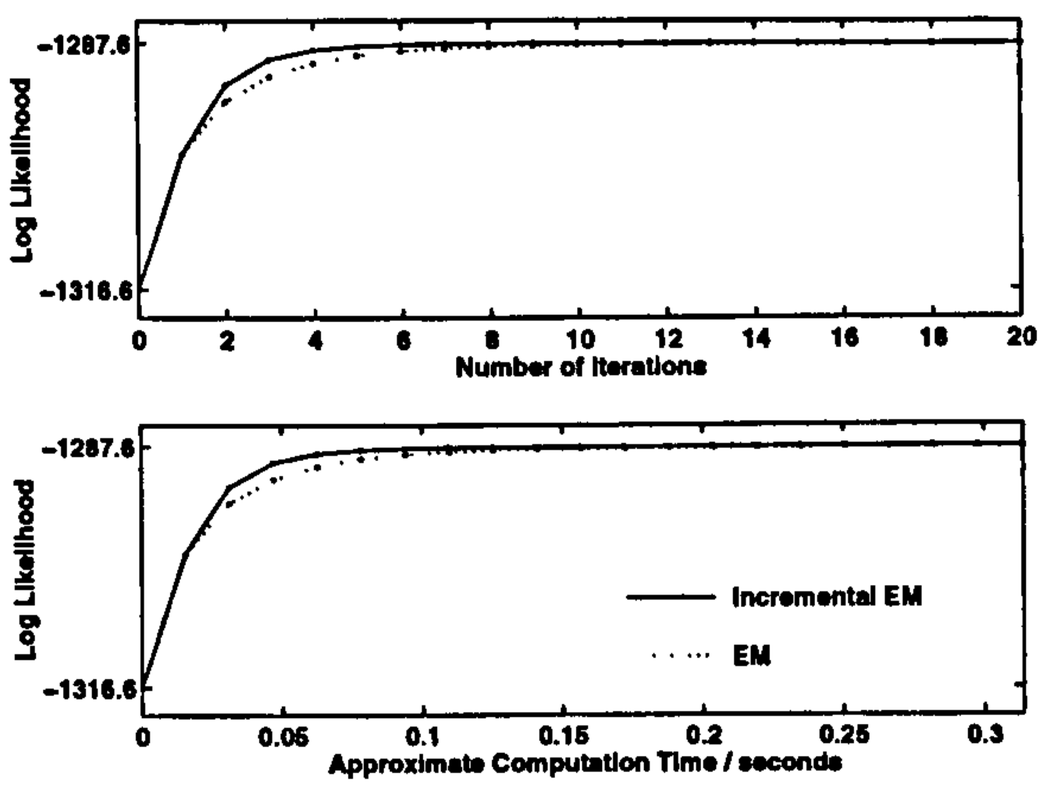


Figure 2.24: Application of the IEM algorithm to a naive Bayesian network with 5 binary observable nodes, and an unobserved root node having 4 possible states. It is assumed that only the parameters of the local distribution of the root node are unknown. The EM algorithm is also shown here for comparison.

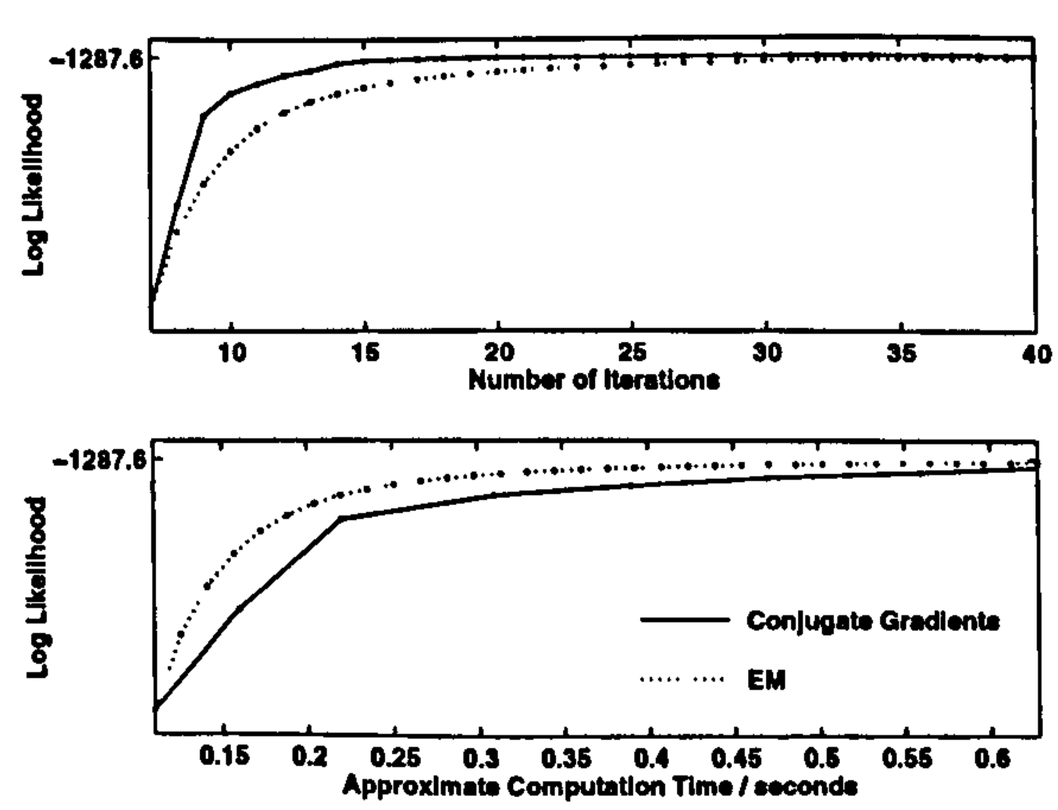


Figure 2.25: Application of the CG algorithm to a naive Bayesian network with 5 binary observable nodes, and an unobserved root node having 4 possible states. It is assumed that only the parameters of the local distribution of the root node are unknown. The EM algorithm is also shown here for comparison. Since the CG algortihm is preceded by a few steps of the EM algorithm step 1 is considered here to be the first CG step.

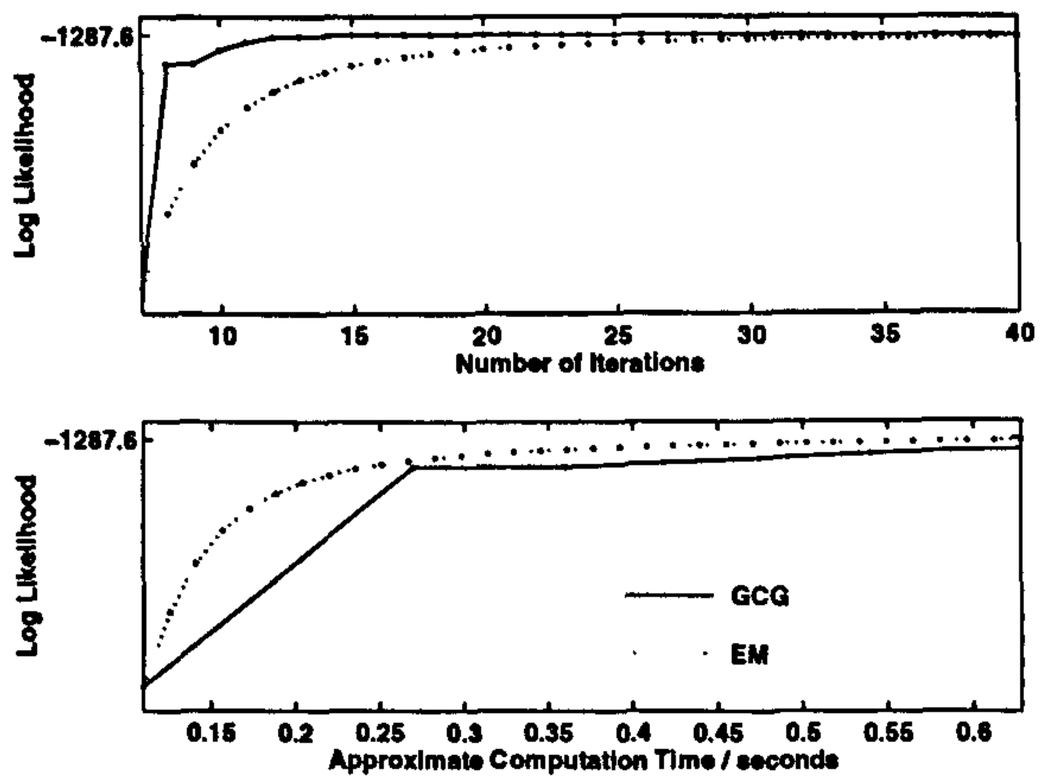


Figure 2.26: Application of the GCG algorithm to a naive Bayesian network with 5 binary observable nodes, and an unobserved root node having 4 possible states. It is assumed that only the parameters of the local distribution of the root node are unknown. The EM algorithm is also shown here for comparison. Since the GCG algortihm is preceded by a few steps of the EM algorithm step 1 is considered here to be the first GCG step.

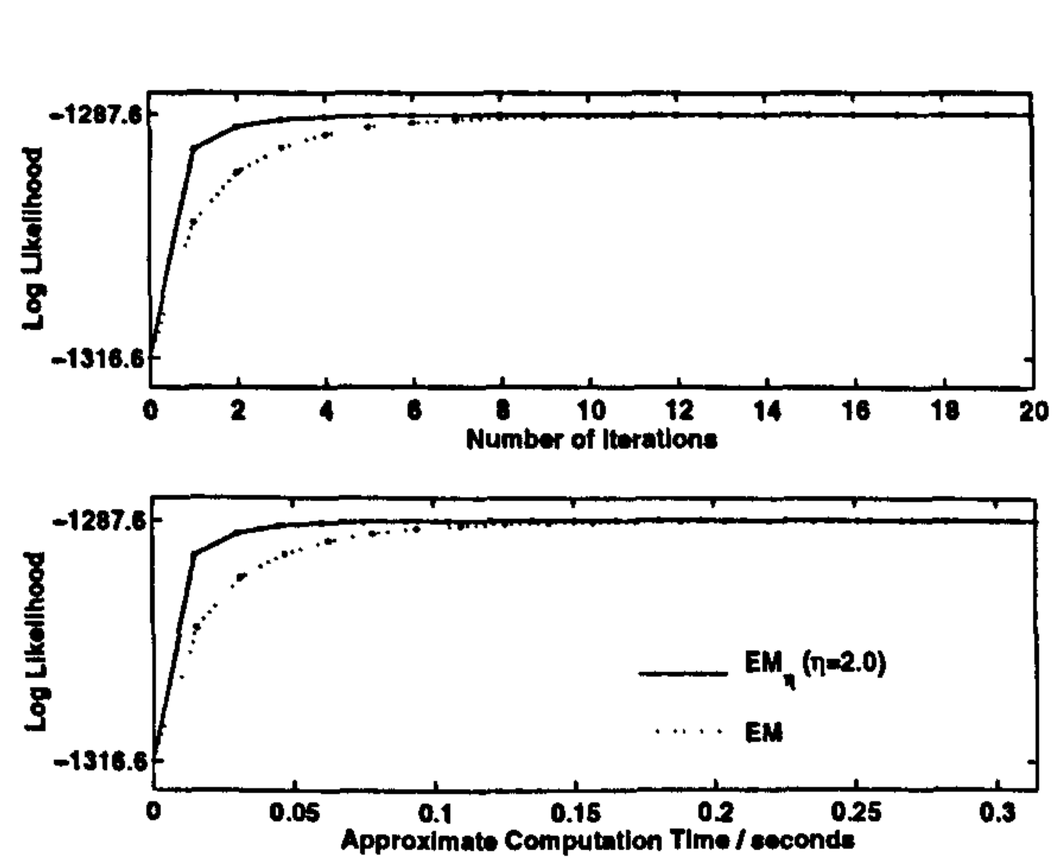


Figure 2.27: Application of the EM_η algorithm to a naive Bayesian network with 5 binary observable nodes, and an unobserved root node having 4 possible states. It is assumed that only the parameters of the local distribution of the root node are unknown. The EM algorithm is also shown here for comparison.

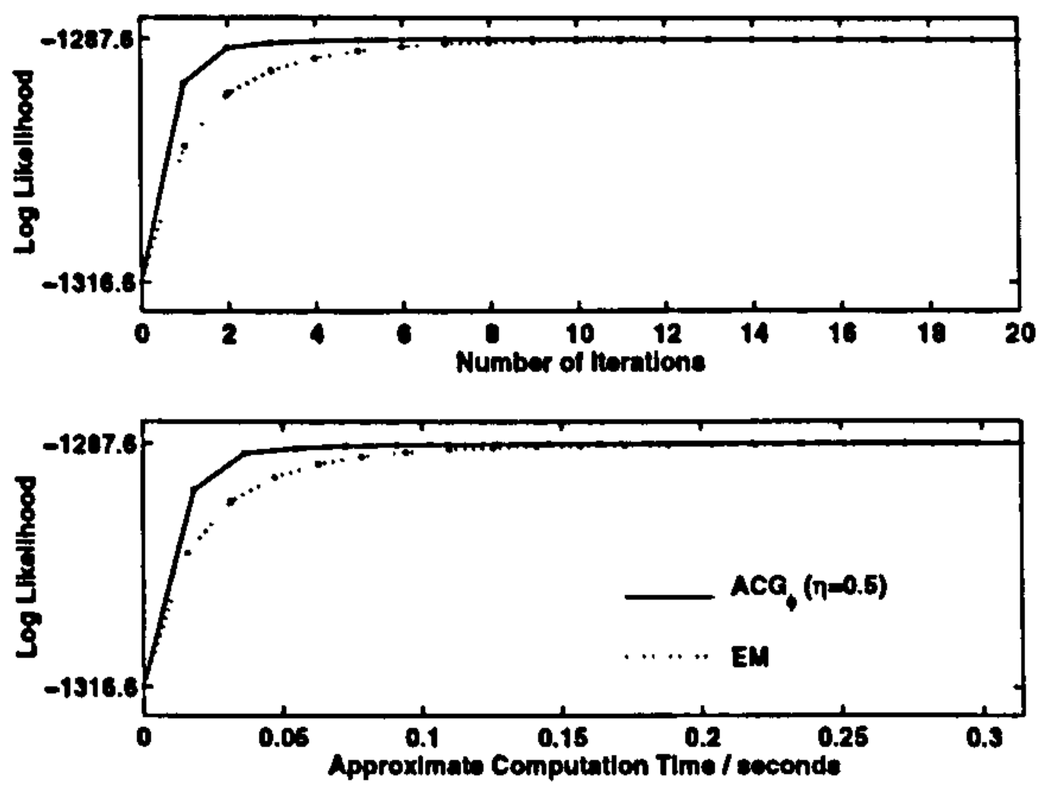


Figure 2.28: Application of the ACG algorithm to a naive Bayesian network with 5 binary observable nodes, and an unobserved root node having 4 possible states. It is assumed that only the parameters of the local distribution of the root node are unknown. The EM algorithm is also shown here for comparison.

2.10 Conclusion

Of the algorithms studied here only three proved to be better than the EM algorithm for both of the types of problem looked at. Of these three the IEM algorithm can be expected to improve performance for little extra effort, while CG and GCG both offer considerable per iteration improvement over the EM algorithm but require more time to be invested in their implementation and require greater computational resources. The other algorithms proved to have very poor convergence properties when applied to more general problems, but a number of them performed well when applied to the proportion vector problem. Helmbold's EM_η and EG_η algorithms were both particularly successful, being beaten on a per iteration basis by only the GCG and CG algorithms, and yet they require far less initial investment and are computationally much less intensive. These algorithms do however have the disadvantage that they require the selection of a learning parameter and that at present no automatic technique for the selection of this learning parameter exists. Choosing a suitable parameter can be time consuming.

If the problem of selecting a learning parameter can be successfully automated then for proportion vector problems Helmbold's techniques are to be recommended. Until such a time the choice is really between IEM and the conjugate gradient algorithms, the choice depending upon the user's willingness to put in the initial investment to prepare the more complicated code required for the conjugate gradient methods. For more complicated models either IEM or GCG would be recommended, the choice again depending upon the criteria just given.

2.11 Deterministic Annealing EM Algorithm

2.11.1 Introduction

We have seen some of the many attempts that have been made to improve on the convergence rate of the EM algorithm, but this is only half the story. As well as the speed of convergence we also have to consider just where these algorithms are converging to. We know that the EM algorithm guarantees convergence to a stationary point of the likelihood surface, but likelihood surfaces often have many stationary points, How can we tell if we are at the global maximum, or merely a local maximum, or even a saddle point? The simple answer is that we can never truly be sure that we have found the global maximum, but we can at least improve our chances of finding it. Previously the only way we had of improving our chances was the multiple re-start technique in which we run the EM algorithm many times with a different, randomly chosen, starting point for each run. We then select the best of the set of convergence points and assume that this is our global maximum, knowing that even if this is not the point that maximises our likelihood function the likelihood surface at this point must be close to its maximum value. Recently, Ueda and Nakano [72] have proposed another possible solution to this problem, their Deterministic Annealing EM (DAEM) algorithm. They reformulate maximising the log-likelihood as minimising the thermodynamic free energy, defined as an effective cost function that depends on a parameter $\frac{1}{\beta}$ called the 'temperature':

$$\mathcal{L}_\beta(\theta_m) = -\frac{1}{\beta}\log\sum_{\chi_{mis}} P(\chi_{obs}, \chi_{mis}; \theta_m)^\beta$$

where χ_{obs} and χ_{mis} represent the observed and missing data respectively. The temperature takes an initial value which is chosen to be high enough such that the function being minimised has a single global minimum. The temperature is then gradually lowered, and at each temperature the function is deterministically optimised. At the final temperature, $\frac{1}{\beta} = 1$, the function being minimised corresponds to the negative log-likelihood, so the point to which this algorithm converges must be a stationary point of our likelihood surface. Ueda and Nakano argue that, while this approach may not find a global maximum of the log-likelihood, it is likely to find a better maximum than that reached by the EM algorithm alone. They also point out that the maximum found by the DAEM algorithm will be independent of the chosen starting point, but will instead depend upon the position of the global minimum of $\mathcal{L}_\beta(\theta_m)$ at the initial temperature, and the chosen cooling schedule. They illustrate the use of the algorithm on training probabilistic neural networks (akin to mixtures of normal distributions) and show that, at least in their chosen example, the DAEM algorithm does find a better maximum than the EM algorithm. However, applying this algorithm to a naive Bayesian network produces some odd results. In this case the function being minimised at each temperature is

$$\mathcal{L}_\beta(\theta_m) = -\frac{1}{\beta}\sum_{m=1}^{N}\log\left(\sum_{j=1}^{r_z} p_j^\beta \prod_{i=1}^{n}\sum_{k=1}^{r_i} I_k(y_{mi})\,\theta_{ijk}^\beta\right),$$

where the root node Z has r_z states, and each of the n observed nodes $Y_1, \ldots, Y_n$ has r_i states, with

$$P(Z = j) = p_j$$

and

$$P(Y_i = k \mid Z = j) = \theta_{ijk};$$

$I_k(y_{mi})$ is the indicator function

$$I_k(y_{mi}) = \begin{cases} 1 & \text{if } y_{mi} = k \\ 0 & \text{otherwise.} \end{cases}$$

We found that the DAEM algorithm generally converges to the equally weighted independence model in which

$$p_j = \frac{1}{r_z} \quad \forall j \qquad \text{and} \qquad \theta_{ijk} = \frac{1}{N} \sum_{m=1}^{N} I_k(y_{mi}) \quad \forall i, j. \tag{2.26}$$

To understand this behaviour we need to consider the following theorems, proofs of which are given in Appendix B.

Theorem 2.1 *If the EM algorithm is started at any independence model, that is to say either*

$$\theta^0_{i1k} = \ldots = \theta^0_{ir_zk} \quad \forall i, k$$

or

$$p^0_h = 1 \quad \textit{and} \quad p^0_j = 0 \quad \forall j \neq h,$$

then the EM algorithm will converge to the maximum likelihood independence model in just one iteration. In the first case it converges to

$$p^1_j = p^0_j \quad \forall j \qquad \textit{and} \qquad \theta_{ijk} = \frac{1}{N} \sum_{m=1}^{N} I_k(y_{mi}) \quad \forall i, j, k$$

and in the second case to

$$p_h^1 = 1, \quad p_j^1 = 0 \quad \forall j \neq k \qquad \textit{and} \qquad \theta_{ihk} = \frac{1}{N} \sum_{m=1}^{N} I_k \left(y_{mi} \right) \quad \forall i, j, k.$$

Corollary 2.2 *The maximum likelihood independence model is a stationary point of the EM algorithm.*

Theorem 2.3 *The gradient of* $\mathcal{L}_\beta \left(\theta_m \right)$ *at the point*

$$p_j = \frac{1}{r_z} \quad \forall j \quad \textit{and} \quad \theta_{ijk} = \frac{1}{N} \sum_{m=1}^{N} I_k \left(x_{mi} \right) \quad \forall i, j, k$$

is zero for all $0 < \beta \leq 1$.

Theorem 2.3 indicates that the equally weighted independence model is a stationary point at all temperatures, the implication of this being that, when the temperature is high enough that the thermodynamic free energy surface has a single local minimum, that minimum must lie at this point. As the algorithm progresses $\frac{1}{\beta}$ will be lowered according to some cooling schedule. However at each new temperature the starting point will be the equally weighted independence model, which is a stationary point for all $1 > \beta > 0$, and the algorithm will be unable to escape that point. Corollary 2.2 then shows us that, at the final temperature, $\frac{1}{\beta} = 1$, this will also be a stationary point. Hence the DAEM algorithm as it stands will, provided the initial temperature is high enough, converge to the equally weighted independence model at this initial temperature and then fail to escape from this point.

Although we have demonstrated why there are problems with the convergence of the DAEM as it stands, Ueda and Nakano suggest a method for escaping from poor stationary points. They suggest that the Hessian of the stationary point be determined, and its eigenvalues calculated. Should the Hessian have negative, as well as positive, eigenvalues then it indicates that the stationary point is a saddlepoint rather than a local minimum, and we can escape that saddle point by conducting a line search in the direction of the eigenvectors corresponding to the negative eigenvalues of the Hessian. This is of course a computationally expensive step to add to the algorithm, and they suggest that it might be possible to replace this step with a random search around the stationary point found at each temperature. In the following discussion we demonstrate that these adaptations of the DAEM algorithm are unlikely to improve the performance of the DAEM algorithm when applied to naive Bayesian networks, particularly when we are considering networks with binary observable nodes. We begin by considering the following theorems, proofs of which are given in Appendix B.

Theorem 2.4 *Consider a naive Bayesian network with $r_i = 2$ for all i. At the point given in (2.26) the Hessian of $\mathcal{L}_\beta$, denoted by H_β, is of the form*

$$H_\beta = \begin{pmatrix} H_\beta^1 & 0 \\ 0 & H_\beta^2 \end{pmatrix}.$$

where H_β^1 is the $r_z \times r_z$ matrix given by

$$H_\beta^1 = \left(\frac{\partial^2}{\partial p_j \partial p_k} \mathcal{L}_\beta \left(\boldsymbol{\theta}_m \right) \right)$$

and H_β^2 *is the* $nr_z \times nr_z$ *matrix given by*

$$H_\beta^1 = \left(\frac{\partial^2}{\partial \theta_{i|j} \partial \theta_{h|k}} \mathcal{L}_\beta \left(\boldsymbol{\theta}_m \right) \right)$$

Theorem 2.5 *The matrix* H_β^1 *is positive definite.*

Theorem 2.6 *Assuming that our maximum likelihood independence model is correct, the expected value of the matrix* H_β^2 *is positive definite for naive Bayesian networks with binary observable variables.*

Theorem 2.7 *For naive Bayesian networks with binary observable nodes, the Hessian at the point given in (2.26) will be positive definite for*

$$\beta \leq \frac{r_z}{2n\left(r_z - 1\right)}.$$

Theorems 2.6 and 2.7 mean that for naive Bayesian networks with binary observable variables we now have reason to believe that for many datasets the point given in (2.26) may be a local minimum at all values of β as well as being a stationary point. The implication of this is that the DAEM algorithm will be unable to escape from this point even with Ueda and Nakano's proposed improvements. In fact numerical experiments in which a simple naive Bayesian network with a binary root node and three binary observable nodes was used to generate 100 datasets of size 500 for each of $\beta = 0.1, 0.2, \ldots, 0.9$, failed to provide an example where the matrix H_β^2 was not positive definite. In practice

even though our assumption that the maximum likelihood independence model is correct is not true, the off-diagonal terms in H_β^2 were consistently close to zero, meaning that H_β^2 was always diagonally dominant and hence positive definite. For networks with observable variables with more than two possible states the situation is more complicated due to the introduction to the Hessian matrix of the terms

$$\frac{\partial^2 \mathcal{L}_\beta}{\partial \theta_{efg} \partial \theta_{efv}} = \frac{N^2}{r_z^2} \left(\beta + r_z \left(1 - \beta\right)\right) \frac{1}{\sum_m I_{r_e}\left(y_{me}\right)}.$$

Numerical experiments indicated that, even if the observable variables have more than two states, for smaller values of β the point given in (2.26) would still be a local minimum, but that for larger values of β it would be a saddlepoint and the modified algorithm would be able to escape this point. Experiments were conducted using networks with a binary root node and three observable nodes. The three observable nodes were taken to have the same number of states, with the number of states being 2, 3, 4, 5 and 6. Table 2.5 shows the number of times (out of 100) that the Hessian at the point given in (2.26) was found to be positive definite.

These results imply that even the modified algorithm will only be able to escape from the equally weighted independence model at relatively low temperatures. At these lower temperatures we no longer have good reason to believe we are minimising a unimodal surface, which in turn casts serious doubts on any likely benefit to be gained from implementing this computationally intensive algorithm.

β = Temperature^{-1}	Number of states of observable variables				
	2	3	4	5	6
0.1	100	100	100	100	100
0.2	100	100	100	100	100
0.3	100	97	100	100	100
0.4	100	83	93	100	97
0.5	100	3	0	0	22
0.6	100	0	0	0	0
0.7	100	0	0	0	0
0.8	100	0	0	0	0
0.9	100	0	0	0	0

Table 2.5: Table showing the number of times (out of 100) that the Hessian at the point given in (2.26) was found to be positive definite, for a simple Bayesian network with a binary root node and three observable nodes.

Chapter 3

Model Identifiability

3.1 Introduction

When fitting a model to data it is important to consider whether the proposed model is identifiable. A model is said to be identifiable if for two values of our model parameters $\theta_m \neq \theta'_m$ there exists an observation y such that $P(y \mid \theta_m) \neq P(y \mid \theta'_m)$. We also define local identifiability to mean that for two values of our model parameters $\theta_m \neq \theta'_m$ such that θ_m and θ'_m are separated by less than a distance $\delta > 0$ there exists a y such that $P(y \mid \theta_m) \neq P(y \mid \theta'_m)$. If a model is identifiable it means that the parameters of that model are uniquely determined by our observed data. Clearly this is a desirable property as ambiguity in our parameter estimates can lead to difficulty in interpreting our fitted model.

It is well known that when dealing with mixture models there is inevitably a certain degree of non-identifiability. When dealing with a mixture of distributions

from the same family

$$P(y \mid \theta_m) = \sum_{j=1}^{r_z} p_j P(y \mid Z = j, \theta_m)$$

the log-likelihood

$$\mathcal{L}(\theta_m) = \sum_{l=1}^{N} \log [P(y_l \mid \theta_m)]$$

will have $r_z!$ maxima corresponding to permutation of the component labels. In practice this form of non-identifiability is trivial. Formally we remove this non-identifiability by insisting that $p_1 \geq p_2 \geq \ldots \geq p_{r_z}$. Fitting the model subject to this restriction would be somewhat problematic, so initially we fit the model without the restriction after which the component labels are permuted to re-impose our restriction.

Robert and Mengerson [63] took another approach to this particular form of non-identifiability in their work on mixtures of normal distributions. They proposed a re-parameterisation they call "splitting" in which the secondary components of the mixture are expressed in terms of the primary component. The distribution of the first mixture component is defined in terms of a global location-scale parameter, (θ_1, τ_1), as $\mathcal{N}(\theta_1, \tau_1^2)$. The location-scale parameter of the second distribution is then defined in terms of this global location-scale parameter to give the mixture distribution

$$q\mathcal{N}(\theta_1, \tau_1) + (1-q)\mathcal{N}\left(\theta_1 + \tau_1\theta_2, \tau_1^2\tau_2^2\right).$$

Subsequent components of the mixture distribution are then defined in terms of the location-scale parameter of the previous component, and hence a r_z

component mixture is expressed as

$$q_1\mathcal{N}(\theta_1,\tau_1^2)+\sum_{i=2}^{r_z-1}(1-q_1)\ldots(1-q_{i-1})\,q_i\mathcal{N}(\theta_1+\ldots+\tau_1\ldots\tau_{i-1}\theta_i,\tau_1^2\ldots\tau_i^2)$$
$$+(1-q_1)(1-q_2)\ldots(1-q_{r_z-2})\mathcal{N}(\theta_1+\ldots+\tau_1\ldots\tau_{r_z-1}\theta_{r_z},\tau_1^2\ldots\tau_{r_z}^2) \tag{3.1}$$

As with the more usual parameterisation, this model is invariant under permutation of the indices and so again a constraint must be imposed. The restriction imposed is that the the variances σ_i^2 are decreasing with i, which in this parameterisation can be equivalently stated as

$$\tau_j \leq 1 \qquad \forall j = 2,\ldots,r_z.$$

Although this results in a much more complicated parameterisation of the model, Robert and Mengersen demonstrate that it leads to improved convergence properties for the Gibbs sampler used for Bayesian inference about the parameters.

The issue of identifiability was first investigated by Teicher in 1961 [36], who was able to prove that a number of continuous finite mixture distributions are identifiable. However, it has long been recognised that identifiability problems exist when working with discrete data. Consider for example the simple naive Bayesian network with a binary root node and two observable binary nodes as shown in Figure 3.1. The probability of an observation $\boldsymbol{y}$ for this model is

$$\begin{aligned} P(\boldsymbol{y}\mid\boldsymbol{\theta_m}) &= p\theta_{1|1}^{y_1}\left(1-\theta_{1|1}\right)^{(1-y_2)}\theta_{2|1}^{y_2}\left(1-\theta_{2|1}\right)^{(1-y_2)} \\ &\quad +(1-p)\,\theta_{1|2}^{y_1}\left(1-\theta_{1|2}\right)^{(1-y_1)}\theta_{2|2}^{y_2}\left(1-\theta_{2|2}\right)^{(1-y_2)}. \end{aligned}$$

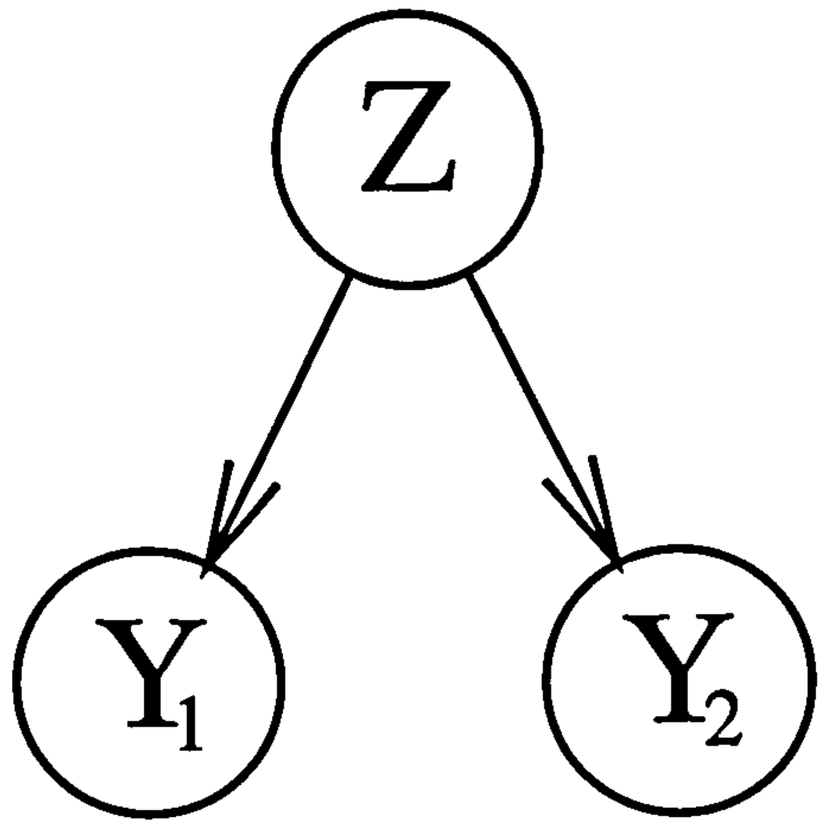


Figure 3.1: A naive Bayesian network with a binary root node and two binary observable nodes.

Clearly for this model there are 4 possible outcomes that may be observed, namely (Y_1, Y_2) must be one of $\{(0,0),(0,1)(1,0),(1,1)\}$. We can assign probabilities to each of these outcomes

$$\begin{aligned} P((Y_1,Y_2)=(0,0)) &= \omega_0 \\ P((Y_1,Y_2)=(0,1)) &= \omega_1 \\ P((Y_1,Y_2)=(1,0)) &= \omega_2 \\ P((Y_1,Y_2)=(1,1)) &= \omega_3 \end{aligned}$$

where $\sum_i \omega_i = 1$. The ω's are known as the parameters of the observable variables, and in this case there are 3 such independent parameters. Maximum likelihood estimates for the parameters of the observable variables can easily be determined; in this case $\{\hat{\omega}_0, \hat{\omega}_1, \hat{\omega}_2, \hat{\omega}_3\}$ could be found by counting up the number of times we observe each of the possible combination of outputs and dividing those counts by the total sample size. This is all very well, but we are

really interested in the parameters of our network. Denoting the ML estimates of the network parameters by $\left\{\hat{p}, \hat{\theta}_{1|0}, \hat{\theta}_{1|1}, \hat{\theta}_{2|0}, \hat{\theta}_{2|1}\right\}$ it must follow that

$$\begin{aligned}
\hat{p}\left(1-\hat{\theta}_{1|0}\right)\hat{\theta}_{2|0} + (1-\hat{p})\left(1-\hat{\theta}_{1|1}\right)\hat{\theta}_{2|1} &= \hat{\omega}_1 \\
\hat{p}\hat{\theta}_{1|0}\left(1-\hat{\theta}_{2|0}\right) + (1-\hat{p})\,\hat{\theta}_{1|1}\left(1-\hat{\theta}_{2|1}\right) &= \hat{\omega}_2 \\
\hat{p}\hat{\theta}_{1|0}\hat{\theta}_{2|0} + (1-\hat{p})\,\hat{\theta}_{1|1}\hat{\theta}_{2|1} &= \hat{\omega}_3
\end{aligned}$$

Here we have a network with 5 unknown parameters and yet only 3 independent equations with which to determine estimates for these parameters. Clearly this model is not identifiable. When dealing with non-identifiable models it is possible to re-parameterise them so that they are identifiable. Here the most obvious way of doing so would be to work instead in the space of the parameters of the observable variables. However, any re-parameterisation will mean that we will not be able to determine anything about the latent structure of this problem.

In a naive Bayesian network we will always have fewer independent equations than we have parameters if

$$\prod_{i=1}^{n} r_i - 1 < r_z \sum_{i=1}^{n} (r_i - 1) + r_z - 1$$

or, in the case of naive Bayesian networks with only binary observable nodes, if

$$2^n - 1 < r_z n + r_z - 1$$

This gives us a necessary, though possibly not sufficient, condition for

identifiability in naive Bayesian networks. Goodman's 1974 paper [29] on maximum likelihood estimation in the m-way contingency table provides us with tools for determining the identifiability of models which do not satisfy this condition for obvious non-identifiability. He showed that the rank of the Jacobian matrix for the transformation between the model parameters and the parameters of the observable variables is equal to the dimension of the parameter space of the model. Hence a model is identifiable if and only if this matrix is of full rank. It is obviously desirable for this matrix to be of full rank for any value of our model parameters $\boldsymbol{\theta_m}$; this is known as essentially full rank. When our model is essentially full rank identifiability is less of a concern, but we will still find that for some incomplete data sets we will find that our fitted model is non-identifiable. However, naive Bayesian networks are only conditionally full rank models, meaning that for certain restrictions on our parameters the model will be 'parameter redundant'. Consider a naive Bayesian network with a binary root node and 3 observable binary nodes; the Jacobian for this model is

$$\begin{pmatrix}
\theta_{1|1}\theta_{2|1}\theta_{3|1} - \theta_{1|2}\theta_{2|2}\theta_{3|2} & p\theta_{2|1}\theta_{3|1} & p\theta_{1|1}\theta_{3|1} & p\theta_{1|1}\theta_{2|1} \\
\left(1-\theta_{1|1}\right)\theta_{2|1}\theta_{3|1} - \left(1-\theta_{1|2}\right)\theta_{2|2}\theta_{3|2} & -p\theta_{2|1}\theta_{3|1} & p\left(1-\theta_{1|1}\right)\theta_{3|1} & p\left(1-\theta_{1|1}\right)\theta_{2|1} \\
\theta_{1|1}\left(1-\theta_{2|1}\right)\theta_{3|1} - \theta_{1|2}\left(1-\theta_{2|2}\right)\theta_{3|2} & p\left(1-\theta_{2|1}\right)\theta_{3|1} & -p\theta_{1|1}\theta_{3|1} & p\theta_{1|1}\left(1-\theta_{2|1}\right) \\
\left(1-\theta_{1|1}\right)\left(1-\theta_{2|1}\right)\theta_{3|1} - \left(1-\theta_{1|2}\right)\left(1-\theta_{2|2}\right)\theta_{3|2} & -p\left(1-\theta_{2|1}\right)\theta_{3|1} & -p\left(1-\theta_{1|1}\right)\theta_{3|1} & p\left(1-\theta_{1|1}\right)\left(1-\theta_{2|1}\right) \\
\theta_{1|1}\theta_{2|1}\left(1-\theta_{3|1}\right) - \theta_{1|2}\theta_{2|2}\left(1-\theta_{3|2}\right) & p\theta_{2|1}\left(1-\theta_{3|1}\right) & p\theta_{1|1}\left(1-\theta_{3|1}\right) & -p\theta_{1|1}\theta_{2|1} \\
\left(1-\theta_{1|1}\right)\theta_{2|1}\left(1-\theta_{3|1}\right) - \left(1-\theta_{1|2}\right)\theta_{2|2}\left(1-\theta_{3|2}\right) & -p\theta_{2|1}\left(1-\theta_{3|1}\right) & p\left(1-\theta_{1|1}\right)\left(1-\theta_{3|1}\right) & -p\left(1-\theta_{1|1}\right)\theta_{2|1} \\
\theta_{1|1}\left(1-\theta_{2|1}\right)\left(1-\theta_{3|1}\right) - \theta_{1|2}\left(1-\theta_{2|2}\right)\left(1-\theta_{3|2}\right) & p\left(1-\theta_{2|1}\right)\left(1-\theta_{3|1}\right) & -p\theta_{1|1}\left(1-\theta_{3|1}\right) & -p\theta_{1|1}\left(1-\theta_{2|1}\right)
\end{pmatrix}$$

$$\begin{pmatrix}
(1-p)\theta_{2|2}\theta_{3|2} & (1-p)\theta_{1|2}\theta_{3|2} & (1-p)\theta_{1|2}\theta_{2|2} \\
-(1-p)\theta_{2|2}\theta_{3|2} & (1-p)\left(1-\theta_{1|2}\right)\theta_{3|2} & (1-p)\left(1-\theta_{1|2}\right)\theta_{2|2} \\
(1-p)\left(1-\theta_{2|2}\right)\theta_{3|2} & -(1-p)\theta_{1|2}\theta_{3|2} & (1-p)\theta_{1|2}\left(1-\theta_{2|2}\right) \\
-(1-p)\left(1-\theta_{2|2}\right)\theta_{3|2} & -(1-p)\left(1-\theta_{1|2}\right)\theta_{3|2} & (1-p)\left(1-\theta_{1|2}\right)\left(1-\theta_{2|2}\right) \\
(1-p)\theta_{2|2}\left(1-\theta_{3|2}\right) & (1-p)\theta_{1|2}\left(1-\theta_{3|2}\right) & -(1-p)\theta_{1|2}\theta_{2|2} \\
-(1-p)\theta_{2|2}\left(1-\theta_{3|2}\right) & (1-p)\left(1-\theta_{1|2}\right)\left(1-\theta_{3|2}\right) & -(1-p)\left(1-\theta_{1|2}\right)\theta_{2|2} \\
(1-p)\left(1-\theta_{2|2}\right)\left(1-\theta_{3|2}\right) & -(1-p)\theta_{1|2}\left(1-\theta_{3|2}\right) & -(1-p)\theta_{1|2}\left(1-\theta_{2|2}\right)
\end{pmatrix}$$

and is of full rank rank for general $\boldsymbol{\theta_m}$. However the parameter space

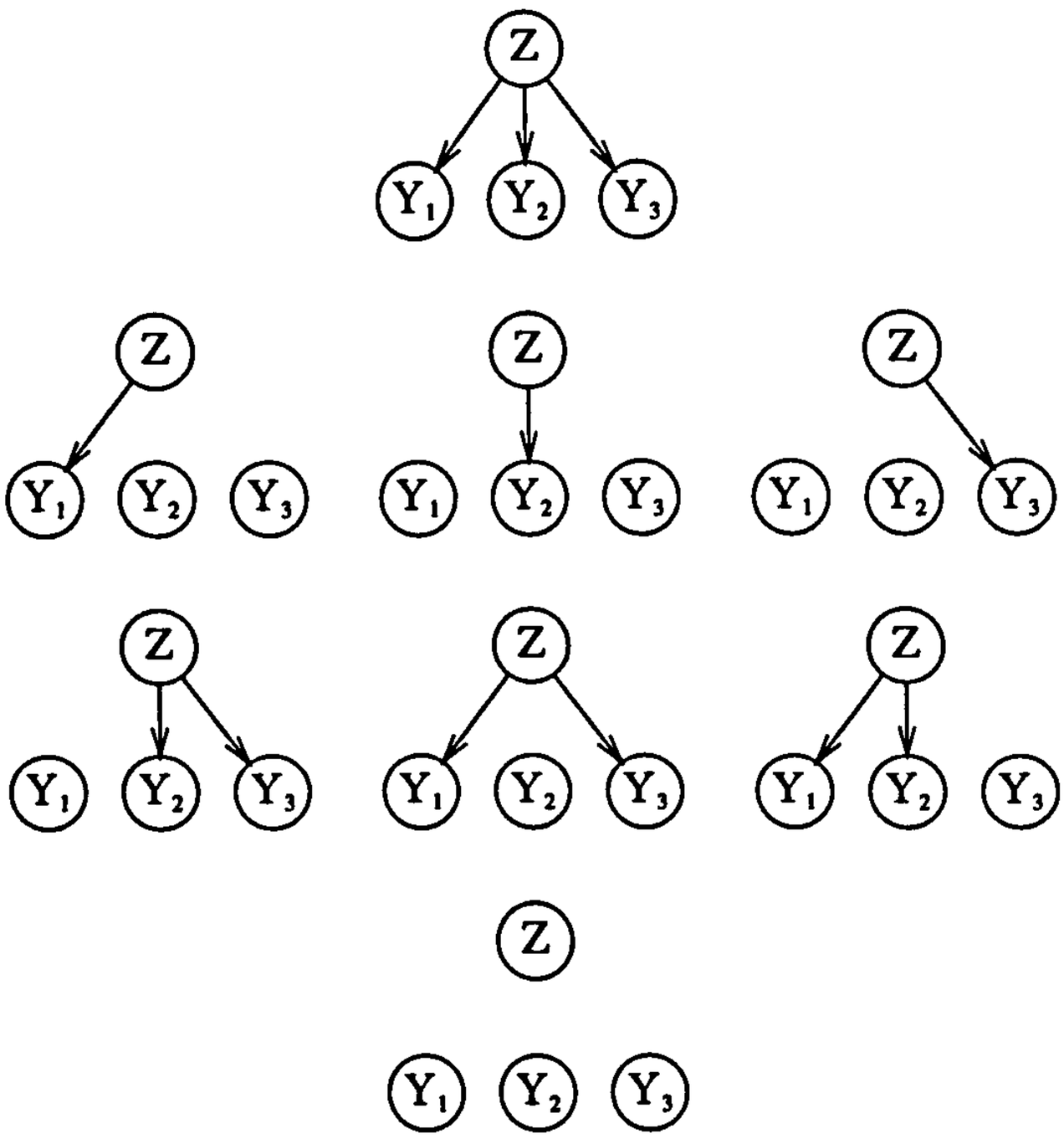


Figure 3.2: The naive Bayesian network with 1 binary root node and 3 binary observable nodes (shown top) contains within its parameter space the 7 non-identifiable models shown above.

for this model also contains a number of non-identifiable models, indicating that for certain parameter values our model is over-parameterised. This over-parameterisation can take two forms. There will be cases in which two or more components of the mixture are identical, in which case we can clearly represent our population with a model with fewer latent states, or we might find that the model we end up with in fact corresponds to a model with a simpler structure. This second identifiability problem is illustrated in Figure 3.2 which shows how a naive Bayesian network with a binary root node and three observable variables, which is a conditionally identifiable model, contains within its parameter space points corresponding to models with a simpler network structure.

As naive Bayesian networks are only conditionally full rank models, it would be a sensible precaution to check the rank of the Jacobian for the final fitted model as it is possible that, depending upon our dataset, our fitted model might be non-identifiable. If the fitted model proves to be non-identifiable then it might be necessary to re-parameterise the model, or to fit a network with a simpler structure. We conjecture that the only points in our parameter space that will lead to identifiability problems will correspond with those simpler models known to be present within the parameter space, and although we are unable to prove the truth of this conjecture we have yet to see a counter example.

Since Goodman's paper there seems to have been little work done on addressing the issue of identifiability in models for discrete data until the more recent work of Geiger *et al.* [24] and Catchpole and Morgan [9], who have independently rediscovered Goodman's methodology.

Catchpole and Morgan are working in the area of animal survival estimation, studying mark-recapture and ring-recovery models. They cite a simple example of a ring-recovery experiment in which the recovery of rings from birds ringed as nestlings is considered. Using p_{ij} to denote the probability that a bird ringed in year i dies and has its ring recovered in year j then a model for an experiment with $r = 2$ years of ringing and $c = 3$ years of recovery is

$$P = \begin{pmatrix} (\phi_{11})\,\lambda_1 & \phi_{11}\,(1-\phi_2)\,\lambda_2 & \phi_{11}\phi_2\,(1-\phi_3)\,\lambda_3 \\ 0 & (1-\phi_{12})\,\lambda_2 & \phi_{12}\,(1-\phi_2)\,\lambda_3 \end{pmatrix}$$

where ϕ_{1i} is the probability of a bird ringed in year i surviving its first year of life, ϕ_k is the conditional probability of surviving the k^{th} year of life having

survived the first year and λ_j is the probability that a bird dies in the j^{th} year of the study and is recovered. Catchpole and Morgan advocate the use of symbolic algebra packages to determine the identifiability of such models. This seems to work adequately for these ring-recapture models which have Jacobians with a reasonably simple structure. The simplicity of the Jacobians of their models also enables them to develop an 'extension theorem' for this class of models.

Theorem 3.1 ('Extension Theorem' from Catchpole and Morgan [9]) *Suppose that a ring-recapture model, with parameter vector $\theta = (\theta_1, \ldots, \theta_p)$ is full rank for an $r \times c$ table. Let $r' \geq r$ and $c' \geq c$. Suppose that, for an $r' \times c'$ experiment, the extension of the table by one row leads to the inclusion of extra parameters $\psi = (\theta_{p+1}, \ldots, \theta_{p+v})$. Regard this extra row as a function of ψ only, and form its derivative matrix. Now repeat this procedure for an extension by one column. If both of these subsidiary derivative matrices are full rank, then the model is full rank for any $r' \times c'$ table with $r' \geq r$ and $c' \geq c$.*

The idea behind the proof of this theorem is that if we take the Jacobian of the original model to be J and of the extended model to be J' then ring-recovery models are such that

$$J' = \begin{pmatrix} J & C \\ 0 & A \end{pmatrix}$$

where A is the subsidiary derivative matrix referred to in the theorem. Clearly if J and A are of full rank then J' must also be of full rank, as a consequence of the zeros in the bottom left quadrant of J'.

Geiger *et al.* are working with Bayesian networks. These models have a more complicated structure than ring-recovery models and it is often beyond the

capability of symbolic algebra packages to calculate the rank of the Jacobian for all but the simplest Bayesian networks. This has led them to develop numerical techniques for determining the dimension of the parameter space of their models. They prove the following theorem.

Theorem 3.2 *Let $\boldsymbol{\theta}$ be the parameters of a network $\boldsymbol{S}$ for variables $\boldsymbol{X}$ with observable variables $\boldsymbol{O} \subset \boldsymbol{S}$. Let $\boldsymbol{w}$ be the parameters of the true joint distribution of the observable variables. If each parameter in $\boldsymbol{w}$ is a polynomial function of $\boldsymbol{\theta}$, then rank $\left[\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{w}}(\theta)\right] = d$ almost everywhere, where d is a constant.*

This leads them to argue that a random algorithm can be used to calculate the rank of the Jacobian. They suggest assigning a random value to $\boldsymbol{\theta}$ and diagonalising the numerical Jacobian. They state that the above theorem guarantees that, with probability 1, the resulting rank is the regular rank of the Jacobian. Table 3.1 shows the results of applying this random algorithm to a number of naive Bayesian networks with binary observable nodes. It shows that all models satisfying the necessary condition for identifiability are indeed identifiable. However, the computational expense in determining the ranks of these matrices is considerable. The largest model in this table has a Jacobian containing more than 20 million elements, and finding the rank of this matrix took many hours of computation. More generally when fitting Bayesian belief networks we would need to perform this diagonalisation of the numerical Jacobian for every new model considered, an onerous task. This indicates that something akin to the extension theorem of Catchpole and Morgan [9] is desirable, particularly when dealing with more general networks.

Geiger *et al.* [24] prove the following result.

Number of Binary Observable Variables \ Number of Latent States	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
16	16	33	50	67	84	101	118	135	152	169	186	203	220	237	254	271	288	305	322	339
15	15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255	271	287	303	319
14	14	29	44	59	74	89	104	119	134	149	164	179	194	209	224	239	254	269	284	299
13	13	27	41	55	69	83	97	111	125	139	153	167	181	195	209	223	237	251	265	279
12	12	25	38	51	64	77	90	103	116	129	142	155	168	181	194	207	220	233	246	259
11	11	23	35	47	59	71	83	95	107	119	131	143	155	167	179	191	203	215	227	239
10	10	21	32	43	54	65	76	87	98	109	120	131	142	153	164	175	186	197	208	219
9	9	19	29	39	49	59	69	79	89	99	109	119	129	139	149	159	169	179	189	199
8	8	17	26	35	44	53	62	71	80	89	98	107	116	125	134	143	152	161	170	179
7	7	15	23	31	29	47	55	63	71	79	87	95	103	111	119	127	127	127	127	127
6	6	13	20	27	34	41	48	55	62	63	63	63	63	63	63	63	63	63	63	63
5	5	11	17	23	29	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31
4	4	9	14	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
3	3	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 3.1: Table showing the dimension of the parameter space of a number of simple naive Bayesian networks. Shaded table entries indicate that that particular model is non-identifiable, whereas non-shaded entries correspond to models which are conditionally identifiable.

Theorem 3.3 *Let S be a naive Bayesian network with one binary hidden root node and $n > 2$ binary observable non-root nodes. Then*

$$2n \leq r \leq 2n + 1,$$

where r is the regular rank of the Jacobian matrix between the parameters of the network and the parameters of the feature variables.

They also conjecture that the stricter condition holds that in fact $1 + 2n$ is the regular rank for all $n > 2$. We now offer a proof of this conjecture.

3.2 Proof of Geiger *et al.*'s Conjecture

Theorem 3.4 *Let m_n be a naive Bayes model with one binary hidden root node and $n > 2$ binary observed non-root nodes. Then, if the Jacobian matrix between the parameters of the network and the parameters of the feature variables is of full rank for $n = r$, it must also be of full rank for $n = r + 1$.*

Proof of Theorem 3.4

For our model m_n we denote the parameters of our network by θ_{m_n}, where θ_{m_n} consists of

$$\begin{aligned} p &= P(Z = 0) \\ \theta_{i|j} &= P(Y_i = 1 \mid Z = j) \qquad i = 1, \ldots, n \quad j = 0, 1, \end{aligned}$$

and we use $\omega_1^n, \omega_2^n, \ldots, \omega_{2^n}^n$ to denote the parameters of the observable variables. If we take B to be the value of the binary number represented by $\left[Y_n \vdots Y_{n-1} \vdots \cdots \vdots Y_1\right]$ (note the reverse ordering of the Y_i) then

$$\omega_l^n = P(B = 2^n - l) \qquad l = 1, \ldots, 2^n.$$

We also define y_l to be the values of the Y_i necessary for $B = 2^n - l$, and y_{li} to be the value of Y_i in y_l. The Jacobian matrix of our model is denoted by J_n, the columns of J_n corresponding to the network variables $p, \theta_{1|0}, \ldots, \theta_{n|0}, \theta_{1|1}, \ldots, \theta_{n|1}$ and the rows corresponding to $\omega_1^n, \ldots, \omega_{2^n-1}^n$. It is not necessary to include the final row, corresponding to $\omega_{2^n}^n$, as its omission will not affect the column rank of the Jacobian; see Goodman [29].

We then take the Jacobians for the models m_r and m_{r+1} and use them to produce two new matrices, $\hat{J}_r$ and $\hat{J}_{r+1}$ respectively, using elementary column operations.

To produce $\hat{J}_r$ we first re-order the columns of J_r so that they correspond to differentiation by $\theta_{1|0}, \ldots, \theta_{r|0}, p, \theta_{1|1}, \ldots, \theta_{r|1}$ and we then divide columns 1 to r by p and columns $(r+2)$ to $(2r+1)$ by $(1-p)$. To produce $\hat{J}_{r+1}$ from J_{r+1} we divide columns 2 to $(r+2)$ by p, and columns $(r+3)$ to $(2r+3)$ by $(1-p)$. We also divide columns 2 to $(r+1)$ by $\theta_{r+1|0}$ and columns $(r+3)$ to $(2r+2)$ by $\theta_{r+1|1}$. Finally, we subtract the $(2r+3)^{rd}$ column of this matrix from the $(r+2)^{nd}$.

Since $\hat{J}_r$ is derived from J_r by simple column operations it must follow that J_r being of full column rank implies that $\hat{J}_r$ is also of full column rank, and vice versa. This must be true because J_r being of full column rank means that

its $2r+1$ columns must be linearly independent. This means that any $2r+1$ columns which consist of simple combinations of these columns must also be linearly independent. Hence it follows that $\hat{J}_r$ is of full rank. The same argument also holds for J_{r+1} and $\hat{J}_{r+1}$. This means that in order to prove our theorem it is sufficient to demonstrate that $\hat{J}_r$ being of full column rank implies that $\hat{J}_{r+1}$ is also of full column rank.

We begin by showing that $\hat{J}_{r+1}$ can be written in the form

$$\hat{J}_{r+1} = \begin{pmatrix} & \hat{J}_r & u_r \\ W_r & \hat{\hat{J}}_r & v_r \\ & K_r & -u_r \end{pmatrix}$$

(row heights: 2^r-1, 1, 2^r-1; total $2^{r+1}-1$)

(column widths: 1, $2r+1$, 1; total $2(r+1)+1$)

Matrix 3.1

We use $\begin{pmatrix} J_r \\ j_r \end{pmatrix}$ to represent the Jacobian matrix for the model m_r without the omission of the last row, and $\begin{pmatrix} \hat{J}_r \\ \hat{j}_r \end{pmatrix}$ to represent the result of applying the column operations for J_r, described above, to this augmented matrix.

The first 2^r rows of J_{r+1} will correspond to all possible combinations of the observed variables $Y_1, \ldots, Y_r$, in the same order that they appear in $\begin{pmatrix} J_r \\ j_r \end{pmatrix}$, with

$Y_{r+1} = 1$ in each case. Therefore for l in the range 1 to 2^r we have

$$\begin{aligned}
\omega_l^r &= p\theta_{1|0}^{y_{l1}}\left(1-\theta_{1|0}\right)^{(1-y_{l1})}\ldots\theta_{r|0}^{y_{lr}}\left(1-\theta_{r|0}\right)^{(1-y_{lr})} \\
&\quad + (1-p)\,\theta_{1|1}^{y_{l1}}\left(1-\theta_{1|1}\right)^{(1-y_{l1})}\ldots\theta_{r|1}^{y_{lr}}\left(1-\theta_{r|1}\right)^{(1-y_{lr})} \\
\omega_l^{r+1} &= p\theta_{1|0}^{y_{l1}}\left(1-\theta_{1|0}\right)^{(1-y_{l1})}\ldots\theta_{r|0}^{y_{lr}}\left(1-\theta_{r0}\right)^{(1-y_{lr})}\theta_{r+1|0} \\
&\quad + (1-p)\,\theta_{1|1}^{y_{l1}}\left(1-\theta_{1|1}\right)^{(1-y_{l1})}\ldots\theta_{r|1}^{y_{lr}}\left(1-\theta_{r|1}\right)^{(1-y_{lr})}\theta_{r+1|1}.
\end{aligned} \tag{3.2}$$

In order to show the correspondence between $\begin{pmatrix}\hat{J}_r\\ \hat{j}_r\end{pmatrix}$ and $\hat{J}_{r+1}$ we first consider columns 1 to r of $\begin{pmatrix}\hat{J}_r\\ \hat{j}_r\end{pmatrix}$, we then consider columns $r+2$ to $2r+1$ and finally we consider column $r+1$.

The lth element of column $k+1$ of $\hat{J}_{r+1}$, where $k = 1,\ldots,r$, is

$$\frac{1}{p}\frac{1}{\theta_{r+1|0}}\frac{\partial\omega_l^{r+1}}{\partial\theta_{k|0}},$$

and the lth element of column k of $\begin{pmatrix}\hat{J}_r\\ \hat{j}_r\end{pmatrix}$, $k = 1,\ldots,r$, is

$$\frac{1}{p}\frac{\partial\omega_l^{r+1}}{\partial\theta_{k|0}}.$$

Thus, using the relationships given in (3.2) it must follow that the top 2^r rows of the 2nd to $(r+1)^{st}$ columns of $\hat{J}_{r+1}$ must be the same as columns 1 to r of $\begin{pmatrix}\hat{J}_r\\ \hat{j}_r\end{pmatrix}$. A similar argument shows that columns $r+2$ to $2r+1$ of $\begin{pmatrix}\hat{J}_r\\ \hat{j}_r\end{pmatrix}$ correspond to the top 2^r-1 elements of columns $r+3$ to $2r+3$ of $\hat{J}_{r+1}$. Finally, the lth element

of column $r+2$ of $\hat{J}_{r+1}$ is

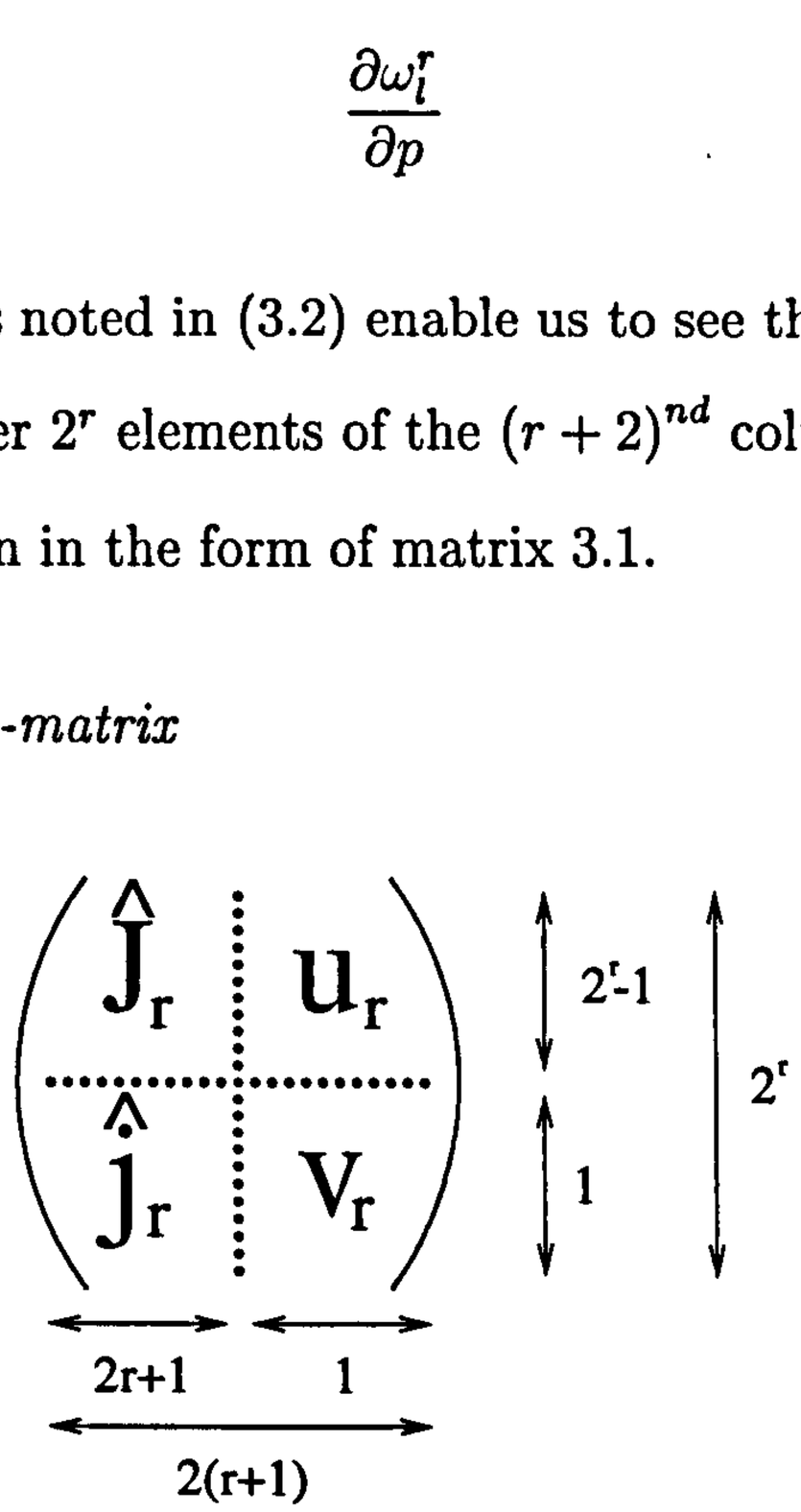
$$\frac{1}{p}\frac{\partial \omega_l^{r+1}}{\partial \theta_{r+1|0}} - \frac{1}{(1-p)}\frac{\partial \omega_l^{r+1}}{\partial \theta_{r+1|1}}$$

and the lth element of column $r+1$ of $\begin{pmatrix} \hat{J}_r \\ \hat{j}_r \end{pmatrix}$ is

$$\frac{\partial \omega_l^r}{\partial p}$$

Again, the relationships noted in (3.2) enable us to see that this column of $\begin{pmatrix} \hat{J}_r \\ \hat{j}_r \end{pmatrix}$ corresponds to the upper 2^r elements of the $(r+2)^{nd}$ column of $\hat{J}_{r+1}$, and hence that $\hat{J}_{r+1}$ can be written in the form of matrix 3.1.

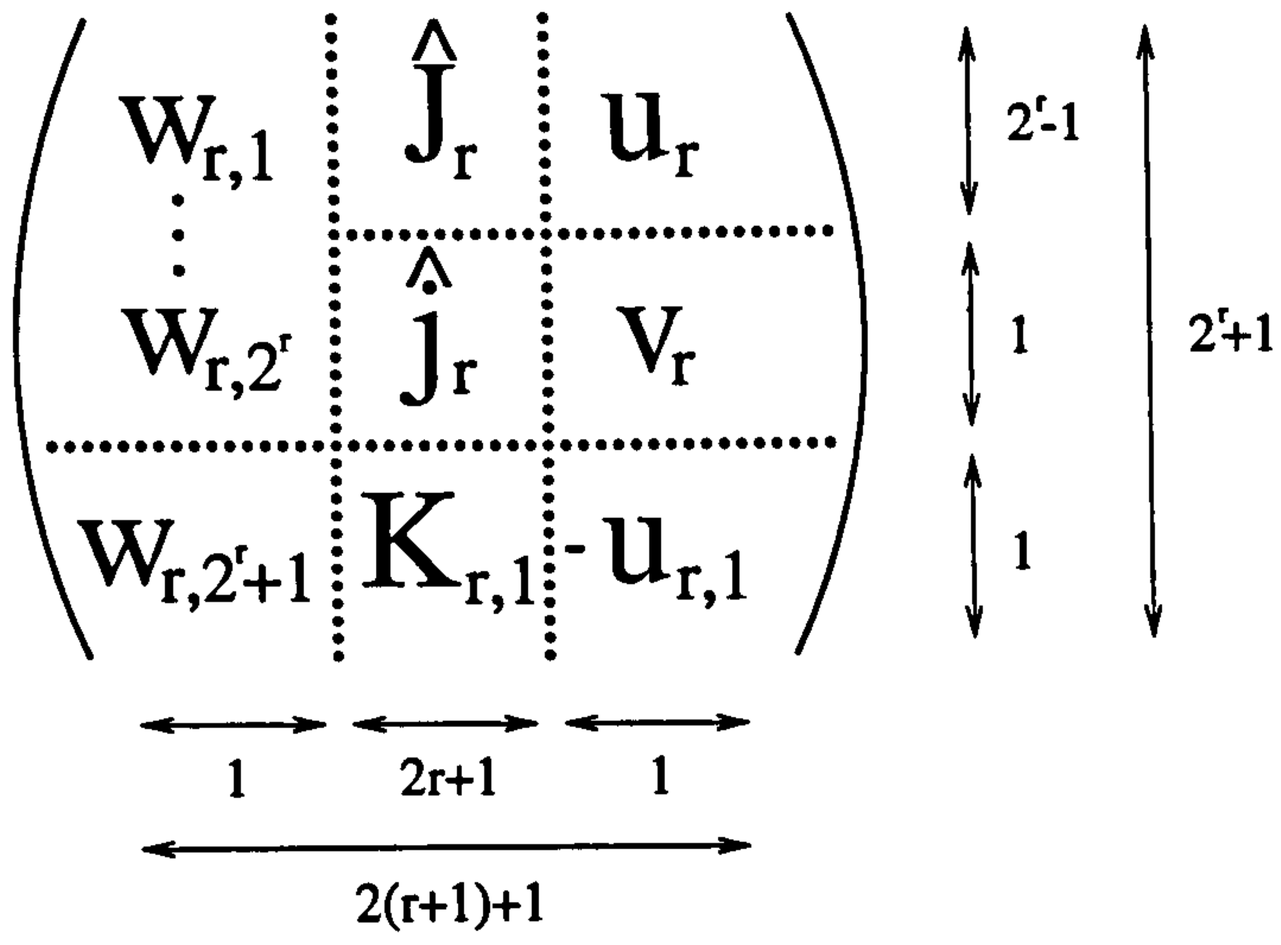
Theorem 3.5 *The sub-matrix*



Matrix 3.2

is of full column rank.

Theorem 3.6 *The sub-matrix*



Matrix 3.3

where

$w_{r,i}$ *is the ith element of the vector* w_r

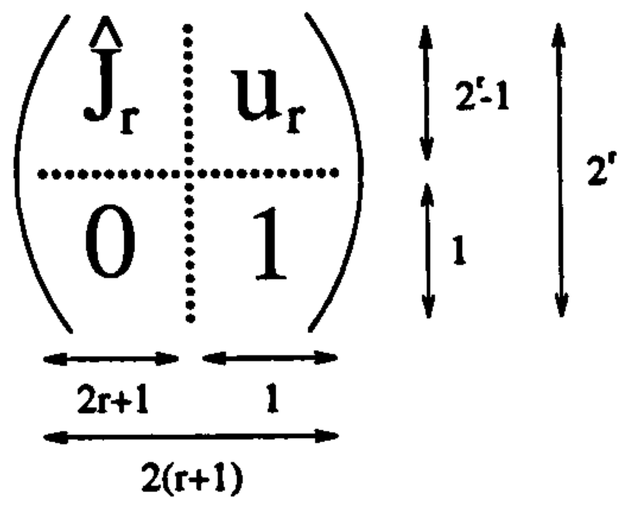
$K_{r,i}$ *is the ith row of the matrix* K_r

$u_{r,i}$ *is the ith element of the vector* u_r,

is of full column rank.

Proof of Theorem 3.5

Adding the upper $2^r - 1$ rows to the final row of matrix 3.2 we get the matrix



Matrix 3.4

To see this, first consider what happens when we add the rows of $\hat{J}_r$ to $\hat{j}_r$. We know that each column of $\begin{pmatrix} J_r \\ j_r \end{pmatrix}$ sums to zero, because

$$\sum_{l=1}^{2^r} \omega_l^r = 1$$

and hence

$$\sum_{l=1}^{2^r} \frac{\partial \omega_l^r}{\partial \epsilon} = 0,$$

where for convenience of notation we use ϵ to mean any one of $\{p, \theta_{1|0}, \ldots, \theta_{r|0}, \theta_{1|1}, \ldots, \theta_{r|1}\}$. Hence, because $\begin{pmatrix} \hat{J}_r \\ \hat{j}_r \end{pmatrix}$ is constructed from $\begin{pmatrix} J_r \\ j_r \end{pmatrix}$ using simple column operations we must find that when we add the rows of $\hat{J}_r$ to $\hat{j}_r$ we will get **0**.

The sum of the column $\begin{pmatrix} u_r \\ v_r \end{pmatrix}$ is given by

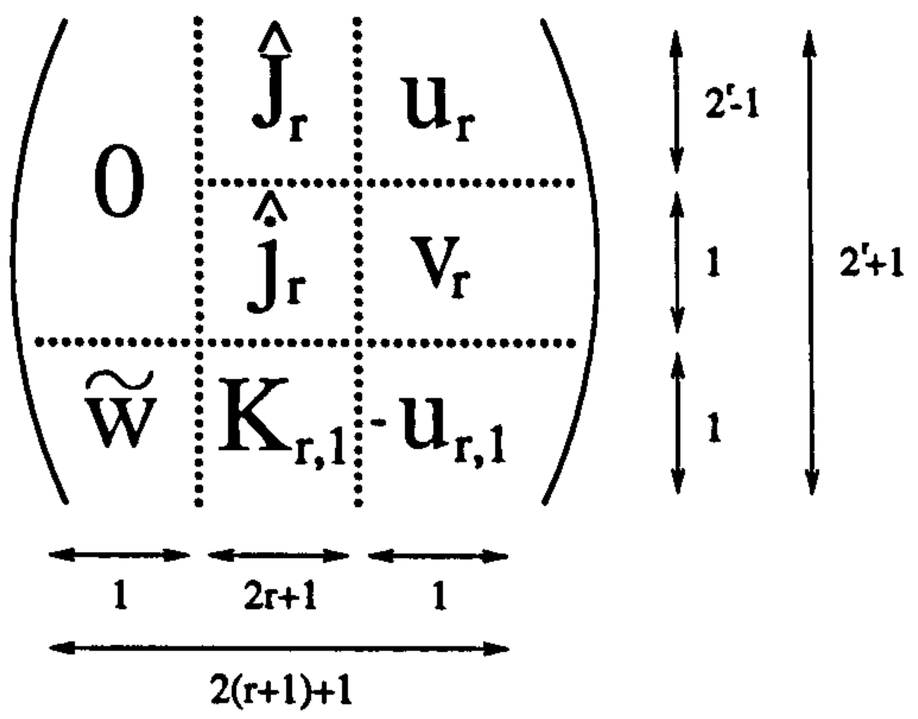
$$\frac{1}{(1-p)} \sum_{l=1}^{2^r} \frac{\partial \omega_l^{r+1}}{\partial \theta_{r+1|1}} = \sum_{l=1}^{2^r} \theta_{1|1}^{y_{l1}} \left(1 - \theta_{1|1}\right)^{(1-y_{l1})} \ldots \theta_{r|1}^{y_{lr}} \left(1 - \theta_{r|1}\right)^{(1-y_{lr})}.$$

The right hand side of this equation is equivalent to $\sum_{i=1}^{2^r} \omega_i^r$ in the case $p = 0$, and we know that this expression must sum to 1.

Since this simple manipulation of sub-matrix 3.2 results in matrix 3.4 we can say that sub-matrix 3.2 must be of full column rank. This follows because in matrix 3.4 we already know that the first $2r+1$ columns are linearly independent as a result of J_r being full rank, and it is clear from the form of the last row of matrix 3.4 that the final column must also be independent of these.

Proof of Theorem 3.6

Subtracting $\theta_{r+1|1}$ times the $(r+2)$nd column from the first column, and adding $(\theta_{r+1|2} + \theta_{r+1|1})$ times the $(2r+3)$rd column gives the matrix



Matrix 3.5

For $1 \leq i \leq 2^r$ the ith element of the $(r+2)$nd column of $\hat{J}_{r+1}$ is

$$\frac{1}{p}\frac{\partial \omega_i^{r+1}}{\partial \theta_{r+1|0}} - \frac{1}{(1-p)}\frac{\partial \omega_i^{r+1}}{\partial \theta_{r+1|1}},$$

of the $(2r+3)$rd column is

$$\frac{1}{(1-p)}\frac{\partial \omega_i^{r+1}}{\partial \theta_{r+1|1}}$$

and of the 1st column is

$$\frac{\partial \omega_i^{r+1}}{\partial p}.$$

Using (3.2) we can show that for $i = 1, \ldots, 2^r$ we have

$$\frac{\partial \omega_i^{r+1}}{\partial p} = \frac{1}{p}\frac{\partial \omega_i^{r+1}}{\partial \theta_{r+1|0}} - \frac{1}{(1-p)}\frac{\partial \omega_i^{r+1}}{\partial \theta_{r+1|1}}.$$

Hence it follows that the combination of columns specified above must have the desired effect.

Finally it is necessary only to show that $\tilde{w}$ is non-zero. The $(2^r + 1)$st row of $\hat{J}_{r+1}$ corresponds to the parameter

$$\omega_{r+1}^{r+1} = \zeta\theta_{1|0}\theta_{2|0}\ldots\theta_{r|0}\left(1-\theta_{r+1|0}\right) + (1-p)\,\theta_{1|1}\theta_{2|1}\ldots\theta_{r|1}\left(1-\theta_{r+1|1}\right)$$

so

$$\begin{aligned}
\tilde{w} &= \frac{\partial \omega_{2r+1}^{r+1}}{\partial p} - \theta_{r+1|0}\left(\frac{1}{p}\frac{\partial \omega_{2r+1}^{r+1}}{\partial \theta_{r+1|0}} - \frac{1}{(1-p)}\frac{\partial \omega_{2r+1}^{r+1}}{\partial \theta_{r+1|1}}\right) + \left(\theta_{r+1|1} + \theta_{r+1|0}\right)\frac{1}{(1-p)}\frac{\partial \omega_{2r+1}^{r+1}}{\partial \theta_{r+1|1}} \\
&= \theta_{1|0}\theta_{2|0}\ldots\theta_{r|0}\left(1-\theta_{r+1|0}\right) - \theta_{1|1}\theta_{2|1}\ldots\theta_{r|1}\left(1-\theta_{r+1|1}\right) \\
&\quad +\theta_{1|0}\theta_{2|0}\ldots\theta_{r|0}\theta_{r+1|0} - \theta_{1|1}\theta_{2|1}\ldots\theta_{r|1}\theta_{r+1|0} \\
&\quad -\left(\theta_{r+1|1} + \theta_{r+1|0}\right)\theta_{1|1}\theta_{2|1}\ldots\theta_{r|1} \\
&\neq 0.
\end{aligned}$$

Since we know from Theorem 3.5 that matrix 3.4 is full rank and it is now clear that the first column of matrix 3.3 is linearly independent of the others, we can now see that matrix 3.3 must be of full column rank.

Matrix 3.3 being of full column rank means that the same must be true of

$\hat{J}_{r+1}$. Hence we have shown that $\hat{J}_r$ being of full rank must imply that $\hat{J}_{r+1}$ is also of full rank, and our theorem is proved. It is perhaps somewhat difficult to grasp the proof as it stands, and the following pages give an example of the method of the proof applied to two simple matrices.

Example

Here we take $r = 3$, so we are using the fact that the model with 3 binary observable variables is identifiable to show that the matrix with 4 binary observable variables must also be identifiable. The following pages show the matrices J_4, $\hat{J}_4$, J_3 and $\hat{J}_3$. Within the matrix $\hat{J}_4$ the part of that matrix corresponding to $\hat{J}_3$ is shaded. Finally the matrices w_r, j_r, v_r and K_r are also given.

$$
J_4 = \begin{pmatrix}
\theta_{1|0}\theta_{2|0}\theta_{3|0}\theta_{4|0} - \theta_{1|1}\theta_{2|1}\theta_{3|1}\theta_{4|1} \\
(1-\theta_{1|0})\theta_{2|0}\theta_{3|0}\theta_{4|0} - (1-\theta_{1|1})\theta_{2|1}\theta_{3|1}\theta_{4|1} \\
\theta_{1|0}(1-\theta_{2|0})\theta_{3|0}\theta_{4|0} - \theta_{1|1}(1-\theta_{2|1})\theta_{3|1}\theta_{4|1} \\
(1-\theta_{1|0})(1-\theta_{2|0})\theta_{3|0}\theta_{4|0} - (1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1}\theta_{4|1} \\
\theta_{1|0}\theta_{2|0}(1-\theta_{3|0})\theta_{4|0} - \theta_{1|1}\theta_{2|1}(1-\theta_{3|1})\theta_{4|1} \\
(1-\theta_{1|0})\theta_{2|0}(1-\theta_{3|0})\theta_{4|0} - (1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1})\theta_{4|1} \\
\theta_{1|0}(1-\theta_{2|0})(1-\theta_{3|0})\theta_{4|0} - \theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1})\theta_{4|1} \\
(1-\theta_{1|0})(1-\theta_{2|0})(1-\theta_{3|0})\theta_{4|0} - (1-\theta_{1|1})(1-\theta_{2|1})(1-\theta_{3|1})\theta_{4|1} \\
\theta_{1|0}\theta_{2|0}\theta_{3|0}(1-\theta_{4|0}) - \theta_{1|1}\theta_{2|1}\theta_{3|1}(1-\theta_{4|1}) \\
(1-\theta_{1|0})\theta_{2|0}\theta_{3|0}(1-\theta_{4|0}) - (1-\theta_{1|1})\theta_{2|1}\theta_{3|1}(1-\theta_{4|1}) \\
\theta_{1|0}(1-\theta_{2|0})\theta_{3|0}(1-\theta_{4|0}) - \theta_{1|1}(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1}) \\
(1-\theta_{1|0})(1-\theta_{2|0})\theta_{3|0}(1-\theta_{4|0}) - (1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1}) \\
\theta_{1|0}\theta_{2|0}(1-\theta_{3|0})(1-\theta_{4|0}) - \theta_{1|1}\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1}) \\
(1-\theta_{1|0})\theta_{2|0}(1-\theta_{3|0})(1-\theta_{4|0}) - (1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1}) \\
\theta_{1|0}(1-\theta_{2|0})(1-\theta_{3|0})(1-\theta_{4|0}) - \theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1})(1-\theta_{4|1})
\end{pmatrix}
$$

$$
\begin{pmatrix}
p\theta_{2|0}\theta_{3|0}\theta_{4|0} \\
-p\theta_{2|0}\theta_{3|0}\theta_{4|0} \\
p(1-\theta_{2|0})\theta_{3|0}\theta_{4|0} \\
-p(1-\theta_{2|0})\theta_{3|0}\theta_{4|0} \\
p\theta_{2|0}(1-\theta_{3|0})\theta_{4|0} \\
-p\theta_{2|0}(1-\theta_{3|0})\theta_{4|0} \\
p(1-\theta_{2|0})(1-\theta_{3|0})\theta_{4|0} \\
-p(1-\theta_{2|0})(1-\theta_{3|0})\theta_{4|0} \\
p\theta_{2|0}\theta_{3|0}(1-\theta_{4|0}) \\
-p\theta_{2|0}\theta_{3|0}(1-\theta_{4|0}) \\
p(1-\theta_{2|0})\theta_{3|0}(1-\theta_{4|0}) \\
-p(1-\theta_{2|0})\theta_{3|0}(1-\theta_{4|0}) \\
p\theta_{2|0}(1-\theta_{3|0})(1-\theta_{4|0}) \\
-p\theta_{2|0}(1-\theta_{3|0})(1-\theta_{4|0}) \\
p(1-\theta_{2|0})(1-\theta_{3|0})(1-\theta_{4|0})
\end{pmatrix}
\begin{pmatrix}
p\theta_{1|0}\theta_{3|0}\theta_{4|0} \\
p(1-\theta_{1|0})\theta_{3|0}\theta_{4|0} \\
-p\theta_{1|0}\theta_{3|0}\theta_{4|0} \\
-p(1-\theta_{1|0})\theta_{3|0}\theta_{4|0} \\
p\theta_{1|0}(1-\theta_{3|0})\theta_{4|0} \\
p(1-\theta_{1|0})(1-\theta_{3|0})\theta_{4|0} \\
-p\theta_{1|0}(1-\theta_{3|0})\theta_{4|0} \\
-p(1-\theta_{1|0})(1-\theta_{3|0})\theta_{4|0} \\
p\theta_{1|0}\theta_{3|0}(1-\theta_{4|0}) \\
p(1-\theta_{1|0})\theta_{3|0}(1-\theta_{4|0}) \\
-p\theta_{1|0}\theta_{3|0}(1-\theta_{4|0}) \\
-p(1-\theta_{1|0})\theta_{3|0}(1-\theta_{4|0}) \\
p\theta_{1|0}(1-\theta_{3|0})(1-\theta_{4|0}) \\
p(1-\theta_{1|0})(1-\theta_{3|0})(1-\theta_{4|0}) \\
-p\theta_{1|0}(1-\theta_{3|0})(1-\theta_{4|0})
\end{pmatrix}
\begin{pmatrix}
p\theta_{1|0}\theta_{2|0}\theta_{4|0} \\
p(1-\theta_{1|0})\theta_{2|0}\theta_{4|0} \\
p\theta_{1|0}(1-\theta_{2|0})\theta_{4|0} \\
p(1-\theta_{1|0})(1-\theta_{2|0})\theta_{4|0} \\
-p\theta_{1|0}\theta_{2|0}\theta_{4|0} \\
-p(1-\theta_{1|0})\theta_{2|0}\theta_{4|0} \\
-p\theta_{1|0}(1-\theta_{2|0})\theta_{4|0} \\
-p(1-\theta_{1|0})(1-\theta_{2|0})\theta_{4|0} \\
p\theta_{1|0}\theta_{2|0}(1-\theta_{4|0}) \\
p(1-\theta_{1|0})\theta_{2|0}(1-\theta_{4|0}) \\
p\theta_{1|0}(1-\theta_{2|0})(1-\theta_{4|0}) \\
p(1-\theta_{1|0})(1-\theta_{2|0})(1-\theta_{4|0}) \\
-p\theta_{1|0}\theta_{2|0}(1-\theta_{4|0}) \\
-p(1-\theta_{1|0})\theta_{2|0}(1-\theta_{4|0}) \\
-p\theta_{1|0}(1-\theta_{2|0})(1-\theta_{4|0})
\end{pmatrix}
\begin{pmatrix}
p\theta_{1|0}\theta_{2|0}\theta_{3|0} \\
p(1-\theta_{1|0})\theta_{2|0}\theta_{3|0} \\
p\theta_{1|0}(1-\theta_{2|0})\theta_{3|0} \\
p(1-\theta_{1|0})(1-\theta_{2|0})\theta_{3|0} \\
p\theta_{1|0}\theta_{2|0}(1-\theta_{3|0}) \\
p(1-\theta_{1|0})\theta_{2|0}(1-\theta_{3|0}) \\
p\theta_{1|0}(1-\theta_{2|0})(1-\theta_{3|0}) \\
p(1-\theta_{1|0})(1-\theta_{2|0})(1-\theta_{3|0}) \\
-p\theta_{1|0}\theta_{2|0}\theta_{3|0} \\
-p(1-\theta_{1|0})\theta_{2|0}\theta_{3|0} \\
-p\theta_{1|0}(1-\theta_{2|0})\theta_{3|0} \\
-p(1-\theta_{1|0})(1-\theta_{2|0})\theta_{3|0} \\
-p\theta_{1|0}\theta_{2|0}(1-\theta_{3|0}) \\
-p(1-\theta_{1|0})\theta_{2|0}(1-\theta_{3|0}) \\
-p\theta_{1|0}(1-\theta_{2|0})(1-\theta_{3|0})
\end{pmatrix}
$$

$$
\begin{pmatrix}
(1-p)\theta_{2|1}\theta_{3|1}\theta_{4|1} \\
-(1-p)\theta_{2|1}\theta_{3|1}\theta_{4|1} \\
(1-p)(1-\theta_{2|1})\theta_{3|1}\theta_{4|1} \\
-(1-p)(1-\theta_{2|1})\theta_{3|1}\theta_{4|1} \\
(1-p)\theta_{2|1}(1-\theta_{3|1})\theta_{4|1} \\
-(1-p)\theta_{2|1}(1-\theta_{3|1})\theta_{4|1} \\
(1-p)(1-\theta_{2|1})(1-\theta_{3|1})\theta_{4|1} \\
-(1-p)(1-\theta_{2|1})(1-\theta_{3|1})\theta_{4|1} \\
(1-p)\theta_{2|1}\theta_{3|1}(1-\theta_{4|1}) \\
-(1-p)\theta_{2|1}\theta_{3|1}(1-\theta_{4|1}) \\
(1-p)(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1}) \\
-(1-p)(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1}) \\
(1-p)\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1}) \\
-(1-p)\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1}) \\
(1-p)(1-\theta_{2|1})(1-\theta_{3|1})(1-\theta_{4|1})
\end{pmatrix}
\begin{pmatrix}
(1-p)\theta_{1|1}\theta_{3|1}\theta_{4|1} \\
(1-p)(1-\theta_{1|1})\theta_{3|1}\theta_{4|1} \\
-(1-p)\theta_{1|1}\theta_{3|1}\theta_{4|1} \\
-(1-p)(1-\theta_{1|1})\theta_{3|1}\theta_{4|1} \\
(1-p)\theta_{1|1}(1-\theta_{3|1})\theta_{4|1} \\
(1-p)(1-\theta_{1|1})(1-\theta_{3|1})\theta_{4|1} \\
-(1-p)\theta_{1|1}(1-\theta_{3|1})\theta_{4|1} \\
-(1-p)(1-\theta_{1|1})(1-\theta_{3|1})\theta_{4|1} \\
(1-p)\theta_{1|1}\theta_{3|1}(1-\theta_{4|1}) \\
(1-p)(1-\theta_{1|1})\theta_{3|1}(1-\theta_{4|1}) \\
-(1-p)\theta_{1|1}\theta_{3|1}(1-\theta_{4|1}) \\
-(1-p)(1-\theta_{1|1})\theta_{3|1}(1-\theta_{4|1}) \\
(1-p)\theta_{1|1}(1-\theta_{3|1})(1-\theta_{4|1}) \\
(1-p)(1-\theta_{1|1})(1-\theta_{3|1})(1-\theta_{4|1}) \\
-(1-p)\theta_{1|1}(1-\theta_{3|1})(1-\theta_{4|1})
\end{pmatrix}
\begin{pmatrix}
(1-p)\theta_{1|1}\theta_{2|1}\theta_{4|1} \\
(1-p)(1-\theta_{1|1})\theta_{2|1}\theta_{4|1} \\
(1-p)\theta_{1|1}(1-\theta_{2|1})\theta_{4|1} \\
(1-p)(1-\theta_{1|1})(1-\theta_{2|1})\theta_{4|1} \\
-(1-p)\theta_{1|1}\theta_{2|1}\theta_{4|1} \\
-(1-p)(1-\theta_{1|1})\theta_{2|1}\theta_{4|1} \\
-(1-p)\theta_{1|1}(1-\theta_{2|1})\theta_{4|1} \\
-(1-p)(1-\theta_{1|1})(1-\theta_{2|1})\theta_{4|1} \\
(1-p)\theta_{1|1}\theta_{2|1}(1-\theta_{4|1}) \\
(1-p)(1-\theta_{1|1})\theta_{2|1}(1-\theta_{4|1}) \\
(1-p)\theta_{1|1}(1-\theta_{2|1})(1-\theta_{4|1}) \\
(1-p)(1-\theta_{1|1})(1-\theta_{2|1})(1-\theta_{4|1}) \\
-(1-p)\theta_{1|1}\theta_{2|1}(1-\theta_{4|1}) \\
-(1-p)(1-\theta_{1|1})\theta_{2|1}(1-\theta_{4|1}) \\
-(1-p)\theta_{1|1}(1-\theta_{2|1})(1-\theta_{4|1})
\end{pmatrix}
\begin{pmatrix}
(1-p)\theta_{1|1}\theta_{2|1}\theta_{3|1} \\
(1-p)(1-\theta_{1|1})\theta_{2|1}\theta_{3|1} \\
(1-p)\theta_{1|1}(1-\theta_{2|1})\theta_{3|1} \\
(1-p)(1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1} \\
(1-p)\theta_{1|1}\theta_{2|1}(1-\theta_{3|1}) \\
(1-p)(1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1}) \\
(1-p)\theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1}) \\
(1-p)(1-\theta_{1|1})(1-\theta_{2|1})(1-\theta_{3|1}) \\
-(1-p)\theta_{1|1}\theta_{2|1}\theta_{3|1} \\
-(1-p)(1-\theta_{1|1})\theta_{2|1}\theta_{3|1} \\
-(1-p)\theta_{1|1}(1-\theta_{2|1})\theta_{3|1} \\
-(1-p)(1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1} \\
-(1-p)\theta_{1|1}\theta_{2|1}(1-\theta_{3|1}) \\
-(1-p)(1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1}) \\
-(1-p)\theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1})
\end{pmatrix}
$$

$$\hat{J}_4 = \begin{pmatrix}
\theta_{1|1}\theta_{2|1}\theta_{3|1}\theta_{4|1} - \theta_{1|2}\theta_{2|2}\theta_{3|2}\theta_{4|2} \\
(1-\theta_{1|1})\theta_{2|1}\theta_{3|1}\theta_{4|1} - (1-\theta_{1|2})\theta_{2|2}\theta_{3|2}\theta_{4|2} \\
\theta_{1|1}(1-\theta_{2|1})\theta_{3|1}\theta_{4|1} - \theta_{1|2}(1-\theta_{2|2})\theta_{3|2}\theta_{4|2} \\
(1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1}\theta_{4|1} - (1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2}\theta_{4|2} \\
\theta_{1|1}\theta_{2|1}(1-\theta_{3|1})\theta_{4|1} - \theta_{1|2}\theta_{2|2}(1-\theta_{3|2})\theta_{4|2} \\
(1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1})\theta_{4|1} - (1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2})\theta_{4|2} \\
\theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1})\theta_{4|1} - \theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2})\theta_{4|2} \\
(1-\theta_{1|1})(1-\theta_{2|1})(1-\theta_{3|1})\theta_{4|1} - (1-\theta_{1|2})(1-\theta_{2|2})(1-\theta_{3|2})\theta_{4|2} \\
\theta_{1|1}\theta_{2|1}\theta_{3|1}(1-\theta_{4|1}) - \theta_{1|2}\theta_{2|2}\theta_{3|2}(1-\theta_{4|2}) \\
(1-\theta_{1|1})\theta_{2|1}\theta_{3|1}(1-\theta_{4|1}) - (1-\theta_{1|2})\theta_{2|2}\theta_{3|2}(1-\theta_{4|2}) \\
\theta_{1|1}(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1}) - \theta_{1|2}(1-\theta_{2|2})\theta_{3|2}(1-\theta_{4|2}) \\
(1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1}) - (1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2}(1-\theta_{4|2}) \\
\theta_{1|1}\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1}) - \theta_{1|2}\theta_{2|2}(1-\theta_{3|2})(1-\theta_{4|2}) \\
(1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1}) - (1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2})(1-\theta_{4|2}) \\
\theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1})(1-\theta_{4|1}) - \theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2})(1-\theta_{4|2})
\end{pmatrix}$$

$$\begin{pmatrix}
\theta_{2|1}\theta_{3|1} \\
-\theta_{2|1}\theta_{3|1} \\
(1-\theta_{2|1})\theta_{3|1} \\
-(1-\theta_{2|1})\theta_{3|1} \\
\theta_{2|1}(1-\theta_{3|1}) \\
-\theta_{2|1}(1-\theta_{3|1}) \\
(1-\theta_{2|1})(1-\theta_{3|1}) \\
-(1-\theta_{2|1})(1-\theta_{3|1}) \\
\theta_{2|1}\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-\theta_{2|1}\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1} \\
(1-\theta_{2|1})(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1}
\end{pmatrix}
\begin{pmatrix}
\theta_{1|1}\theta_{3|1} \\
(1-\theta_{1|1})\theta_{3|1} \\
-\theta_{1|1}\theta_{3|1} \\
-(1-\theta_{1|1})\theta_{3|1} \\
\theta_{1|1}(1-\theta_{3|1}) \\
(1-\theta_{1|1})(1-\theta_{3|1}) \\
-\theta_{1|1}(1-\theta_{3|1}) \\
-(1-\theta_{1|1})(1-\theta_{3|1}) \\
\theta_{1|1}\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-\theta_{1|1}\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
(1-\theta_{1|1})\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-\theta_{1|1}\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-(1-\theta_{1|1})\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
\theta_{1|1}(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1} \\
(1-\theta_{1|1})(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-\theta_{1|1}(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1}
\end{pmatrix}
\begin{pmatrix}
\theta_{1|1}\theta_{2|1} \\
(1-\theta_{1|1})\theta_{2|1} \\
\theta_{1|1}(1-\theta_{2|1}) \\
(1-\theta_{1|1})(1-\theta_{2|1}) \\
-\theta_{1|1}\theta_{2|1} \\
-(1-\theta_{1|1})\theta_{2|1} \\
-\theta_{1|1}(1-\theta_{2|1}) \\
-(1-\theta_{1|1})(1-\theta_{2|1}) \\
\theta_{1|1}\theta_{2|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-\theta_{1|1}\theta_{2|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
(1-\theta_{1|1})\theta_{2|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
\theta_{1|1}(1-\theta_{2|1})(1-\theta_{4|1})\theta_{4|1}^{-1} \\
(1-\theta_{1|1})(1-\theta_{2|1})(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-\theta_{1|1}\theta_{2|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-(1-\theta_{1|1})\theta_{2|1}(1-\theta_{4|1})\theta_{4|1}^{-1} \\
-\theta_{1|1}(1-\theta_{2|1})(1-\theta_{4|1})\theta_{4|1}^{-1}
\end{pmatrix}
\begin{pmatrix}
\theta_{1|1}\theta_{2|1}\theta_{3|1} - \theta_{1|2}\theta_{2|2}\theta_{3|2} \\
(1-\theta_{1|1})\theta_{2|1}\theta_{3|1} - (1-\theta_{1|2})\theta_{2|2}\theta_{3|2} \\
\theta_{1|1}(1-\theta_{2|1})\theta_{3|1} - \theta_{1|2}(1-\theta_{2|2})\theta_{3|2} \\
(1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1} - (1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2} \\
\theta_{1|1}\theta_{2|1}(1-\theta_{3|1}) - \theta_{1|2}\theta_{2|2}(1-\theta_{3|2}) \\
(1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1}) - (1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2}) \\
\theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1}) - \theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2}) \\
(1-\theta_{1|1})(1-\theta_{2|1})(1-\theta_{3|1}) - (1-\theta_{1|2})(1-\theta_{2|2})(1-\theta_{3|2}) \\
-\theta_{1|1}\theta_{2|1}\theta_{3|1} + \theta_{1|2}\theta_{2|2}\theta_{3|2} \\
-(1-\theta_{1|1})\theta_{2|1}\theta_{3|1} + (1-\theta_{1|2})\theta_{2|2}\theta_{3|2} \\
-\theta_{1|1}(1-\theta_{2|1})\theta_{3|1} + \theta_{1|2}(1-\theta_{2|2})\theta_{3|2} \\
-(1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1} + (1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2} \\
-\theta_{1|1}\theta_{2|1}(1-\theta_{3|1}) + \theta_{1|2}\theta_{2|2}(1-\theta_{3|2}) \\
-(1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1}) + (1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2}) \\
-\theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1}) + \theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2})
\end{pmatrix}$$

$$\begin{pmatrix}
\theta_{2|2}\theta_{3|2} \\
-\theta_{2|2}\theta_{3|2} \\
(1-\theta_{2|2})\theta_{3|2} \\
-(1-\theta_{2|2})\theta_{3|2} \\
\theta_{2|2}(1-\theta_{3|2}) \\
-\theta_{2|2}(1-\theta_{3|2}) \\
(1-\theta_{2|2})(1-\theta_{3|2}) \\
-(1-\theta_{2|2})(1-\theta_{3|2}) \\
\theta_{2|2}\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-\theta_{2|2}\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
(1-\theta_{2|2})\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-(1-\theta_{2|2})\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
\theta_{2|2}(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-\theta_{2|2}(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1} \\
(1-\theta_{2|2})(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1}
\end{pmatrix}
\begin{pmatrix}
\theta_{1|2}\theta_{3|2} \\
(1-\theta_{1|2})\theta_{3|2} \\
-\theta_{1|2}\theta_{3|2} \\
-(1-\theta_{1|2})\theta_{3|2} \\
\theta_{1|2}(1-\theta_{3|2}) \\
(1-\theta_{1|2})(1-\theta_{3|2}) \\
-\theta_{1|2}(1-\theta_{3|2}) \\
-(1-\theta_{1|2})(1-\theta_{3|2}) \\
\theta_{1|2}\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-\theta_{1|2}\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
(1-\theta_{1|2})\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-\theta_{1|2}\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-(1-\theta_{1|2})\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
\theta_{1|2}(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1} \\
(1-\theta_{1|2})(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-\theta_{1|2}(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1}
\end{pmatrix}
\begin{pmatrix}
\theta_{1|2}\theta_{2|2} \\
(1-\theta_{1|2})\theta_{2|2} \\
\theta_{1|2}(1-\theta_{2|2}) \\
(1-\theta_{1|2})(1-\theta_{2|2}) \\
-\theta_{1|2}\theta_{2|2} \\
-(1-\theta_{1|2})\theta_{2|2} \\
-\theta_{1|2}(1-\theta_{2|2}) \\
-(1-\theta_{1|2})(1-\theta_{2|2}) \\
\theta_{1|2}\theta_{2|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-\theta_{1|2}\theta_{2|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
(1-\theta_{1|2})\theta_{2|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
\theta_{1|2}(1-\theta_{2|2})(1-\theta_{4|2})\theta_{4|2}^{-1} \\
(1-\theta_{1|2})(1-\theta_{2|2})(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-\theta_{1|2}\theta_{2|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-(1-\theta_{1|2})\theta_{2|2}(1-\theta_{4|2})\theta_{4|2}^{-1} \\
-\theta_{1|2}(1-\theta_{2|2})(1-\theta_{4|2})\theta_{4|2}^{-1}
\end{pmatrix}
\begin{pmatrix}
\theta_{1|2}\theta_{2|2}\theta_{3|2} \\
(1-\theta_{1|2})\theta_{2|2}\theta_{3|2} \\
\theta_{1|2}(1-\theta_{2|2})\theta_{3|2} \\
(1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2} \\
\theta_{1|2}\theta_{2|2}(1-\theta_{3|2}) \\
(1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2}) \\
\theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2}) \\
(1-\theta_{1|2})(1-\theta_{2|2})(1-\theta_{3|2}) \\
-\theta_{1|2}\theta_{2|2}\theta_{3|2} \\
-(1-\theta_{1|2})\theta_{2|2}\theta_{3|2} \\
-\theta_{1|2}(1-\theta_{2|2})\theta_{3|2} \\
-(1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2} \\
-\theta_{1|2}\theta_{2|2}(1-\theta_{3|2}) \\
-(1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2}) \\
-\theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2})
\end{pmatrix}$$

$$J_3 = \begin{pmatrix}
\theta_{1|1}\theta_{2|1}\theta_{3|1} - \theta_{1|2}\theta_{2|2}\theta_{3|2} & p\theta_{2|1}\theta_{3|1} & p\theta_{1|1}\theta_{3|1} & p\theta_{1|1}\theta_{2|1} \\
(1-\theta_{1|1})\theta_{2|1}\theta_{3|1} - (1-\theta_{1|2})\theta_{2|2}\theta_{3|2} & -p\theta_{2|1}\theta_{3|1} & p(1-\theta_{1|1})\theta_{3|1} & p(1-\theta_{1|1})\theta_{2|1} \\
\theta_{1|1}(1-\theta_{2|1})\theta_{3|1} - \theta_{1|2}(1-\theta_{2|2})\theta_{3|2} & p(1-\theta_{2|1})\theta_{3|1} & -p\theta_{1|1}\theta_{3|1} & p\theta_{1|1}(1-\theta_{2|1}) \\
(1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1} - (1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2} & -p(1-\theta_{2|1})\theta_{3|1} & -p(1-\theta_{1|1})\theta_{3|1} & p(1-\theta_{1|1})(1-\theta_{2|1}) \\
\theta_{1|1}\theta_{2|1}(1-\theta_{3|1}) - \theta_{1|2}\theta_{2|2}(1-\theta_{3|2}) & p\theta_{2|1}(1-\theta_{3|1}) & p\theta_{1|1}(1-\theta_{3|1}) & -p\theta_{1|1}\theta_{2|1} \\
(1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1}) - (1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2}) & -p\theta_{2|1}(1-\theta_{3|1}) & p(1-\theta_{1|1})(1-\theta_{3|1}) & -p(1-\theta_{1|1})\theta_{2|1} \\
\theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1}) - \theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2}) & p(1-\theta_{2|1})(1-\theta_{3|1}) & -p\theta_{1|1}(1-\theta_{3|1}) & -p\theta_{1|1}(1-\theta_{2|1})
\end{pmatrix}$$

$$\begin{pmatrix}
(1-p)\theta_{2|2}\theta_{3|2} & (1-p)\theta_{1|2}\theta_{3|2} & (1-p)\theta_{1|2}\theta_{2|2} \\
-(1-p)\theta_{2|2}\theta_{3|2} & (1-p)(1-\theta_{1|2})\theta_{3|2} & (1-p)(1-\theta_{1|2})\theta_{2|2} \\
(1-p)(1-\theta_{2|2})\theta_{3|2} & -(1-p)\theta_{1|2}\theta_{3|2} & (1-p)\theta_{1|2}(1-\theta_{2|2}) \\
-(1-p)(1-\theta_{2|2})\theta_{3|2} & -(1-p)(1-\theta_{1|2})\theta_{3|2} & (1-p)(1-\theta_{1|2})(1-\theta_{2|2}) \\
(1-p)\theta_{2|2}(1-\theta_{3|2}) & (1-p)\theta_{1|2}(1-\theta_{3|2}) & -(1-p)\theta_{1|2}\theta_{2|2} \\
-(1-p)\theta_{2|2}(1-\theta_{3|2}) & (1-p)(1-\theta_{1|2})(1-\theta_{3|2}) & -(1-p)(1-\theta_{1|2})\theta_{2|2} \\
(1-p)(1-\theta_{2|2})(1-\theta_{3|2}) & -(1-p)\theta_{1|2}(1-\theta_{3|2}) & -(1-p)\theta_{1|2}(1-\theta_{2|2})
\end{pmatrix}$$

$$\hat{J}_3 = \begin{pmatrix}
\theta_{2|1}\theta_{3|1} & \theta_{1|1}\theta_{3|1} & \theta_{1|1}\theta_{2|1} & \theta_{1|1}\theta_{2|1}\theta_{3|1} - \theta_{1|2}\theta_{2|2}\theta_{3|2} \\
-\theta_{2|1}\theta_{3|1} & (1-\theta_{1|1})\theta_{3|1} & (1-\theta_{1|1})\theta_{2|1} & (1-\theta_{1|1})\theta_{2|1}\theta_{3|1} - (1-\theta_{1|2})\theta_{2|2}\theta_{3|2} \\
(1-\theta_{2|1})\theta_{3|1} & -\theta_{1|1}\theta_{3|1} & \theta_{1|1}(1-\theta_{2|1}) & \theta_{1|1}(1-\theta_{2|1})\theta_{3|1} - \theta_{1|2}(1-\theta_{2|2})\theta_{3|2} \\
-(1-\theta_{2|1})\theta_{3|1} & -(1-\theta_{1|1})\theta_{3|1} & (1-\theta_{1|1})(1-\theta_{2|1}) & (1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1} - (1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2} \\
\theta_{2|1}(1-\theta_{3|1}) & \theta_{1|1}(1-\theta_{3|1}) & -\theta_{1|1}\theta_{2|1} & \theta_{1|1}\theta_{2|1}(1-\theta_{3|1}) - \theta_{1|2}\theta_{2|2}(1-\theta_{3|2}) \\
-\theta_{2|1}(1-\theta_{3|1}) & (1-\theta_{1|1})(1-\theta_{3|1}) & -(1-\theta_{1|1})\theta_{2|1} & (1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1}) - (1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2}) \\
(1-\theta_{2|1})(1-\theta_{3|1}) & -\theta_{1|1}(1-\theta_{3|1}) & -\theta_{1|1}(1-\theta_{2|1}) & \theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1}) - \theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2})
\end{pmatrix}$$

$$\begin{pmatrix}
\theta_{2|2}\theta_{3|2} & \theta_{1|2}\theta_{3|2} & \theta_{1|2}\theta_{2|2} \\
-\theta_{2|2}\theta_{3|2} & (1-\theta_{1|2})\theta_{3|2} & (1-\theta_{1|2})\theta_{2|2} \\
(1-\theta_{2|2})\theta_{3|2} & -\theta_{1|2}\theta_{3|2} & \theta_{1|2}(1-\theta_{2|2}) \\
-(1-\theta_{2|2})\theta_{3|2} & -(1-\theta_{1|2})\theta_{3|2} & (1-\theta_{1|2})(1-\theta_{2|2}) \\
\theta_{2|2}(1-\theta_{3|2}) & \theta_{1|2}(1-\theta_{3|2}) & -\theta_{1|2}\theta_{2|2} \\
-\theta_{2|2}(1-\theta_{3|2}) & (1-\theta_{1|2})(1-\theta_{3|2}) & -(1-\theta_{1|2})\theta_{2|2} \\
(1-\theta_{2|2})(1-\theta_{3|2}) & -\theta_{1|2}(1-\theta_{3|2}) & -\theta_{1|2}(1-\theta_{2|2})
\end{pmatrix}$$

$$w_3 = \begin{pmatrix} \theta_{1|1}\theta_{2|1}\theta_{3|1}\theta_{4|1} - \theta_{1|2}\theta_{2|2}\theta_{3|2}\theta_{4|2} \\ (1-\theta_{1|1})\theta_{2|1}\theta_{3|1}\theta_{4|1} - (1-\theta_{1|2})\theta_{2|2}\theta_{3|2}\theta_{4|2} \\ \theta_{1|1}(1-\theta_{2|1})\theta_{3|1}\theta_{4|1} - \theta_{1|2}(1-\theta_{2|2})\theta_{3|2}\theta_{4|2} \\ (1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1}\theta_{4|1} - (1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2}\theta_{4|2} \\ \theta_{1|1}\theta_{2|1}(1-\theta_{3|1})\theta_{4|1} - \theta_{1|2}\theta_{2|2}(1-\theta_{3|2})\theta_{4|2} \\ (1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1})\theta_{4|1} - (1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2})\theta_{4|2} \\ \theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1})\theta_{4|1} - \theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2})\theta_{4|2} \\ (1-\theta_{1|1})(1-\theta_{2|1})(1-\theta_{3|1})\theta_{4|1} - (1-\theta_{1|2})(1-\theta_{2|2})(1-\theta_{3|2})\theta_{4|2} \\ \theta_{1|1}\theta_{2|1}\theta_{3|1}(1-\theta_{4|1}) - \theta_{1|2}\theta_{2|2}\theta_{3|2}(1-\theta_{4|2}) \\ (1-\theta_{1|1})\theta_{2|1}\theta_{3|1}(1-\theta_{4|1}) - (1-\theta_{1|2})\theta_{2|2}\theta_{3|2}(1-\theta_{4|2}) \\ \theta_{1|1}(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1}) - \theta_{1|2}(1-\theta_{2|2})\theta_{3|2}(1-\theta_{4|2}) \\ (1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1}) - (1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2}(1-\theta_{4|2}) \\ \theta_{1|1}\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1}) - \theta_{1|2}\theta_{2|2}(1-\theta_{3|2})(1-\theta_{4|2}) \\ (1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1}) - (1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2})(1-\theta_{4|2}) \\ \theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1})(1-\theta_{4|1}) - \theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2})(1-\theta_{4|2}) \end{pmatrix} \qquad u_3 = \begin{pmatrix} \theta_{1|2}\theta_{2|2}\theta_{3|2} \\ (1-\theta_{1|2})\theta_{2|2}\theta_{3|2} \\ \theta_{1|2}(1-\theta_{2|2})\theta_{3|2} \\ (1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2} \\ \theta_{1|2}\theta_{2|2}(1-\theta_{3|2}) \\ (1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2}) \\ \theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2}) \end{pmatrix}.$$

$$j_3 = \begin{pmatrix} -(1-\theta_{2|1})(1-\theta_{3|1}) \\ -(1-\theta_{1|1})(1-\theta_{3|1}) \\ -(1-\theta_{1|1})(1-\theta_{2|1}) \\ (1-\theta_{1|1})(1-\theta_{2|1})(1-\theta_{3|1}) - (1-\theta_{2|2})(1-\theta_{2|2})(1-\theta_{3|2}) \\ -(1-\theta_{2|2})(1-\theta_{3|2}) \\ -(1-\theta_{1|2})(1-\theta_{3|2}) \\ -(1-\theta_{1|2})(1-\theta_{2|2}) \end{pmatrix}^T \qquad v_3 = (1-\theta_{1|2})(1-\theta_{2|2})(1-\theta_{3|2})$$

$$K_3 = \begin{pmatrix} \theta_{2|1}\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & \theta_{1|1}\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & \theta_{1|1}\theta_{2|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & \theta_{1|1}\theta_{2|1}\theta_{3|1} - \theta_{1|2}\theta_{2|2}\theta_{3|2} \\ -\theta_{2|1}\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & (1-\theta_{1|1})\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & (1-\theta_{1|1})\theta_{2|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & (1-\theta_{1|1})\theta_{2|1}\theta_{3|1} - (1-\theta_{1|2})\theta_{2|2}\theta_{3|2} \\ (1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & -\theta_{1|1}\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & \theta_{1|1}(1-\theta_{2|1})(1-\theta_{4|1})\theta_{4|1}^{-1} & \theta_{1|1}(1-\theta_{2|1})\theta_{3|1} - \theta_{1|2}(1-\theta_{2|2})\theta_{3|2} \\ -(1-\theta_{2|1})\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & -(1-\theta_{1|1})\theta_{3|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & (1-\theta_{1|1})(1-\theta_{2|1})(1-\theta_{4|1})\theta_{4|1}^{-1} & (1-\theta_{1|1})(1-\theta_{2|1})\theta_{3|1} - (1-\theta_{1|2})(1-\theta_{2|2})\theta_{3|2} \\ \theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1} & \theta_{1|1}(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1} & -\theta_{1|1}\theta_{2|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & \theta_{1|1}\theta_{2|1}(1-\theta_{3|1}) - \theta_{1|2}\theta_{2|2}(1-\theta_{3|2}) \\ -\theta_{2|1}(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1} & (1-\theta_{1|1})(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1} & -(1-\theta_{1|1})\theta_{2|1}(1-\theta_{4|1})\theta_{4|1}^{-1} & (1-\theta_{1|1})\theta_{2|1}(1-\theta_{3|1}) - (1-\theta_{1|2})\theta_{2|2}(1-\theta_{3|2}) \\ (1-\theta_{2|1})(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1} & -\theta_{1|1}(1-\theta_{3|1})(1-\theta_{4|1})\theta_{4|1}^{-1} & -\theta_{1|1}(1-\theta_{2|1})(1-\theta_{4|1})\theta_{4|1}^{-1} & \theta_{1|1}(1-\theta_{2|1})(1-\theta_{3|1}) - \theta_{1|2}(1-\theta_{2|2})(1-\theta_{3|2}) \\ \theta_{2|2}\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & \theta_{1|2}\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & \theta_{1|2}\theta_{2|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & \\ -\theta_{2|2}\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & (1-\theta_{1|2})\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & (1-\theta_{1|2})\theta_{2|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & \\ (1-\theta_{2|2})\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & -\theta_{1|2}\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & \theta_{1|2}(1-\theta_{2|2})(1-\theta_{4|2})\theta_{4|2}^{-1} & \\ -(1-\theta_{2|2})\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & -(1-\theta_{1|2})\theta_{3|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & (1-\theta_{1|2})(1-\theta_{2|2})(1-\theta_{4|2})\theta_{4|2}^{-1} & \\ \theta_{2|2}(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1} & \theta_{1|2}(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1} & -\theta_{1|2}\theta_{2|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & \\ -\theta_{2|2}(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1} & (1-\theta_{1|2})(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1} & -(1-\theta_{1|2})\theta_{2|2}(1-\theta_{4|2})\theta_{4|2}^{-1} & \\ (1-\theta_{2|2})(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1} & -\theta_{1|2}(1-\theta_{3|2})(1-\theta_{4|2})\theta_{4|2}^{-1} & -\theta_{1|2}(1-\theta_{2|2})(1-\theta_{4|2})\theta_{4|2}^{-1} & \end{pmatrix}$$

Theorem 3.7 *Let S_3 be the naive Bayes network with one binary hidden root node and $n = 3$ binary observed nodes. Then*

$$r = 2n + 1 = 7,$$

where r is the regular rank of the Jacobian matrix between the parameters of the network and the parameters of the feature variables.

Proof of Theorem 3.7

The theorem is easily proved by simple algebraic row reduction of the Jacobian matrix.

Combining Theorems 3.4 and 3.7 gives the inductive proof for Geiger et al's conjecture.

3.3 Discussion

We have shown that for a naive Bayesian network with a binary root node and n binary observable nodes that the model is identifiable for $n > 2$. Clearly this is not the full story; we have only considered what is a very small subset of all naive Bayesian networks. We make the following conjectures, the first concerning binary observables and the second concerning more general observables.

Conjecture 3.8 *Let S be a naive Bayesian network with one hidden root node having r_z possible states and n binary observable non-root nodes, where n satisfies the condition $2^n \geq r_z(n+1)$. Then*

$$r = r_z(n+1),$$

where r is the regular rank of the Jacobian matrix between the parameters of the network and the parameters of the feature variables.

Conjecture 3.9 *Let S be a naive Bayesian network with one hidden root node having r_z possible states and n observable non-root nodes, having r_i possible states $i = 1, \ldots, n$, where n satisfies the condition $\prod_{i=1}^{n} r_i \geq \sum_{i=1}^{n} r_z (r_i - 1) + r_z$ Then*

$$r = \sum_{i=1}^{n} r_z (r_i - 1) + r_z - 1,$$

where r is the regular rank of the Jacobian matrix between the parameters of the network and the parameters of the feature variables.

Table 3.1 shows the first conjecture to be true at least for the models looked at, and using the same numerical techniques we have been unable to provide a counter example to the second conjecture. However, we have not been able to prove these conjectures in general; the 'similar matrices' approach we have used to prove theorem 3.4 does not seem to extend to these more complicated problems.

Chapter 4

Model Selection

4.1 Introduction

As well as using data to learn about the parameters of our model we may also be interested in using our data to help us select our model. In the discussion that follows we will assume that we are interested in choosing from a set of candidate models $M = \{m_1, \ldots, m_\mu\}$. The Bayesian approach to this model selection is to use Bayes' rule to calculate the posterior distributions for each of the candidate models,

$$P(m \mid D) = \frac{P(m)\, P(D \mid m)}{\sum_{m'} P(m')\, P(D \mid m')},$$

where

$$P(D \mid m) = \int P(D \mid \theta_m, m)\, P(\theta_m \mid m)\, d\theta_m$$

is the likelihood averaged over the parameters with respect to their priors and is known as the marginal likelihood. The model then chosen as 'best' is the one which maximises this quantity. In practice the slightly simpler quantity, the

relative posterior probability of a model structure

$$P(D, m) = P(m)\, P(D \mid m)$$

is used to select the best model, or if our prior distribution across the set of candidate models is uniform we can simply look at the quantity $P(D \mid m)$. An alternative, but equivalent criterion for model selection is the *Bayes Factor* defined as

$$\frac{P(m \mid D)}{P(m_0 \mid D)} = \frac{P(m)\, P(D \mid m)}{P(m_0)\, P(D \mid m_0)}.$$

If our dataset is complete, the sets, $\theta_{ij} = \{\theta_{ij1}, \ldots, \theta_{ijr_i}\}$, of model parameters are independent and the parameter priors are Dirichlet with $\theta_{ij} \sim Di(\alpha_{ij1}, \alpha_{ij2}, \ldots, \alpha_{ijr_i})$, then Cooper and Herskovits [17] show that the computation of $P(D \mid m)$ is straightforward, and for a general Bayesian network with n nodes is given by

$$P(D \mid m) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}, \tag{4.1}$$

where

$$\begin{aligned} N_{ijk} &= \sum_{l=1}^{N} I(x_{li} = k, \boldsymbol{pa}_{li} = j), \\ I(x_{il} = k, \boldsymbol{pa}_{il} = j) &= \begin{cases} 1 \text{ if } x_{l1} = k \text{ and } \boldsymbol{pa}_{li} = j \\ 0 \text{ otherwise} \end{cases} \\ N_{ij} &= \sum_{k=1}^{r_i} N_{ijk}, \end{aligned}$$

$$\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}.$$

However, more generally we will be working with incomplete datasets, in which case calculation of this quantity becomes intractable. In this case we must resort to the use of one of a number of approximations to this quantity which have been suggested. One such approximation that will be considered here is a Monte-Carlo technique known as the *Candidate method.* Monte-Carlo techniques are known normally to converge to accurate results, but can be computationally very expensive, which can lead us to use another class of approximations based on the large sample properties of probability distributions. In this chapter we will consider a number of these approximations along with two other model selection criteria proposed by Akaike [1] and Biernacki *et al.* [3].

4.2 The Candidate Method

This MCMC approach suggested by Chib [14] takes advantage of the identity

$$P(D \mid m) = \frac{P(D \mid \theta_m^*, m)\, P(\theta_m^* \mid m)}{P(\theta_m^* \mid D, m)} \tag{4.2}$$

to simplify calculation of the quantity of interest. The idea is that in this identity, true for any value of θ_m^*, we can calculate the numerator on the right-hand side exactly and approximate the denominator using a simple Gibbs sampler. Gibbs samplers came to prominence in the 1984 paper by Geman and Geman [25], and allow us to approximate the expectation of a function $f(\boldsymbol{x})$ with respect to $P(\boldsymbol{x})$,

the joint distribution of variables $\boldsymbol{X} = (X_1, \ldots, X_n)$. We first choose an initial state for $\boldsymbol{X}$, which is generally selected 'at random'. We then take each variable in turn and re-assign its value according to its conditional distribution given all of the other variables in $\boldsymbol{X}$. This gives us a new sample $\boldsymbol{x}$, and we calculate the expected value of $f(\boldsymbol{x})$ by keeping track of the simple average of the function $f(\boldsymbol{x})$ over the samples we construct. It is possible to show that after a 'burn-in' phase in which we discard all of our samples, the configurations of $\boldsymbol{x}$ can be assumed to be sampled with probability $P(\boldsymbol{x})$ if our Gibbs sampler is irreducible and satisfies detailed balance, the two necessary conditions for ergodicity. Irreducibility simply means that no matter what our starting configuration is it is possible to visit every possible configuration of $\boldsymbol{X}$, even though in practice we may not run our Gibbs sampler for long enough to sample all possible configurations. Detailed balance is the condition

$$\frac{P(\text{move from configuration A to configuration B})}{P(\text{move from configuration B to configuration A})} = \frac{P(B)}{P(A)},$$

for all possible A and B, which simply ensures that we are sampling from the correct distribution.

In this case, we initialise the Gibbs sampler by first assigning states to the unobserved variables at random, giving us a complete sample D_c. We then choose some variable X_{li}, which is the unobserved variable X_i in observation l for updating. We update X_{li} by selecting its state according to the probability distribution

$$P(x'_{li} \mid D_c \setminus x_{li}, \boldsymbol{m}) = \frac{P(x'_{li}, D_c \setminus x_{li} \mid \boldsymbol{m})}{\sum_{x''_{li}} P(x''_{li}, D_c \setminus x_{li} \mid \boldsymbol{m})},$$

where the required probabilities can be calculated using Equation (4.1) and where $D_c \setminus x_{li}$ denotes the completed dataset D_c with the observation x_{li} removed. This is done for all unobserved variables in D_c to produce a new completed dataset D'_c. After a suitable burn-in period we begin to collect the values of $P(\theta^*_m \mid D'_c, m)$ for our samples from the posterior, and take the simple average of this quantity as our estimate of $P(\theta^*_m \mid D, m)$. While the identity given in equation 4.2 is true for any value of θ^*_m it is also true that an unwise choice of θ^*_m will lead to very poor convergence of the Gibbs sampler. Chib, and also Raftery [57], recommend using the MAP or ML estimates of θ_m for θ^*_m. Their argument for this choice is that selecting a value for θ^*_m corresponding to a high density point should improve the accuracy of the estimate. Chickering *et al.* [16] claim that this choice of θ^*_m leads to underestimation of the quantity $P(\theta^*_m \mid D, m)$. They suggest that the reason for this error is that there can exist configurations D_c such that both $p(\theta^*_m \mid D_c, m)$ is large and the probability of visiting this particular configuration is small. They suggest an alternative method for selecting θ^*_m, claiming that it provides accurate low-noise estimates of $P(D \mid m)$. For a fixed number of samples after the burn-in phase they keep track of the completed configurations and set θ^*_m to be the configuration that maximises $P(\theta_m \mid D^*_c, m)$, where D^*_c is the completed dataset observed to have occurred most often; or, in the case of ties, the one for which $P(D \mid m)$ is greatest.

4.3 The Laplace Approximation

The idea of the Laplace approximation is that as our sample size N increases it becomes reasonable to approximate $P(\theta_m \mid D, m) \propto P(D \mid \theta_m, m)\, P(\theta_m \mid m)$

with a multivariate Gaussian distribution. In particular, we define

$$g(\boldsymbol{\theta}_m) \equiv \log\left(P(D \mid \boldsymbol{\theta}_m, m)\, P(\boldsymbol{\theta}_m \mid m)\right) \tag{4.3}$$

and $\tilde{\boldsymbol{\theta}}_m$ to be the configuration of $\boldsymbol{\theta}_m$ that maximises $g(\boldsymbol{\theta}_m)$. It can be seen that $\tilde{\boldsymbol{\theta}}_m$ also maximises $P(\boldsymbol{\theta}_m \mid D, m)$ and is therefore the MAP configuration of $\boldsymbol{\theta}_m$.

A second order Taylor polynomial can be used to expand $g(\boldsymbol{\theta}_m)$ about the MAP parameter estimates to give

$$g(\boldsymbol{\theta}_m) \approx g\left(\tilde{\boldsymbol{\theta}}_m\right) - \frac{1}{2}\left(\boldsymbol{\theta}_m - \tilde{\boldsymbol{\theta}}_m\right)^T A \left(\boldsymbol{\theta}_m - \tilde{\boldsymbol{\theta}}_m\right),$$

where A is the negative Hessian of $g(\boldsymbol{\theta}_m)$ evaluated at $\tilde{\boldsymbol{\theta}}_m$. Exponentiating $g(\boldsymbol{\theta}_m)$ and using Equation 4.3 gives us

$$\begin{aligned} &P(D \mid \boldsymbol{\theta}_m, m)\, P(\boldsymbol{\theta}_m \mid m) \\ &\quad \approx P\left(D \mid \tilde{\boldsymbol{\theta}}_m, m\right) P\left(\tilde{\boldsymbol{\theta}}_m \mid m\right) \exp\left\{-\tfrac{1}{2}\left(\boldsymbol{\theta}_m - \tilde{\boldsymbol{\theta}}_m\right)^T A \left(\boldsymbol{\theta}_m - \tilde{\boldsymbol{\theta}}_m\right)\right\}. \end{aligned} \tag{4.4}$$

This gives us a Gaussian approximation for $P(\boldsymbol{\theta}_m \mid D, m) \propto \exp\{g(\boldsymbol{\theta}_m)\}$. We then integrate both sides of Equation 4.4, and take the logarithm to give the final approximation

$$\log P(D \mid m) \approx \log P\left(D \mid \tilde{\boldsymbol{\theta}}_m, m\right) + \log P\left(\tilde{\boldsymbol{\theta}}_m \mid m\right) + \frac{d}{2}\log(2\pi) - \frac{1}{2}\log \mid A \mid, \tag{4.5}$$

where d is the number of parameters in m known as the *dimension* of the model.

4.3.1 Choice of Basis for the Laplace Approximation

The choice of basis, that is to say the parameterisation, in which the Laplace approximation is made is an issue that seems to have been largely ignored until the 1998 paper of MacKay [43]. He demonstrated that for models in which the parameters are probabilities, such as Bayesian belief networks, the most obvious basis is not necessarily the best. He argues that the Laplace approximation is better computed in the 'softmax' representation. That is to say we re-parameterise our model in terms of the parameter space Φ_m, where $\phi_{ijk} = \log\left(\frac{\theta_{ijk}}{\theta_{ij1}}\right)$. Figure 4.1, after the figure given in MacKay's 1998 paper, illustrates the intuition for his claim. The aim of the Laplace approximation is to estimate the integral of a function $\int d^k w f(w)$ by fitting a Gaussian at the maximum $\hat{w}$ of $f(w)$ and computing the volume under that Gaussian. Clearly the better the approximation to the function of interest given by the Gaussian, the better will be the final estimate. In the natural basis the Dirichlet distribution of the parameters is very poorly estimated by a Gaussian distribution; Figure 4.1 clearly shows that the approximation places far too much weight outside the range $[0, 1]$. Working in the softmax basis reduces the problem; Figure 4.1 demonstrates that in this basis the Gaussian approximation, while light-tailed, is quite a close match for the true density.

MacKay also shows that if we have a Dirichlet prior on the parameters θ_{ij} in our original parameter space, that is to say

$$P(\theta_{ij} \mid m) = \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{ijk}\right)}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \left(1 - \sum_{k=2}^{r_i} \theta_{ijk}\right)^{\alpha_{ij1}-1} \theta_{ij2}^{\alpha_{ij2}-1} \ldots \theta_{ijr_i}^{\alpha_{ijr_i}-1},$$

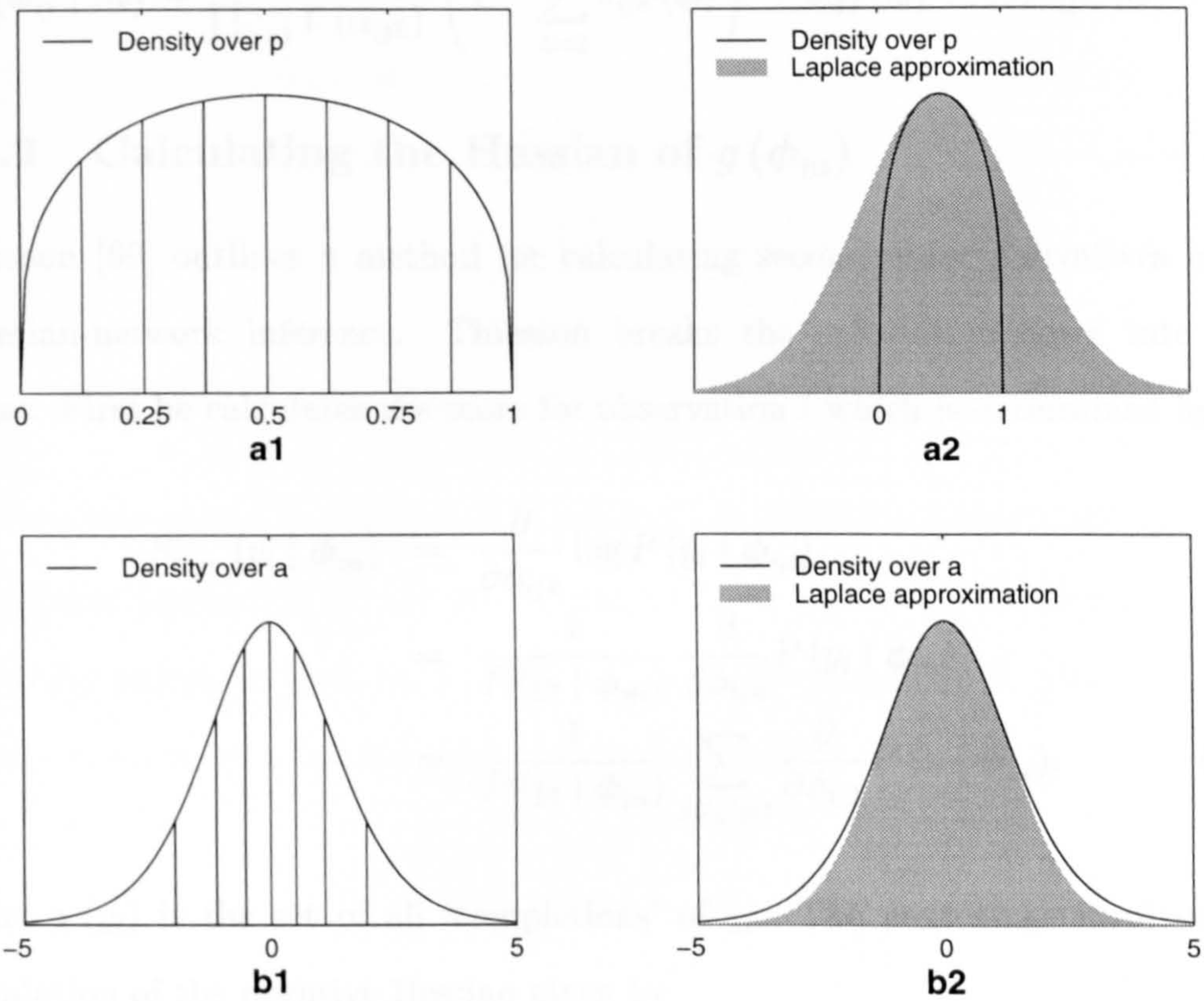


Figure 4.1: After MacKay [43]. This figure shows the Laplace approximation to a Dirichlet approximation and to the same distribution re-parameterised in a softmax basis. Frame a1 shows the density function $B(1.3, 1.3)$ in traditional p-space, so $P(p) \propto p^{0.3}(1-p)^{0.3}$. In b2 the same distribution is shown transformed into a-space, where $a = \log(p/(1-p))$ so $P(a) \propto p(a)^{1.3}(1-p(a))^{1.3}$ with $p(a) = \frac{e^a}{1+e^a}$. Frames a2 and b2 show the Laplace approximations to each of these distributions, clearly the Laplace approximation in the traditional parameter space is a very poor match to the true distribution whereas the Laplace approximation in the transformed parameter space is much closer to the true distribution.

then the corresponding priors on the parameters ϕ_{ij} are given by

$$P(\phi_{ij} \mid m) = \frac{\Gamma(\sum_{k=1}^{r_i} \alpha_{ijk})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \left(1 - \sum_{k=2}^{r_i} \theta_{ijk}(\phi)\right)^{\alpha_{ij1}} \theta_{ij2}(\phi)^{\alpha_{ij2}} \ldots \theta_{ijr_i}(\phi)^{\alpha_{ijr_i}}.$$

4.3.2 Calculating the Hessian of $g(\phi_m)$

Thiesson [69] outlines a method for calculating second order derivatives using Bayesian-network inference. Thiesson breaks the calculation down into two stages. First he calculates the score for observation l which is determined by

$$\begin{aligned} S_{\phi_{ijk}}(y_l \mid \phi_m) &= \frac{\partial}{\partial \phi_{ijk}} \log P(y_l \mid \phi_m) \\ &= \frac{1}{P(y_l \mid \phi_m)} \frac{\partial}{\partial \phi_{ijk}} P(y_l \mid \phi_m) \\ &= \frac{1}{P(y_l \mid \phi_m)} \sum_{x \in \chi(y_l)} \frac{\partial}{\partial \phi_{ijk}} P(x \mid \phi_m), \end{aligned} \tag{4.6}$$

where $\chi(y_l)$ is the set of all 'completions' of y_l. The next stage involves the calculation of the negative Hessian given by

$$\begin{aligned} -H_{\phi_{ijk}\phi_{stu}}(y_l \mid \phi_m) &= -\frac{\partial^2}{\partial \phi_{ijk} \partial \phi_{stu}} \log P(y_l \mid \phi_m) \\ &= -\frac{1}{P(y_l \mid \phi_m)} \frac{\partial^2}{\partial \phi_{ijk} \partial \phi_{stu}} P(y_l \mid \phi_m) \\ &\quad + \frac{1}{P(y_l \mid \phi_m)^2} \frac{\partial}{\partial \phi_{ijk}} P(y_l \mid \phi_m) \frac{\partial}{\partial \phi_{stu}} P(y_l \mid \phi_m) \\ &= S_{\phi_{ijk}}(y_l \mid \phi_m) S_{\phi_{stu}}(y_l \mid \phi_m) \\ &\quad - \frac{1}{P(y_l \mid \phi_m)} \sum_{x \in \chi(y_l)} \frac{\partial^2}{\partial \phi_{ijk} \partial \phi_{stu}} P(x \mid \phi_m). \end{aligned} \tag{4.7}$$

Clearly it is a simple matter of Bayesian network inference to calculate the

quantities $P(y_l \mid \phi_m)$, and Thiesson further shows that the particular structure of a Bayesian network makes the calculation of the quantities $\frac{\partial}{\partial \phi_i} P(x \mid \phi_m)$ and $\frac{\partial^2}{\partial \phi_i \partial \phi_j} P(x \mid \phi_m)$ very straightforward. Hence we can calculate the elements of the Hessian matrix very simply by summing the terms $H_{\phi_{ijk}\phi_{stu}}(y_l \mid \phi_m)$ over our entire dataset. The calculation of this Hessian matrix is explained in greater detail in Appendix C.

4.4 Akaike's Information Criterion (AIC)

AIC was introduced in Akaike's 1974 paper [1] on statistical hypothesis testing in time series analysis. If we are attempting to fit to our data a parametric family of density functions given by $f(x \mid \theta)$ with a vector parameter θ, where the true density is given by $g(x)$, then the average log-likelihood, given by

$$\frac{1}{N} \sum_{i=1}^{N} \log f(x_i \mid \theta), \tag{4.8}$$

will tend to

$$S(g; f(\cdot \mid \theta)) = \int g(x) \log f(x \mid \theta)\, dx,$$

as N increases. The quantity

$$I(g; f(\cdot \mid \theta)) = S(g; g) - S(g; f(\cdot \mid \theta))$$

is the Kullback-Leiber mean information for discrimination between $g(x)$ and $f(x \mid \theta)$ and can be shown to take positive values unless $f(x \mid \theta) = g(x)$ almost everywhere. This implies that a reasonable choice of criterion for model selection

is the minimisation of the quantity $-S(g; f(\cdot \mid \theta))$ as this will bring the Kullback-Leiber mean information as close to 0 as possible. Fortunately the natural estimator for $S(g; f(\cdot \mid \theta))$ is the mean log-likelihood (4.8) and hence knowledge of the true density is not necessary in order to estimate the quantity we are using as our model selection criterion. If we are considering only a single family of distributions then we use Equation (4.8) to produce our ML parameter estimates. However if we are in a position to choose between a number of different models for our data the principle of maximum likelihood does not provide a suitable solution as it will always favour the more complicated model. Akaike recognised this problem and it was this that motivated the development of the AIC. He began by considering the situation where $g(x) = f(x \mid \theta_0)$, that is to say the true density is contained somewhere within our model space. Denoting $I(g; f(\cdot \mid \theta))$ and $S(g; f(\cdot \mid \theta))$ by $I(\theta_0; \theta)$ and $S(\theta_0; \theta)$ respectively he noted that for θ sufficiently close to θ_0 it is possible to make the approximation

$$I(\theta_0; \theta_0 + \Delta\theta) \approx \frac{1}{2}\|\Delta\theta\|_{J^2},$$

where $\|\Delta\theta\|_{J^2} = \Delta\theta^T J \Delta\theta$ and J is the positive definite Fisher Information matrix defined as

$$J_{ij} = \mathbb{E}\left\{\frac{\partial \log f(X \mid \theta)}{\partial \theta_i}\frac{\partial \log f(X \mid \theta)}{\partial \theta_j}\right\}.$$

Thus it follows that when the MLE $\hat{\theta}$ of θ_0 lies close to θ_0 a measure of the difference between the two functions $f\left(x \mid \hat{\theta}\right)$ and $f(x \mid \theta_0)$ is given by $\frac{1}{2}\|\hat{\theta} - \theta_0\|_{J^2}$. Akaike then observes that in the situation where the parameter

θ is restricted to some lower dimensional subspace which does not contain θ_0 it can be shown under certain regularity conditions that, if the choice θ' of θ which minimises $-S(\theta_0;\theta)$ is sufficiently close to θ_0, then the distribution of $N\|\hat{\theta}-\theta\|_{J^2}$ is approximately chi-squared with degrees of freedom equal to the dimension of the restricted parameter space. It therefore holds that

$$E_\infty 2NI\left(\theta_0;\hat{\theta}\right) = N\|\theta' - \theta_0\|_{J^2} + d, \tag{4.9}$$

where E_∞ denotes the mean of the approximate distribution and d is the dimension of the subspace.

As already stated, minimising the Kullback-Leiber distance is a sensible criterion for model selection, so some approximation to the expression on the right-hand side of (4.9) could make a useful model selection tool. The asymptotic distribution of $\sqrt{N}\left(\hat{\theta}-\theta'\right)$ is usually approximately normal with mean 0 and variance matrix J^{-1}. Hence, if we use

$$2\left(\sum_{i=1}^{N}\log f(x_i \mid \theta_0) - \sum_{i=1}^{N}\log f\left(x_i \mid \hat{\theta}\right)\right) \tag{4.10}$$

as an estimate of $N\|\theta' - \theta_0\|_{J^2}$, then we must correct for the downward bias introduced by replacing θ' by $\hat{\theta}$ by adding d to (4.10). Hence minimising the Kullback-Leiber distance is equivalent to minimising

$$2\left(\sum_{i=1}^{N}\log f(x_i \mid \theta_0) - \sum_{i=1}^{N}\log f\left(x_i \mid \hat{\theta}\right)\right) + 2d$$

From this we derive Akaike's model selection criterion, which is to select

the model which maximises the quantity

$$\mathrm{AIC}\left(\hat{\theta}\right) = \log\left\{P\left(D \mid \hat{\theta}\right)\right\} - d.$$

This scoring function is intuitively attractive as it contains both a term measuring how well our model fits the observed data and a term penalising model complexity.

4.5 The Bayesian Information Criterion (BIC)

This model selection criterion was first suggested by Schwarz in 1978 [66]. Like Akaike he was attempting to modify maximum likelihood so that it could be applied to the problem of choosing between a number of different families of model. By considering the asymptotic behaviour of Bayes estimators he derives an approximation to $\log P(D \mid \boldsymbol{m})$ which is independent of both the prior distribution of the parameters and the basis chosen for the parameters. If we consider Equation (4.5), and retain only those terms which increase with N then we are left with the terms $\log P\left(D \mid \tilde{\boldsymbol{\theta}}_{\boldsymbol{m}}, \boldsymbol{m}\right)$, which increases linearly with N, and $\log|A|$, which increases as $d \log N$. Also, for large N we observe that $\tilde{\boldsymbol{\theta}}_{\boldsymbol{m}}$ can be approximated by $\hat{\boldsymbol{\theta}}_{\boldsymbol{m}}$, and hence we obtain

$$\log P(D \mid \boldsymbol{m}) \approx \log P\left(D \mid \hat{\boldsymbol{\theta}}_{\boldsymbol{m}}, \boldsymbol{m}\right) - \frac{d}{2} \log N.$$

Schwarz observes that his procedure differs from Akaike's only in that the dimension is multiplied by $\frac{1}{2} \log N$ and notes that this will give his procedure a tendency to choose more parsimonious models (provided there are 8 or more

observations).

It is interesting to note that the BIC is the negative of the Minimum Description Length (MDL) criterion described in Rissanen [62]. Rissanen arrives at this model selection criterion by considering the concept of stochastic complexity, which can be considered to be a formal measure of the level of randomness in the data relative to the selected model. He shows that his MDL criterion approximates this stochastic complexity, where the minimum description length is the minimum number of binary digits required to describe the observed data given the selected model.

4.5.1 Draper's Scoring Function

Draper [21] suggests another approximation to Equation (4.5) in which he also retains the term $\frac{d}{2}\log(2\pi)$:

$$\log P(D \mid \boldsymbol{m}) \approx \log P\left(D \mid \hat{\theta}_{\boldsymbol{m}}, \boldsymbol{m}\right) - \frac{d}{2}\log N + \frac{d}{2}\log(2\pi)$$

He claims that in his experience the inclusion of this term gives more accurate results than the traditional BIC approximation.

4.6 Cheeseman & Stutz's (CS) Scoring Function

4.6.1 Marginal Likelihood of the Expected Data (MLED)

When using the EM algorithm to fit our model to the data we treat expected sufficient statistics as though they are actual sufficient statistics. This suggests another possible approximation to the marginal likelihood, the Marginal Likelihood of the Expected Data (MLED),

$$\log P\left(D \mid \boldsymbol{m}\right) \approx \mathrm{MLED}\left(\boldsymbol{\theta}_{\boldsymbol{m}}\right) = \log P\left(D' \mid \boldsymbol{m}\right),$$

where D' is an imaginary dataset which is consistent with the expected sufficient statistics. This means that, for our Bayesian networks, MLED is given by

$$\mathrm{MLED}\left(\boldsymbol{\theta}_{\boldsymbol{m}}\right) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma\left(\alpha_{ij}\right)}{\Gamma\left(\alpha_{ij} + \mathbb{E}\left(N_{ij} \mid \hat{\boldsymbol{\theta}}_{\boldsymbol{m}}\right)\right)} \prod_{k=1}^{r_i} \frac{\Gamma\left(\alpha_{ijk} + \mathbb{E}\left(N_{ijk} \mid \hat{\boldsymbol{\theta}}_{\boldsymbol{m}}\right)\right)}{\Gamma\left(\alpha_{ijk}\right)}.$$

As we are utilising expected sufficient statistics computed as part of the EM algorithm this measure is computationally efficient, and because we are computing a marginal likelihood we again punish model complexity. However, one problem with this scoring function is that there is no guarantee of its asymptotic correctness.

4.6.2 Cheeseman & Stutz's (CS) Scoring Function

Cheeseman and Stutz [12] observed that the equivalence between $P(D \mid \boldsymbol{m})$ and $P(D' \mid \boldsymbol{m})$ would hold only if the quantities

$$w_{ij} = P(\text{observation } i \text{ comes from distribution } j)$$

used in the calculation of $P(D' \mid \boldsymbol{m})$ are indicator functions, that is $w_{ij} \in \{0, 1\}$ and $\sum_j w_{ij} = 1$, and that as they diverge from indicator values the value of $P(D' \mid \boldsymbol{m})$ will be substantially less than $P(D \mid \boldsymbol{m})$. They suggest that it is possible to compensate for this by multiplying $P(D' \mid \boldsymbol{m})$ by the correction term

$$\frac{P\left(D \mid \hat{\boldsymbol{\theta}}_{\boldsymbol{m}}, \boldsymbol{m}\right)}{P\left(D' \mid \hat{\boldsymbol{\theta}}_{\boldsymbol{m}}, \boldsymbol{m}\right)}.$$

Hence they arrive at the approximation

$$\log P(D \mid \boldsymbol{m}) \approx \log P(D' \mid \boldsymbol{m}) + \log P\left(D \mid \hat{\boldsymbol{\theta}}_{\boldsymbol{m}}, \boldsymbol{m}\right) - \log P\left(D' \mid \hat{\boldsymbol{\theta}}_{\boldsymbol{m}}, \boldsymbol{m}\right).$$

Chickering and Heckerman [16] note that this approximation can be derived by applying the BIC approximation to the identity

$$P(D \mid \boldsymbol{m}) = P(D' \mid \boldsymbol{m}) \frac{\int P(D \mid \boldsymbol{\theta}_{\boldsymbol{m}}, \boldsymbol{m}) P(\boldsymbol{\theta}_{\boldsymbol{m}} \mid \boldsymbol{m}) \, d\boldsymbol{\theta}_{\boldsymbol{m}}}{\int P(D' \mid \boldsymbol{\theta}_{\boldsymbol{m}}, \boldsymbol{m}) P(\boldsymbol{\theta}_{\boldsymbol{m}} \mid \boldsymbol{m}) \, d\boldsymbol{\theta}_{\boldsymbol{m}}}.$$

4.7 Integrated Classification Likelihood (ICL)

Biernacki *et al.* [3] propose this method as an alternative to Schwarz's BIC when considering mixture distributions. They point out that the validity of the BIC for assessing the number of components in a mixture distribution is the subject of some concern. The BIC estimate is valid only when the estimated parameters lie well within the parameter space. In the case in which the number of mixture components, c, in the model being fitted is greater than the true number of components, c', the estimates of $c - c'$ of the mixing proportions will tend to 0 as the sample size increases. The implication of this is that we can expect our necessary conditions for the validity of the BIC to be breached when we are fitting models larger than the true model.

They propose an Integrated Classification Likelihood (ICL) criterion as a possible alternative. In the following discussion we consider the application of the ICL criterion to naive Bayesian networks. We will use $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)$ to denote the observed data, $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N)$ to denote the unobserved data and $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$ to denote the complete data. In a slight change to previous notation we now define $\boldsymbol{z}_l = (z_{l1}, \ldots, z_{lr_z})$ where $z_{lj} = 1$ if observation l comes from component j and 0 otherwise. The classification log-likelihood, also known as the complete log-likelihood, of $\boldsymbol{\theta}_m$ for the complete sample $\boldsymbol{x}$ is then given by

$$\mathcal{CL}(\boldsymbol{\theta}_m) = \sum_{l=1}^{N} \sum_{k=1}^{r_z} z_{lk} \log \left(p_k P\left(\boldsymbol{y}_i \mid Z_{lk} = 1, \boldsymbol{\theta}_m \right) \right),$$

This complete log-likelihood is related to the observed log-likelihood by the

relationship

$$\mathcal{CL}(\theta_m) = \mathcal{L}(\theta_m) - \mathcal{EC}(\theta_m), \tag{4.11}$$

where

$$\mathcal{EC}(\theta_m) = -\sum_{k=1}^{r_z}\sum_{l=1}^{N} z_{lk} \log t_{lk} \geq 0,$$

with

$$t_{lk} = \frac{p_k P(y_l \mid Z_{lk} = 1, \theta_m)}{\sum_{j=1}^{r_z} p_j P(y_l \mid Z_{lk} = 1, \theta_m)} \quad 1 \leq i \leq n \text{ and } 1 \leq k \leq r_z$$

denoting the conditional probability that y_l arises from the k^{th} mixture component.

Equation (4.11) shows that the classification log-likelihood can be thought of as the log-likelihood penalised by $-\mathcal{EC}(\theta_m)$. Biernacki *et al.* further show that $\mathcal{EC}(\theta_m)$ has expected value

$$\mathbb{E}(\mathcal{EC}(\theta_m)) = -\sum_{l=1}^{N}\sum_{k=1}^{r_z} t_{lk} \log t_{lk} \geq 0.$$

This quantity, the entropy, is a measure of how well the model partitions the observed data, taking smaller values when the model provides a good description of the data. This has led to the use of the variable $\mathcal{CL}(\theta_m)$ in model selection, with the unknown $\boldsymbol{Z}$ being replaced with $\tilde{\boldsymbol{z}}$ where

$$\tilde{z}_{lk} = \begin{cases} 1 \text{ if} \arg\max_k t_{lk}\left(\hat{\theta}_m\right) = k \\ 0 \text{ otherwise.} \end{cases} \tag{4.12}$$

However, as Biernacki *et al.* point out, in practice this is a poor model selection criterion, not least because it tends to over-estimate the size of model required

as it does not contain a term for penalising the number of parameters in the model. It is this shortcoming of the classification likelihood as a model selection criterion that in part motivates Biernacki *et al.* to propose their integrated classification likelihood (ICL) criterion. Rather than looking to maximise the integrated likelihood as in criteria such as BIC, they look to maximise the integrated classification likelihood,

$$f(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}_m) = \int_{\Theta_m} f(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}_m, m)\, \pi(\boldsymbol{\theta}_m \mid m)\, d\boldsymbol{\theta}_m. \tag{4.13}$$

They argue that this is a more appropriate criterion than the integrated observed likelihood as the classification likelihood takes into account the ability of the model to give evidence for the data's clustering structure. However, it is still a problem that the BIC approximation for the integral given in (4.13) is not strictly valid. To overcome this they note that the priors on our parameters are independent, which allows them to rewrite (4.13) as

$$f(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}_m) = \int_{\Theta_m} f(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\theta}_m, m)\, f(\boldsymbol{z} \mid \boldsymbol{\theta}_m, m)\, \pi(\boldsymbol{\theta}_m \mid m)\, d\boldsymbol{\theta}_m. \tag{4.14}$$

The BIC approximation is then valid for $f(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\theta}_m, m)$, and $f(\boldsymbol{z} \mid \boldsymbol{\theta}_m, m)$ can be calculated exactly. Using a Dirichlet prior, $Di(\alpha, \ldots, \alpha)$, for $\boldsymbol{p}$ we have

$$f(\boldsymbol{z} \mid \boldsymbol{\theta}_m, m) = \int p_1^{N_1} \ldots p_{r_z}^{N_{r_z}} \frac{\Gamma(r_z\alpha)}{\Gamma(\alpha)\ldots\Gamma(\alpha)} p_1^{\alpha-1} \ldots p_{r_z}^{\alpha-1} d\boldsymbol{p},$$

where N_k is the number of observations in which $z_{lk} = 1$. It follows that

$$f(\boldsymbol{z} \mid \boldsymbol{\theta}_m, m) = \frac{\Gamma(r_z\alpha)}{\Gamma(\alpha)^{r_z}} \frac{\Gamma(N_1 + \alpha)\ldots\Gamma(N_{r_z} + \alpha)}{\Gamma(N + r_z\alpha)}.$$

So if we use the Jeffreys non-informative prior ($\alpha = 1/2$), and bear in mind that $\boldsymbol{z}$ is in fact unobserved, the integrated log-likelihood takes the form

$$\begin{aligned}\log f(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}_m, m) \approx & \max_{\boldsymbol{\theta}_m} \log f(\boldsymbol{x} \mid \tilde{\boldsymbol{z}}, \boldsymbol{\theta}_m, m) - \frac{d}{2} \log N \\ & + \log \Gamma\left(\frac{r_z}{2}\right) + \sum_{k=1}^{c} \log \Gamma\left(N_k + \frac{1}{2}\right) \\ & - r_z \log \Gamma\left(\frac{1}{2}\right) - \log \Gamma\left(N + \frac{r_z}{2}\right), \end{aligned} \tag{4.15}$$

where $\tilde{\boldsymbol{z}}$ is defined by 4.12.

4.8 Model Selection Using Simulated Data

4.8.1 Introduction

In the discussion that follows we will attempt to study the similarities and differences between the model selection techniques described in this chapter. We will look at these techniques both in terms of their ability to determine the number of latent states present in the population under consideration and their computational complexity. The models considered here will typically be quite small, having at most a dozen or so binary observable variables and few latent states.

We begin by considering the effect that Mackay's [43] proposed change of basis has on the Laplace approximation when used in model selection. We will then look at the use of this and other approximations in model selection. First we consider the simple case in which we have a single latent variable in the network, then we extend this work to consider the case in which we have two

or more independent latent variables. We hope to find that at least some of the model selection techniques being considered will be able to discern the presence of multiple latent variables. Finally we will apply what we have learned about these model selection techniques to a 'real life' problem, relating to a study of the survival of patients following admission to an adult intensive care unit (ICU) [35].

4.8.2 Choice of Prior Distributions

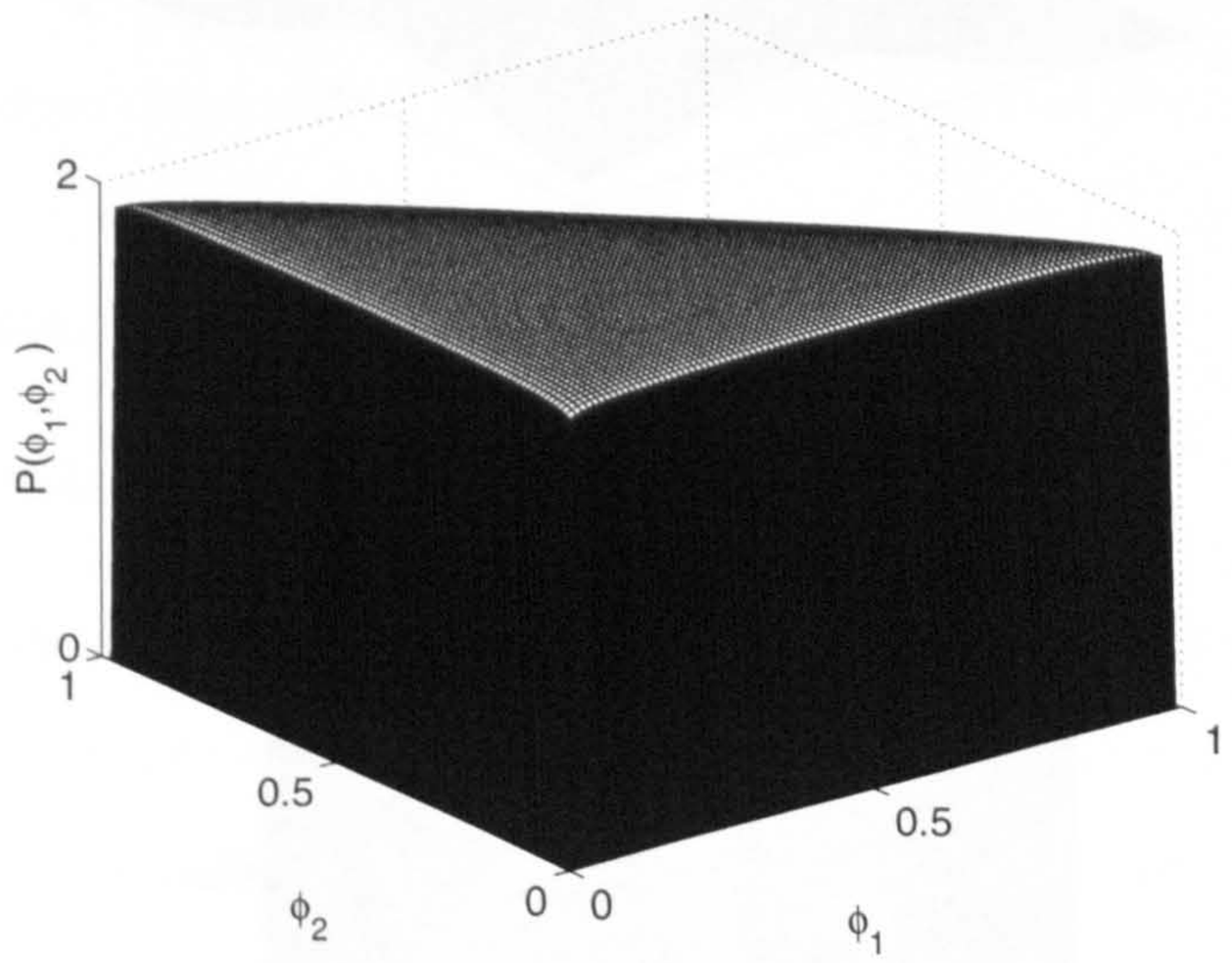
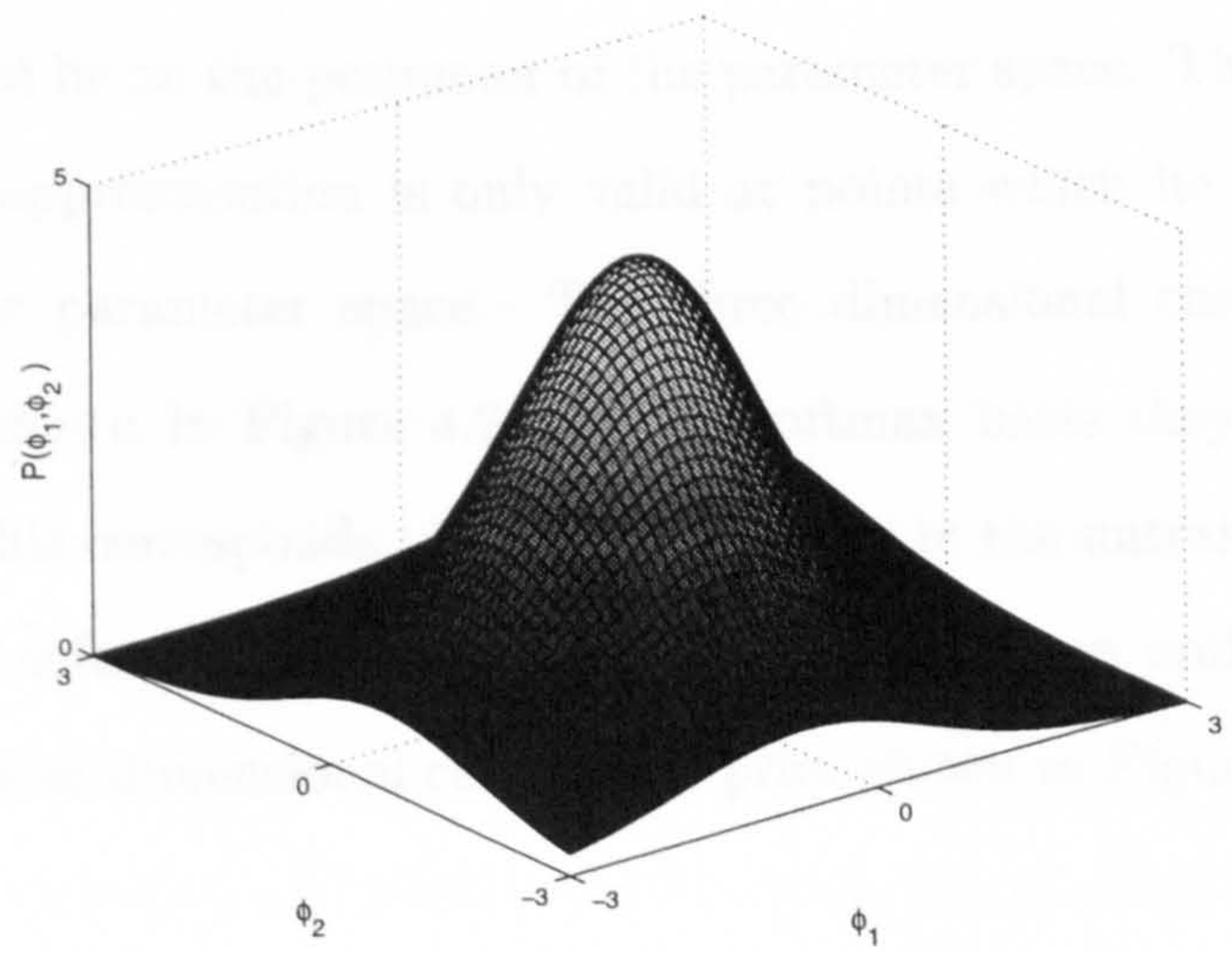


Figure 4.2: A three dimensional Dirichlet distribution in the traditional basis with $\alpha = 1.01$.

In considering the MAP parameter estimates we will use the prior distributions proposed by Chickering and Heckerman [16]; in the traditional basis we will use the Dirichlet distribution in which all of the shape parameters are selected to be $\alpha = 1.01$ and in the softmax basis we will use the Dirichlet distribution in which all of the shape parameters are chosen to be $\alpha = 1$.



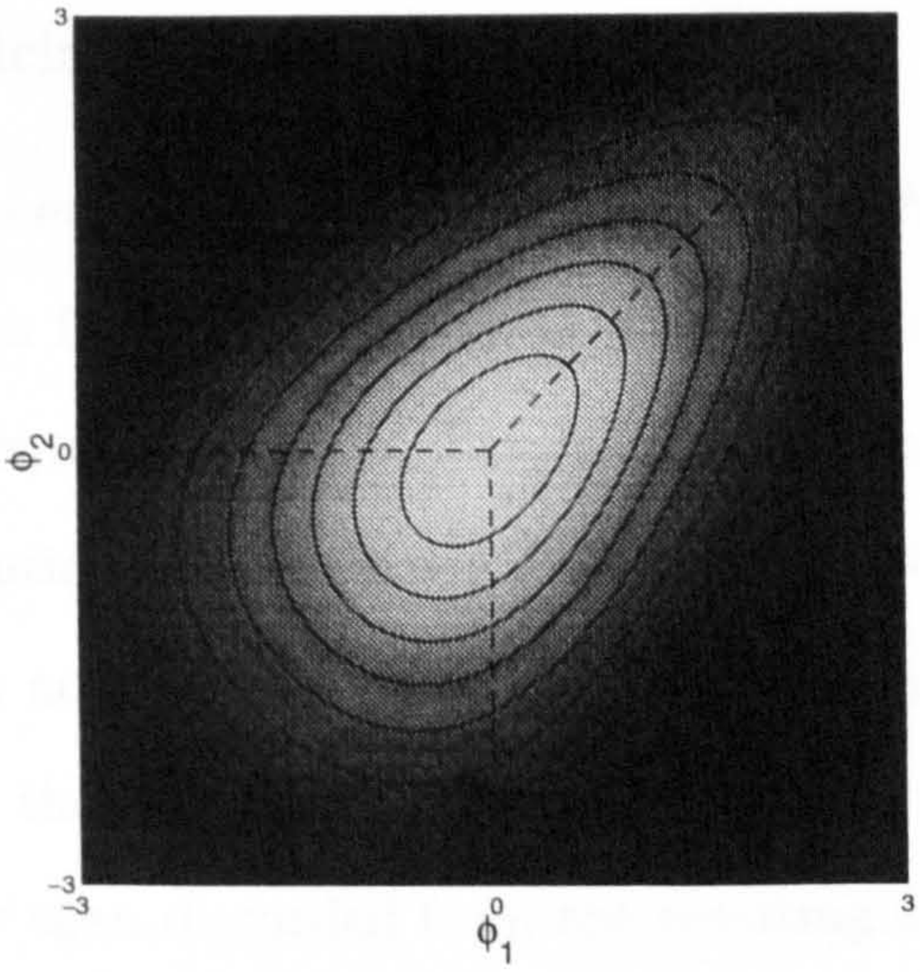


Figure 4.3: A three dimensional Dirichlet distribution transformed into softmax space with $\alpha = 1$.

Chickering and Heckerman choose to use $\alpha = 1.01$ in the traditional basis because it gives a prior which is close to uniform ($\alpha = 1$ would correspond to a uniform prior) but which is sufficiently non-uniform to ensure that our parameter estimates will not lie on the perimeter of the parameter space. This is important as our Laplace approximation is only valid at points which lie away from the boundary of our parameter space. The three dimensional case of this prior distribution is shown in Figure 4.2. In the softmax basis they choose to use $\alpha = 1$ because this corresponds to the uniform prior in the untransformed space. However, in the softmax basis the resulting prior is far from uniform, as can be seen from the three dimensional case of this prior shown in Figure 4.3

4.8.3 Experiment Methodology

Generation of Artificial Datasets

First a naive Bayesian network known as the *generative network* was set up. The number of latent states for the generative network was defined along with the number of binary observable variables. Parameters for this network were drawn from a uniform distribution. To generate a dataset of size N we begin by drawing a value for the hidden node Z from its known distribution. We can then draw values for the Y_i from their conditional distributions given this value of Z. All observations of Z were then discarded from the resulting dataset.

Model Selection

Before selecting our 'best' model we must first define the set of test models we are considering. In each case the set of test models were taken to be the set of

models with the same structure as the generative network except that the number of latent states was allowed to vary. Typically r_z was allowed to vary between 1 and some pre-determined maximum value r_{max}. For each of these test models the model had to be fitted to the data before any model selection could take place. The EM algorithm was used to do this and a variant of the multiple re-start technique was used in an attempt to find the global maximum of the likelihood function (for the ML parameter estimates) or the posterior distribution of the network parameters (for the MAP parameter estimates). This method was based upon that used by Chickering and Heckerman [15]. For each test model we began with 64 different randomly generated starting points, ran one step of the EM algorithm on each and retained the 32 resulting parameter configurations that gave the largest values of the function being maximised. We than ran 2 EM steps on each of these 32 configurations retaining the 16 best configurations upon which we ran 4 steps of the EM algorithm. This was continued until only one configuration remained; we then ran the EM algorithm on this configuration until convergence was reached. Each of the model selection criteria was applied to the resulting model. The procedure was slightly different for the test model with $r_z = 1$ in that the value of $P(D \mid m)$ could be calculated exactly in this case using Equation 4.1.

4.8.4 Effect of the Change of Basis

MacKay [43] justifies his proposal to work in the softmax basis rather than the traditional basis by demonstrating that a Dirichlet distribution can be better approximated by a Gaussian when transformed into the softmax basis; see section

4.3.1. He then uses an example in which both the prior and posterior distributions of the variables are Dirichlet to demonstrate the superiority of his approach. While he states that he can see no reason why this approach cannot successfully be extended to a latent variable model he makes no attempt to demonstrate this, nor do Chickering and Heckerman [15], simply citing MacKay's paper as justification for choosing to work in the softmax basis. Although in latent variable problems we will generally choose to work with Dirichlet priors, it does not follow that the marginal distributions, which are what we are really interested in approximating, will also be Dirichlet. We conducted a simple experiment in which a simple network with 2 latent states and 4 observed variables was used to generate datasets of size 100. The parameters of the observed variables were then assumed to be known so that we had a model with a single unknown parameter, corresponding to the component weights, and the EM algorithm was used to find the value of this parameter in the two bases. In the traditional basis a Dirichlet prior was used with $\alpha = 1.01$ and in the softmax basis a transformed Dirichlet prior with $\alpha = 1$ was used. A number of datasets were generated, some in which the true parameter was in the centre of the traditional parameter space (that is close to 0.5), some in which it was close to the boundary of the traditional parameter space (that is close to 0 or 1), and some intermediate cases.

Figure 4.4 shows some typical examples of the resulting marginal distributions and approximating Gaussian curves. It can be seen that for the central and intermediate parameter values the Laplace approximation is fairly accurate in both bases, and in fact there is some evidence that it is more accurate in the traditional basis. This is perhaps to be expected as the prior distribution being used in the traditional basis is virtually uniform in this region and so the shape

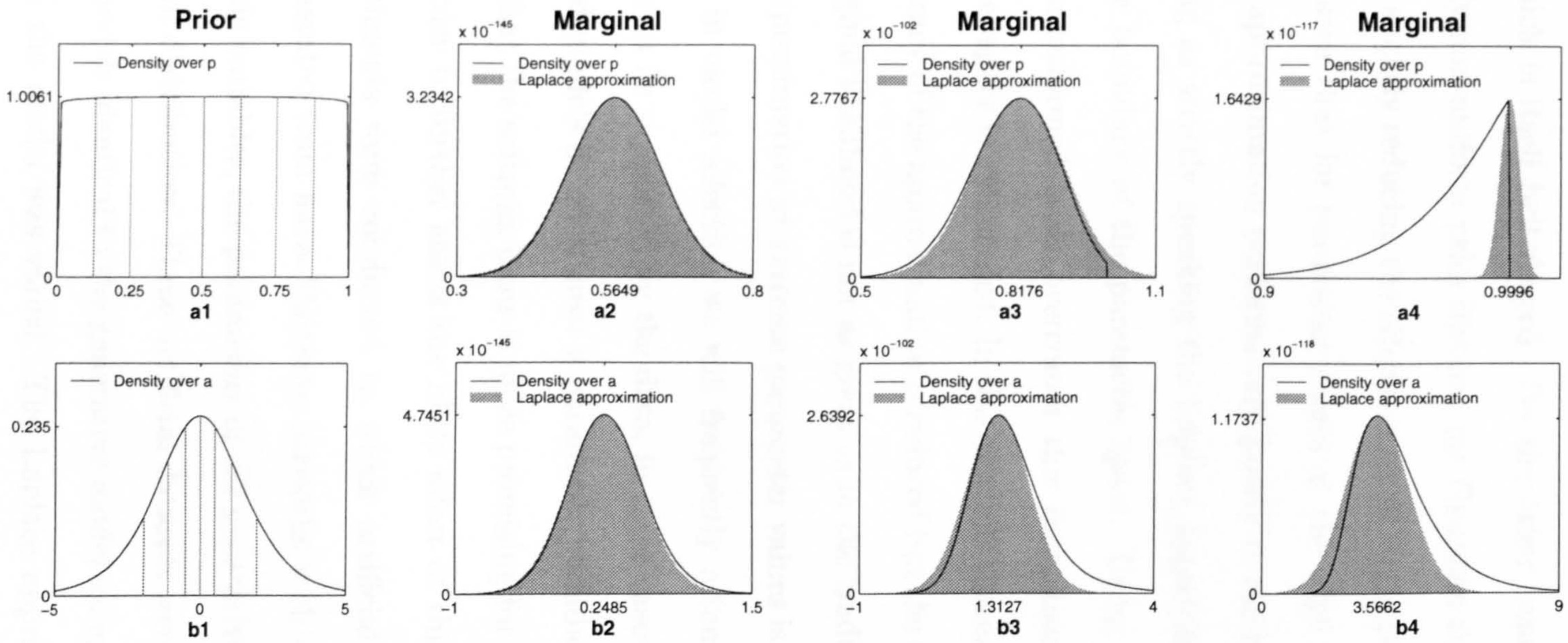


Figure 4.4: The accuracy of the Laplace approximation depends upon how far the point about which we are approximating lies from the boundary of the parameter space. This figure looks at how the approximation fares as we move from a point lying well within the parameter space, frames a2 and b2, to a point lying at the extremes of the parameter space, frames a4 and b4. The prior distribution is given in frames a1 and b1, a1 being a Dirichlet prior with $\alpha = 1.01$ and b1 being a transformed Dirichlet prior in the softmax basis with $\alpha = 1$.

of the marginal distribution depends only on the shape of the likelihood of the data which is itself bell-shaped. On the other hand, in the softmax basis the distinctly non-uniform prior distorts the Gaussian shape of the likelihood of the data, so slightly reducing the effectiveness of the approximation. However, it can also be seen that for parameter values at the edge of the parameter space the Laplace approximation performs very poorly in the traditional basis. This is not surprising as strictly speaking the Laplace approximation is only defined away from the boundary of the parameter space. Using the Laplace approximation in the transformed basis overcomes this limitation because in this basis the parameter space is unbounded. In this example this extreme value still means that the accuracy of the approximation is reduced but the resulting underestimation of the marginal likelihood is not as great as in the traditional basis. The behaviour of the approximation at extreme parameter values is an important consideration because in model selection we will frequently attempt to fit models which are too large to be supported by the data; in these cases we can expect to find that our fitted models lie on or near a parameter boundary. For this reason it would appear that the softmax basis is more promising for use in model selection.

In order to further assess the likely effect of this change of basis a number of experiments were conducted in which artificial datasets of size $N = 50$ were generated from naive Bayesian networks with 4 latent states and 8 binary observable variables, the parameters of the generative network being drawn from a uniform distribution. These artificial datasets were then used to fit a number of test models identical to the generative model except that the number of latent states in the model was varied. The Laplace approximation was then applied to the models which were fitted using the EM algorithm. Figure 4.5 shows how

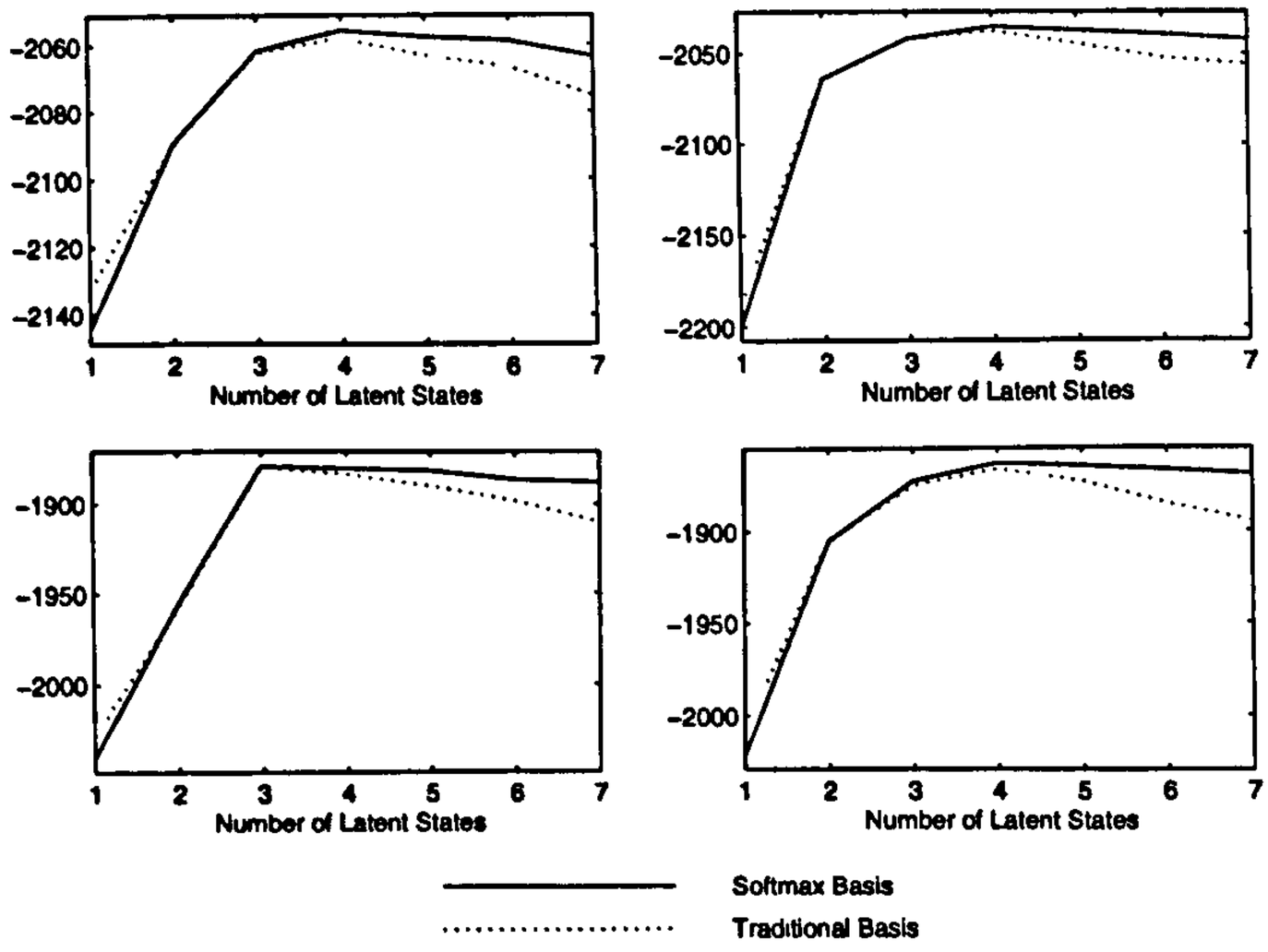


Figure 4.5: Comparison of the Laplace approximation in traditional and softmax parameterisations when applied to model selection on datasets of size $N = 50$ generated from a model with 4 latent states and 8 binary observable variables.

the Laplace approximation differed in the two bases. For smaller test models we would not expect the fitted model to lie close to the boundary of the parameter space, and in these cases the Laplace approximation is very similar in each of the bases. However when we are fitting larger models we can expect there to be problems in the traditional basis as the parameter estimates are likely to lie on, or close to, the perimeter of the parameter space. In this situation we have seen that both methods will have a tendency to underestimate the marginal probability of interest, but that the effect will be greater in the traditional basis. Figure 4.5 bears this out as it can be clearly seen that for these larger models the Laplace approximation in the traditional basis leads to a smaller estimate of the marginal probability than in the softmax basis.

The implication of this is that when using the Laplace approximation for

model selection we can expect to achieve more accurate results by working in the softmax basis.

4.8.5 Aliasing

Chickering and Heckerman [16] draw attention to a phenomenon they term 'aliasing' which appears to have been neglected in the other literature on model selection using the marginal likelihood $P(D \mid m)$. They note that, because our root node Z is hidden, the likelihood $P(D \mid \phi_m, m)$ will be invariant to arbitrary relabelling of the states of Z. Each of these equivalent relabellings is known as an 'alias', and if each alias is non-degenerate, that is to say they are all unique, then there will be $r_z!$ of them. This means that rather than our likelihood surface having a single global maximum we can expect it to have $r_z!$ such maxima. If we assume that the aliases are well separated then we can determine our approximations by fitting a Gaussian to each alias, calculating the volume under each of these Gaussians and summing these values together. In effect we calculate the marginal probability for one alias and multiply it by $r_z!$. This correction factor applies to all of our scoring functions which are attempting to approximate $P(D \mid m)$.

In Chickering and Heckerman's experiments this aliasing did not have a serious effect on the results as they found that the MAP parameter estimates were always non-degenerate, but for the models we worked with degeneracy did prove to be a problem.

We found that when we fitted test models with larger numbers of latent states the EM algorithm had a definite tendency to converge to one of these degenerate

	Component						
	1	2	3	4	5	6	7
Weight	0.204	0.022	0.048	0.172	0.022	0.307	0.225
θ_1	0.822	0.501	0.309	0.934	0.501	0.188	0.258
θ_2	0.522	0.435	0.709	0.832	0.435	0.339	0.047
θ_3	0.294	0.451	0.261	0.876	0.451	0.201	0.393
θ_4	0.536	0.514	0.751	0.867	0.514	0.139	0.527
θ_5	0.301	0.463	0.206	0.587	0.463	0.084	0.899
θ_6	0.890	0.542	0.454	0.208	0.542	0.788	0.101
θ_7	0.950	0.664	0.669	0.948	0.664	0.860	0.927
θ_8	0.843	0.521	0.275	0.296	0.521	0.907	0.681

Table 4.1: MAP_S parameter estimates for a model with 7 latent states and 8 binary observable variables. Weights of each component are given across the top of the table and rows in the table correspond to the probabilities of an observed variable taking the value 1 given membership of each component.

configurations. In the example shown in Table 4.1 we have generated a dataset of size 100 from a model with 4 latent states and 8 binary observable variables. Here we are fitting a test model to these data with 7 latent states and we find that two components, here labelled 2 and 5, are identical. What this means is that rather than having 7! peaks our likelihood surface will only have $7!/2!$ peaks. If, as here, we spot these problems when they occur we can easily compensate for them by altering the correction factor by which we multiply our scoring functions. This does of course raise the issue of whether or not it is sensible to even consider these models, as the obvious implication here is that we are fitting a model with too many components. See Crawford [18] for further discussion of this issue.

4.8.6 The Candidate Method

Chickering and Heckerman observed that the Candidate method failed to converge if they worked with naive Bayesian networks containing fewer than 32 observable variables, an observation that was confirmed by our results. Figure

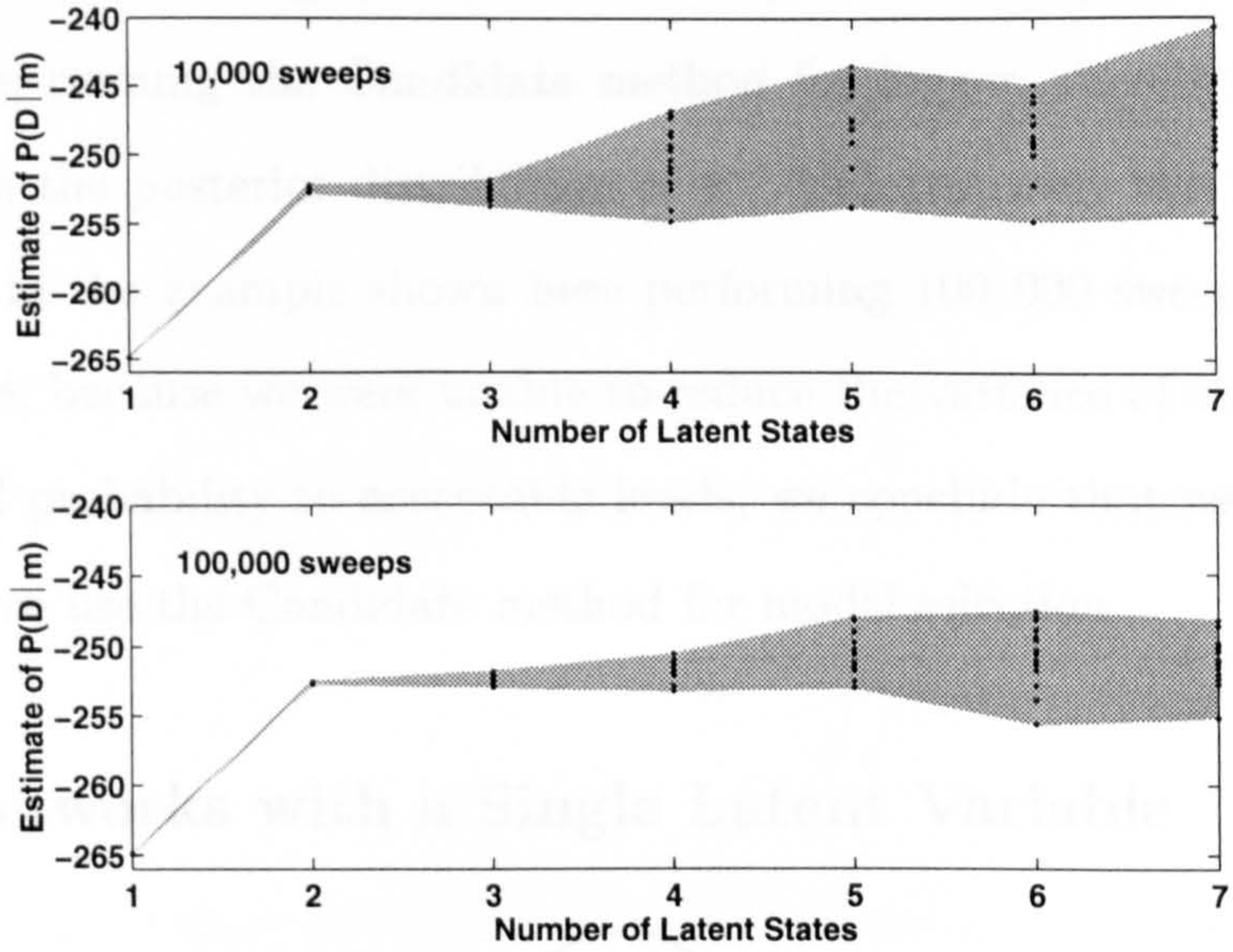


Figure 4.6: The results of applying the candidate method to a test model with 8 binary observable variables and 4 latent states to a dataset of size $N = 50$ generated from a network with the same structure. The candidate method was applied 15 times, the top figure shows the spread of estimates for $P(D \mid \boldsymbol{m})$ given when 10,000 sweeps of the algorithm were used in calculating the estimate and the bottom figure shows the spread of estimates when 100,000 sweeps were used.

4.6 shows what happened when we used the Candidate method on a simple model in which a dataset of size 50 was generated from a model with 4 latent states and 8 binary observable variables. We ran the candidate method 15 times and considered the resulting approximation to the marginal probability of the data given the model after $10,000$ and $100,000$ sweeps. It can be seen that the

method converges relatively well for the smaller models but that convergence was poor for the test models with a greater number of latent states. Increasing the number of sweeps used to calculate the approximation had the expected result of reducing the variability in the approximation but even after 100,000 sweeps the approximation was highly unstable. Clearly we could expect to decrease the variability by running the Candidate method for longer, thereby taking more samples from the posterior distribution of θ^*. Unfortunately this proved to be impractical; in the example shown here performing 100,000 sweeps took many hours. Hence, because we were unable to reduce the variance of our estimate of the marginal probability to acceptable levels, we conclude that we cannot with any confidence use the Candidate method for model selection.

4.8.7 Networks with a Single Latent Variable

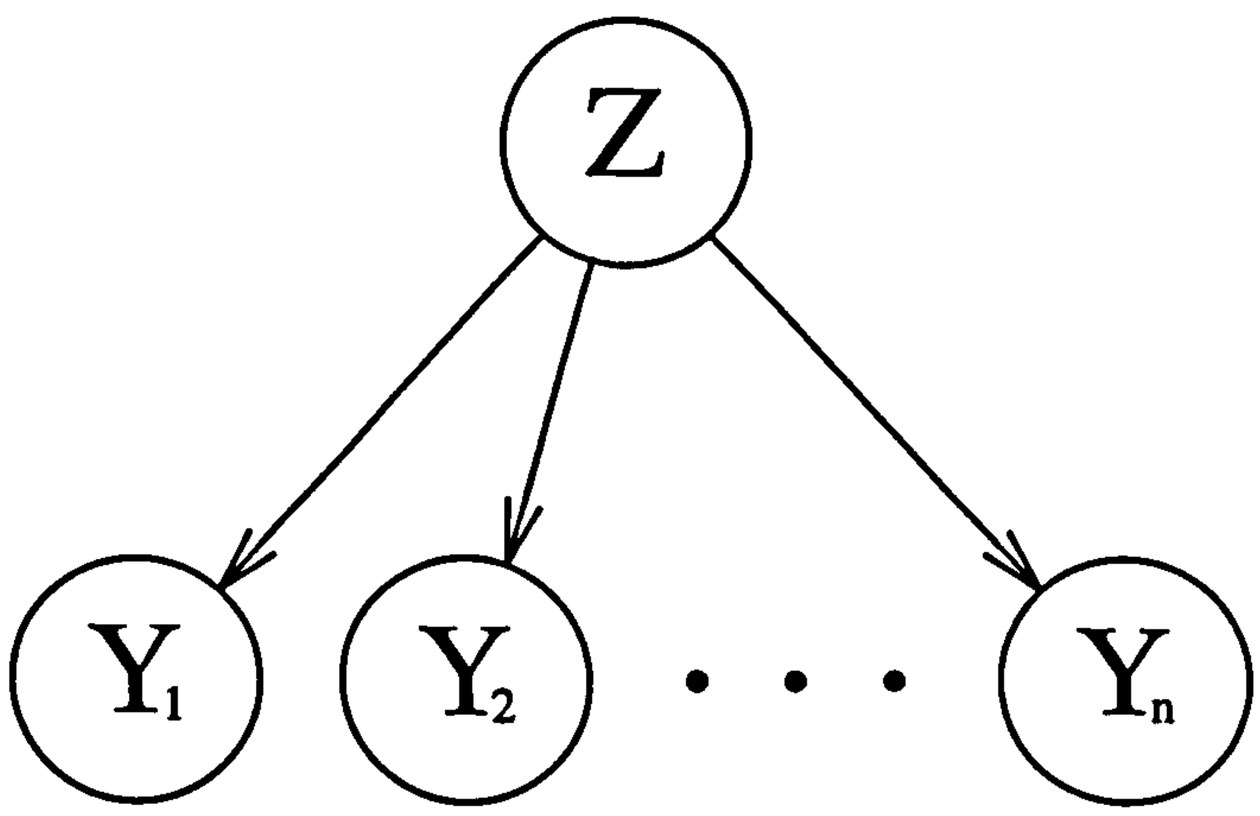


Figure 4.7: A naive Bayesian network with a single latent variable.

Here we consider networks with a single latent variable, such as that shown in Figure 4.7. Initially we will consider those scoring functions which are attempting

to approximate $\log P(D \mid \boldsymbol{m})$, namely the Laplace, Block, Diagonal, AIC [1], BIC, Draper and CS scoring functions. The Laplace, Block and Diagonal scoring functions will be evaluated at the MAP parameter estimates whereas the AIC, BIC and Draper scores should be evaluated at the ML parameter estimates. Cheeseman and Stutz [12] do not say whether their score ought to be evaluated at the MAP or ML parameter estimates, so for the sake of this comparison we follow Chickering and Heckerman [16] and consider this approximation at both points. We will then compare these scoring functions to the ICL scoring function which approximates $\log P(D, \boldsymbol{z} \mid \boldsymbol{m})$ rather than $\log P(D \mid \boldsymbol{m})$. The ICL scoring function is evaluated at the ML parameter estimates, as in the paper by Biernacki *et al.* [3].

Results

Figures 4.8 and 4.9 show how the model selection techniques typically compare. Taking the Laplace approximation as our base we can see that the Block and Diagonal approximations are both, as might be expected, very similar to the Laplace approximation though with a tendency to under-estimate $P(D \mid \boldsymbol{m})$, and are themselves almost indistinguishable. The BIC approximation has a tendency to greatly under-estimate the marginal probability of interest, with this tendency being more pronounced for larger models. This is perhaps unsurprising as we know that the assumptions for the BIC approximation break down in these larger models. The approximation suggested by Draper does seem to give more accurate results as he claimed. The Draper approximation is quite close to

[1] Strictly speaking AIC doesn't approximate $P(D \mid \boldsymbol{m})$ but is included here due to its similarity to BIC

the Laplace approximation for smaller models but tends to under-estimate the marginal probability for larger models, though not by the same degree as the BIC approximation. The AIC scoring function consistently returns higher values than the other scoring functions, and tends to be 'flatter' which may mean it is not very useful for model selection. Finally, the CS approximation seems to fare particularly badly; using both the MAP and ML parameter estimates it leads to a serious under-estimation of the marginal probability.

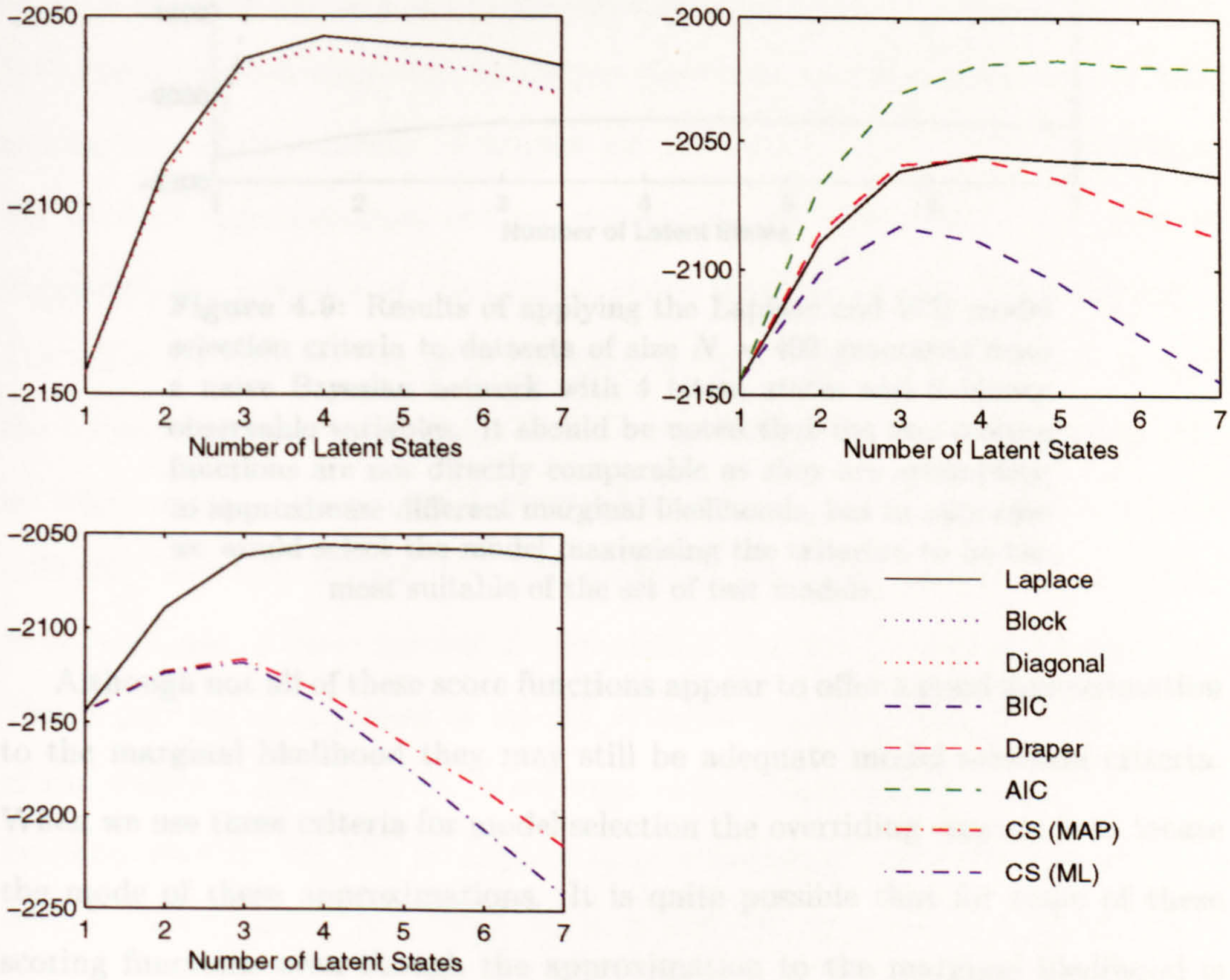


Figure 4.8: Results of applying a number of model selection criteria to datasets of size $N = 400$ generated from a naive Bayesian network with 4 latent states and 8 binary observable variables. The Laplace approximation is shown in each figure to aid comparison.

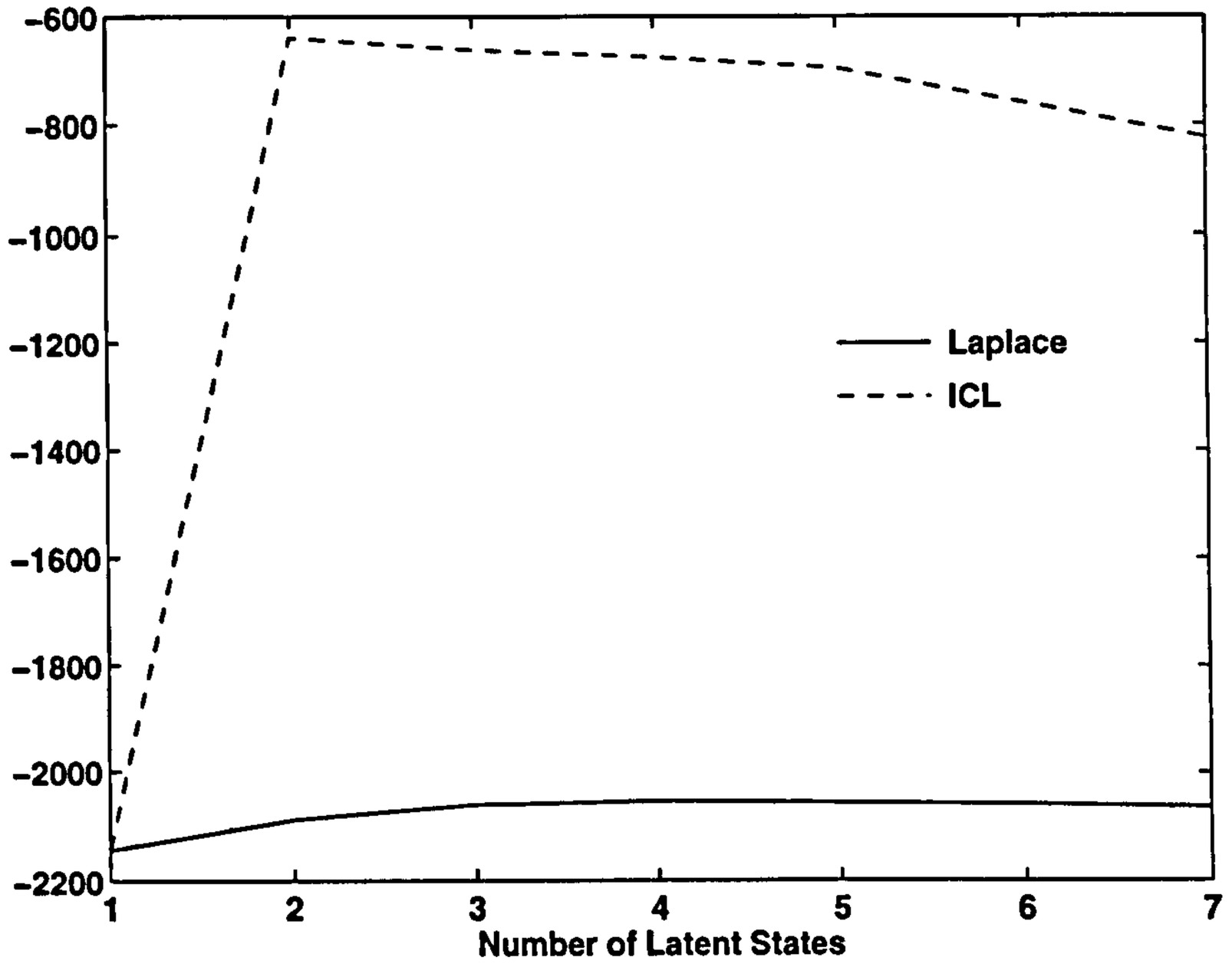


Figure 4.9: Results of applying the Laplace and ICL model selection criteria to datasets of size $N = 400$ generated from a naive Bayesian network with 4 latent states and 8 binary observable variables. It should be noted that the two scoring functions are not directly comparable as they are attempting to approximate different marginal likelihoods, but in each case we would select the model maximising the criterion to be the most suitable of the set of test models.

Although not all of these score functions appear to offer a good approximation to the marginal likelihood they may still be adequate model selection criteria. When we use these criteria for model selection the overriding concern is to locate the mode of these approximations. It is quite possible that for some of these scoring functions even though the approximation to the marginal likelihood is not itself very good the location of the mode of the scoring function may still be a useful criterion for model selection. Tables 4.2a-d show the location of these modes for a number of test models.

It would appear that the Block and Diagonal approximations give very similar results to the Laplace approximation; these two approximations selected a different model from the Laplace approximation on just one occasion. The AIC criterion seems to have a tendency to indicate the presence of more latent states than the Laplace approximation does, while the BIC approximation displays the opposite tendency. This would seem to be in agreement with the previously observed tendencies of the AIC and BIC scoring functions to respectively over-estimate and under-estimate $P(D \mid m)$. The adaptation of the BIC approximation suggested by Draper does seem to compensate for the BIC scoring function's tendency to suggest the presence of fewer latent states than the Laplace approximation, but on occasion it over-compensates and ends up suggesting the presence of more latent states than the Laplace approximation. The CS score function consistently selects models with fewer latent states than the Laplace approximation does, with a marked tendency to prefer models with no latent states. Comparison between the Laplace and ICL criterion are more difficult. The ICL criteria does not display a consistent pattern of indicating either a greater or lesser number of latent states than the Laplace approximation. This is perhaps unsurprising as the ICL criterion is looking at a different feature of the observed data from the other approximations considered here. However, deciding whether the ICL or Laplace approximation is leading to the selection of the correct model is difficult as it is difficult to visualise both the data and the models we are fitting. One possible reason for preferring the Laplace approximation over the ICL criterion is that on one occasion the ICL criterion suggested that the data contained 6 latent states, far more than the true number and a possible indication of instability in this model selection criterion.

Model Selection Criterion	Trial Number				
	1	2	3	4	5
Laplace	2	2	2	2	2
Block	2	2	2	2	2
Diagonal	2	2	2	2	2
AIC	2	4	2	2	2
BIC	2	2	2	2	2
Draper	2	4	2	2	2
CS (MAP)	1	1	1	1	1
CS (ML)	1	1	1	1	1
ICL	3	4	2	2	2

a: $N = 50$

Model Selection Criterion	Trial Number				
	1	2	3	4	5
Laplace	3	3	3	2	3
Block	3	3	3	2	3
Diagonal	3	3	3	2	3
AIC	3	3	3	2	3
BIC	2	3	2	2	2
Draper	3	3	3	2	3
CS (MAP)	2	2	2	2	2
CS (ML)	2	2	2	2	1
ICL	3	2	2	3	4

b: $N = 100$

Model Selection Criterion	Trial Number				
	1	2	3	4	5
Laplace	4	3	2	3	4
Block	4	3	2	3	3
Diagonal	4	3	2	3	3
AIC	5	5	4	3	3
BIC	3	2	2	2	3
Draper	3	3	3	2	3
CS (MAP)	2	1	1	2	2
CS (ML)	1	2	1	1	2
ICL	3	3	4	3	4

c: $N = 200$

Model Selection Criterion	Trial Number				
	1	2	3	4	5
Laplace	2	4	4	3	4
Block	2	4	4	3	4
Diagonal	2	4	4	3	4
AIC	3	5	4	3	4
BIC	2	3	3	3	3
Draper	2	4	4	3	4
CS (MAP)	2	3	2	3	3
CS (ML)	2	3	2	3	2
ICL	4	2	2	6	4

d: $N = 400$

Table 4.2: Number of latent states selected as optimal by 9 different scoring functions for datasets of size $N = 50$, 100, 200 and 400 generated from 20 different models with $n = 8$ binary observable variables and 4 latent states.

4.8.8 Networks with Two Latent Variables

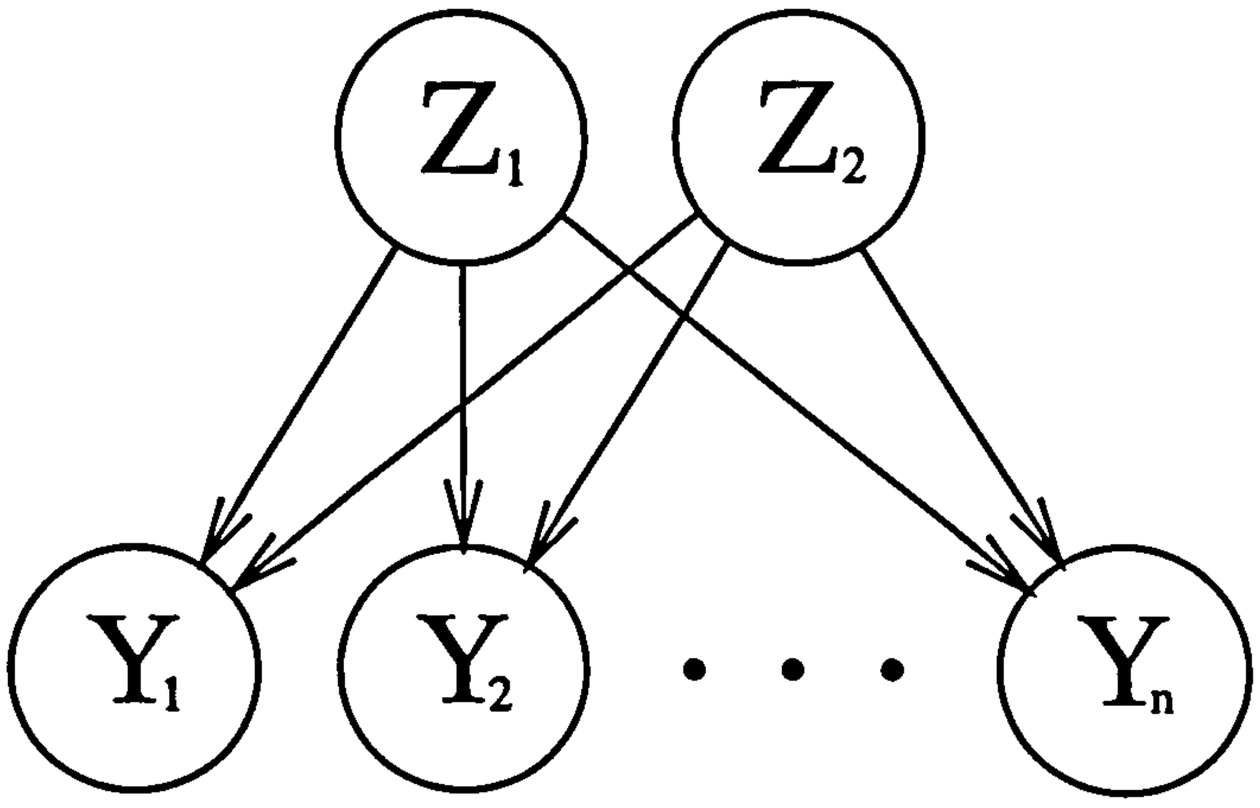


Figure 4.10: A naive Bayesian network with two latent variables.

Here we consider models with two latent variables such as that shown in Figure 4.10. In this model the two root nodes Z_1 and Z_2 are independent of each other and together form the set of parents of each of the observable variables. Figure 4.11 shows the results of applying these scoring functions to a range of test models fitted to a dataset of size $N = 400$ generated from a model with 8 binary observable variables and two latent variables each with 3 states. The results of applying these scoring functions to test models fitted to several different datasets are given in Tables 4.3 and 4.4.

The results are very similar to those seen in models with a single latent variable. The Block and Diagonal approximations are again very similar to the Laplace approximation, though with a slight tendency to under-estimate $\log P(D \mid \boldsymbol{m})$. The AIC criterion once again over-estimates the marginal likelihood while the BIC criterion under-estimates it; again this leads to the AIC having a tendency to pick more complicated models against the BIC's tendency

to select more parsimonious models. Draper's scoring function again goes some way to correcting the under-estimation of the BIC criterion. For the CS criterion we again find that it has a marked tendency to identify models with few if any latent states and here the same is also true of the ICL criterion.

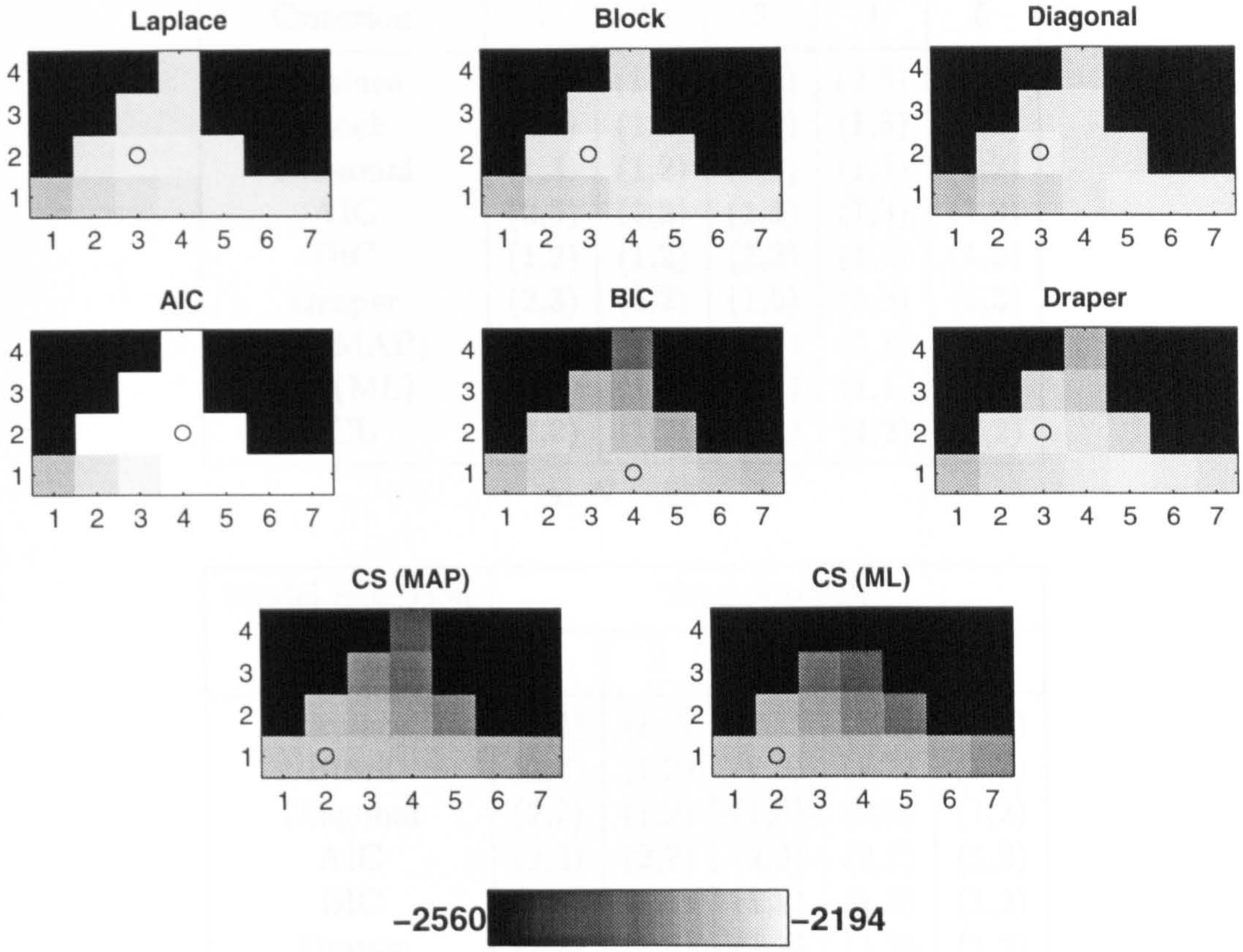


Figure 4.11: Results of applying a number of model selection criteria to datasets of size $N = 400$ generated from a naive Bayesian network with two latent variables each with 3 states and 8 binary observable variables. Figures along the x and y axes indicate the number of states the latent variables Z_1 and Z_2 have in each of the test models, noting of course that the model is invariant to relabelling of these two latent variables. The shading indicates the estimates of $P(D \mid \boldsymbol{m})$ for each test model, lighter shades indicating larger values of the scoring function in accordance with the key shown at the bottom of the figure. The circles indicate the models identified as best by each of the scoring functions.

Model Selection Criterion	Trial Number				
	1	2	3	4	5
Laplace	(1,7)	(1,7)	(1,7)	(2,3)	(1,7)
Block	(1,7)	(1,2)	(1,3)	(1,3)	(1,2)
Diagonal	(1,3)	(1,2)	(1,3)	(1,3)	(1,2)
AIC	(2,3)	(1,2)	(1,5)	(1,3)	(1,3)
BIC	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)
Draper	(2,3)	(1,2)	(1,5)	(1,3)	(1,3)
CS (MAP)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
CS (ML)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
ICL	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)

a: $N = 50$

Model Selection Criterion	Trial Number				
	1	2	3	4	5
Laplace	(2,2)	(1,2)	(2,2)	(2,3)	(1,2)
Block	(2,2)	(1,2)	(1,3)	(2,2)	(1,2)
Diagonal	(2,2)	(1,2)	(1,3)	(2,2)	(1,2)
AIC	(2,3)	(2,2)	(2,3)	(2,2)	(1,3)
BIC	(2,1)	(1,1)	(1,2)	(1,2)	(1,2)
Draper	(2,2)	(1,2)	(1,3)	(1,2)	(1,2)
CS (MAP)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
CS (ML)	(1,1)	(1,1)	(1,1)	(1,1)	(1,1)
ICL	(1,2)	(1,1)	(1,2)	(1,2)	(1,2)

b: $N = 100$

Table 4.3: Number of latent states selected as optimal by 9 different scoring functions for datasets of size N=50 and 100 generated from 10 different models with n=8 binary observable variables and two latent variables each with 3 states. The figures in brackets indicate the number of states of the two variables Z_1 and Z_2. It should be remembered that the model is invariant to the relabelling of Z_1 and Z_2 so (a, b) is equivalent to (b, a).

Model Selection Criterion	Trial Number				
	1	2	3	4	5
Laplace	(1,3)	(1,2)	(1,3)	(2,3)	(2,2)
Block	(1,3)	(1,2)	(1,3)	(1,4)	(2,2)
Diagonal	(1,3)	(1,2)	(1,3)	(1,4)	(2,2)
AIC	(2,4)	(1,3)	(2,4)	(2,4)	(2,2)
BIC	(1,3)	(1,2)	(1,2)	(1,2)	(1,3)
Draper	(1,3)	(1,2)	(1,3)	(1,3)	(2,2)
CS (MAP)	(1,1)	(1,2)	(1,1)	(1,1)	(1,1)
CS (ML)	(1,1)	(1,2)	(1,1)	(1,1)	(1,1)
ICL	(1,1)	(1,2)	(1,1)	(1,2)	(1,2)

a: $N = 200$

Model Selection Criterion	Trial Number				
	1	2	3	4	5
Laplace	(1,3)	(2,3)	(2,3)	(1,2)	(2,2)
Block	(1,3)	(2,3)	(2,3)	(1,2)	(1,3)
Diagonal	(1,3)	(2,3)	(2,3)	(1,2)	(1,3)
AIC	(2,3)	(2,3)	(2,4)	(1,2)	(2,3)
BIC	(1,2)	(1,3)	(1,4)	(1,2)	(1,3)
Draper	(1,2)	(1,5)	(2,3)	(1,2)	(1,3)
CS (MAP)	(1,1)	(1,2)	(1,2)	(1,2)	(1,2)
CS (ML)	(1,1)	(1,2)	(1,2)	(1,2)	(1,2)
ICL	(1,1)	(1,3)	(1,1)	(1,2)	(1,2)

b: $N = 400$

Table 4.4: Number of latent states selected as optimal by 9 different scoring functions for datasets of size N=200 and 400 generated from 10 different models with n=8 binary observable variables and two latent variables each with 3 states. The figures in brackets indicate the number of states of the two variables Z_1 and Z_2. It should be remembered that the model is invariant to the relabelling of Z_1 and Z_2 so (a, b) is equivalent to (b, a).

4.9 Conclusions

Of those scoring functions which attempt to approximate $P(D \mid \boldsymbol{m})$ it is probably safe to say that we expect to get the best results from the Laplace scoring function as we have seen that when working in the softmax basis this provides a reasonably good approximation to this marginal likelihood. We have seen that the Block and Diagonal approximations, probably unsurprisingly, give very similar results to the Laplace method. Unfortunately these methods are computationally expensive, particularly the Laplace method. Of the other methods based upon approximations to $P(D \mid \boldsymbol{m})$ we have seen that the CS approximation has a tendency to woefully under-estimate $P(D \mid \boldsymbol{m})$ which in turn leads it to select models with very few (if any) latent states. The BIC approximation has a lesser but still clearly apparent tendency to under-estimate $P(D \mid \boldsymbol{m})$ and hence to select more parsimonious models than the Laplace approximation. Draper's suggested amendment to the BIC approximation does seem to go some way towards correcting this under-estimation and although it does not always select the same model as the Laplace approximation it should be remembered that although less accurate this criterion is considerably easier and faster to compute. The AIC approximation which is similar in form to the BIC approximation, as Schwarz noted [66], leads to the selection of higher dimensional models than the BIC criterion, and we have seen here that on occasion this can lead it to select far larger models than the other criteria. Finally, the ICL criterion. This criterion attempts to select a model using a different approach from the other methods studied here. Not surprisingly this leads to results which differ from the other criteria. Without the capacity to visualise the datasets studied here it is difficult

to decide whether the ICL is justified in selecting the models it does. However there is one possible indication of problems with this method in that on a few occasions it selected models which were surprisingly large or surprisingly small.

4.10 Adult Intensive Care Unit (ICU) Data

4.10.1 Introduction

This dataset is taken from Hosmer and Lemeshow [35] and consists of a sample of 200 subjects who were part of a study on survival of patients following admission to an adult intensive care unit. Table 4.5 shows the outcome and binary observable variables in this dataset. The aim of the study was to develop a model to predict the chances of survival of future patients. Here we will discard information about the classification of individuals in terms of their 'Vital Status' (alive or dead) and apply our model selection criteria to choose the best of a number of test models. The aim here is to see if the model selection criteria will lead us to a model in which there are two latent classes and further to see if the resulting classes correspond to the results of the previous study.

4.10.2 Analysis

The test model which most of the scoring functions identified to be the best was that with a single latent variable with 2 states. The only scoring functions to identify models other than this were the AIC scoring function which chose a model with 5 latent states and the CS scoring function which selected the model with no latent state. Since the simulation study had indicated that AIC has

a tendency to overestimate the number of latent states while CS has a marked tendency to underestimate it seemed sensible to conclude that this dataset did indeed contain two sub-populations. The technique used here to learn about the structure of the data is unsupervised so although it has successfully identified two distinct populations we cannot be sure how it is discriminating between them. Since these data were originally collected with the aim of predicting the survival of ICU patients it is certainly desirable that the discrimination between these two groups is equivalent to 'Vital Status'. To see if it might be reasonable to assume that the naive Bayesian network was discriminating between these two groups by their 'Vital Status' the results from the naive Bayesian network were compared with three supervised learning techniques which were taught to discriminate between the two 'Vital Status' groups.

Naive Bayesian Classifier

In the naive Bayesian classifier we first calculate the quantities

$$\begin{aligned} p_j &= P(Z=j) &&= \frac{N_j}{N} \\ \theta_{i|j} &= P(Y_i=1 \mid Z=j) &&= \frac{N_{ij}}{N_j} \end{aligned}$$

where N_{ij} is the number of times we observe $(Y_1 = 1, Z = j)$, N_j is the number of times we observe $Z = j$ and N is the size of the dataset. We then classify each of the observations $X_1, \ldots, X_N$ according to

$$\text{Class Membership}\,(X_i) = \arg\max_j p_j \prod_k \theta_{k|j}.$$

Adapted Naive Bayesian Classifier

Since the two groups in this dataset are of quite different sizes, 160 to 40, the

naive Bayesian classifier is unlikely to do a good job of discriminating the two groups. To compensate for this we can slightly adapt the classification rule of the naive Bayesian classifier to be to classify $X_1, \ldots, X_N$ according to

$$\text{Class Membership}\,(X_i) = \arg\max_j \prod_k \theta_{k|j}.$$

This adaptation is in effect ignoring the fact that the two groups in the dataset are of different sizes.

Logistic Regression Classifier

Logistic regression was applied to the dataset and $X_1, \ldots, X_N$ were then classified according to

$$\text{Class Membership}\,(X_i) = \begin{cases} 1 & \text{if } p\,(X_i) > 0.5 \\ 0 & \text{if } p\,(X_i) \leq 0.5 \end{cases}$$

where $p\,(X_i)$ gives the probability of $X_i = 1$ under the model resulting from the logistic regression.

Results

Results are given in Tables 4.6a-d and indicate that the resulting model provides a reasonable means for distinguishing those patients who would die. The overall misclassification rate of the naive Bayesian network is second only to that given by the Logistic regression classifier, but it could perhaps be argued that the naive Bayesian network model would be preferable as the misclassification rate of those patients who died is lower. It is not surprising that all four methods had difficulties distinguishing the group who did not survive as it is difficult to predict low-prevalence classes, see for example Titterington *et al.*, 1981 [71]. Of the four

methods the only two that were in any way able to correctly classify the dead group were the amended Bayesian classifier and the naive Bayesian network. Of the two the amended Bayesian classifier was slightly better, but at the expense of much worse performance in classifying the surviving patients.

Overall, considering the naive Bayesian network is an unclassified learning technique it has performed remarkably well.

Variable	Name	Codes
1	Identification Code	ID Number
2	Vital Status	0 = Lived 1 = Died
3	Sex	0 = Male 1 = Female
4	Service at ICU admission	0 = Medical 1 = Surgical
5	Cancer part of present problem	0 = No 1 = Yes
6	History of chronic renal failure	0 = No 1 = Yes
7	Infection probable at ICU admission	0 = No 1 = Yes
8	CPR prior to ICU admission	0 = No 1 = Yes
9	Previous admission to ICU within 6 months	0 = No 1 = Yes
10	Type of admission	0 = Elective 1 = Emergency
11	Long bone, multiple, neck, single area, or hip fracture	0 = No 1 = Yes
12	PO2 from initial blood gases	0 = >60 1 = <60
13	PH from initial blood gases	0 = $\geq$7.25 1 = <7.25
14	PCO2 from initial blood gases	0 = $\leq$45 1 = >45
15	Bicarbonate from Initial blood gases	0 = $\geq$18.0 1 = <18.0
16	Creatinine from initial blood gases	0 = $\leq$2.0 1 = >2.0

Table 4.5: Code sheet for ICU data.

Predicted Outcome		Actual Outcome: Survives (160)	Actual Outcome: Dies (40)
Survives	(189)	151	38
Dies	(11)	9	2
		5.6%	95.0%
		23.5%	

a: Naive Bayesian Classifier

Predicted Outcome		Actual Outcome: Survives (160)	Actual Outcome: Dies (40)
Survives	(148)	125	23
Dies	(52)	35	17
		21.9%	57.5%
		29.0%	

b: Amended Naive Bayesian Classifier

Predicted Outcome		Actual Outcome: Survives (160)	Actual Outcome: Dies (40)
Survives	(189)	157	32
Dies	(11)	3	8
		1.9%	80.0%
		17.5%	

c: Logistic Regression Classifier

Predicted Outcome		Actual Outcome: Survives (160)	Actual Outcome: Dies (40)
Survives	(166)	140	26
Dies	(34)	20	14
		12.5%	65.0%
		23.0%	

d: Naive Bayesian Network Classifier

Table 4.6: Comparison of the classification of the ICU data given by 4 different methods. Figures within the table compare the true classifications to the classifications the 4 methods determined. Figures below the tables give the misclassification rates both separately for those patients who lived and died, and the total misclassification rate.

Chapter 5

Reversible Jump MCMC

5.1 Introduction

The Reversible Jump Markov chain Monte Carlo algorithm was proposed by Green in his 1995 paper [31]. Previously MCMC approaches had been limited to problems in which the joint distribution is of fixed dimension, and hence were not applicable to model selection procedures in which one of the unknowns is the dimension of the parameter space. He proposed a new framework for the construction of reversible Markov chain samplers that are able to jump between parameter subspaces of different dimensionalities.

Green considered the case in which we have a countable family of move types, indexed by $m = 1, 2, \ldots$, and we are trying to sample from the posterior distribution of our state variable ψ which is denoted by $\pi(d\psi)$. If the current state is ψ a move of type m and a destination ψ' are proposed with joint distribution given by the measure $q_m(\psi, d\psi')$ which is essentially arbitrary. This

proposed move is then accepted with probability

$$\alpha_m(\psi, \psi') = \min\left\{1, \frac{\pi(d\psi')\, q_m(\psi', d\psi)}{\pi(d\psi)\, q_m(\psi, d\psi')}\right\}. \tag{5.1}$$

When we are considering move types which do not alter the dimension of our parameter space this procedure reduces to the familiar Metropolis-Hastings algorithm. However, Green looks at the more general case in which ψ' lies in a higher dimensional space than ψ. He implements this move by drawing a vector of continuous random variables u, which are independent of ψ, and setting ψ' to be some invertible deterministic function of ψ and u. In this situation the acceptance probability in (5.1) reduces to

$$\min\left\{1, \frac{p(\psi' \mid y)\, r_m(\psi')}{p(\psi \mid y)\, r_m(\psi)\, q(u)} \left|\frac{\partial \psi'}{\partial(\psi, u)}\right|\right\}, \tag{5.2}$$

where y is our observed data, r_m is the probability of attempting move type m when in state ψ, and $q(u)$ is the density function of u. A similar argument also holds for ψ' in a lower dimensional space than ψ, though in this case we consider ψ to be an invertible function of ψ' and u, and rather than drawing u at random its value is determined uniquely by the value of ψ. In this case the acceptance probability is given by

$$\min\left\{1, \frac{p(\psi' \mid y)\, r_m(\psi')\, q(u)}{p(\psi \mid y)\, r_m(\psi)} \left|\frac{\partial(\psi', u)}{\partial \psi}\right|\right\}. \tag{5.3}$$

5.2 Application to Naive Bayesian Networks

In a naive Bayesian network we have a single hidden node Z with k possible states and n observed nodes $Y_1, \ldots, Y_n$ which here are taken to be binary although that need not necessarily be the case. The parameters of the network are $\boldsymbol{p} = \{p_1, \ldots, p_k\}$ and $\boldsymbol{\theta_m} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k\}$, where $\boldsymbol{\theta}_j = \{\theta_{1|j}, \ldots, \theta_{n|j}\}$ and $\theta_{i|j} = P(Y_i = 1 \mid Z_i = j)$. Hence the mixture model we are working with is given by

$$\boldsymbol{y}_l \sim \sum_{j=1}^{k} p_j f(\boldsymbol{y}_l \mid \boldsymbol{\theta}_j), \tag{5.4}$$

where

$$f(\boldsymbol{y}_l \mid \boldsymbol{\theta}_j) = \theta_{1|j}^{y_{l1}} (1 - \theta_{1|j})^{(1-y_{l1})} \ldots \theta_{n|j}^{y_{ln}} (1 - \theta_{n|j})^{(1-y_{ln})}.$$

In the present context a more convenient interpretation of this model is to think of Z as a latent 'allocation variable', which is independently drawn from the distribution

$$P(Z_l = j) = p_j, \tag{5.5}$$

and that given our values of z_l the observations are drawn from that subpopulation. Hence

$$\boldsymbol{y}_l \mid z_l \sim f(\boldsymbol{y}_l \mid \boldsymbol{\theta}_{z_l}). \tag{5.6}$$

We can recover (5.4) from (5.5) and (5.6) by 'integrating out' z. In our Bayesian framework we regard our unknowns as realisations from appropriate prior distributions. We use

$$\boldsymbol{p} \sim Di(\delta_1, \ldots, \delta_1)$$

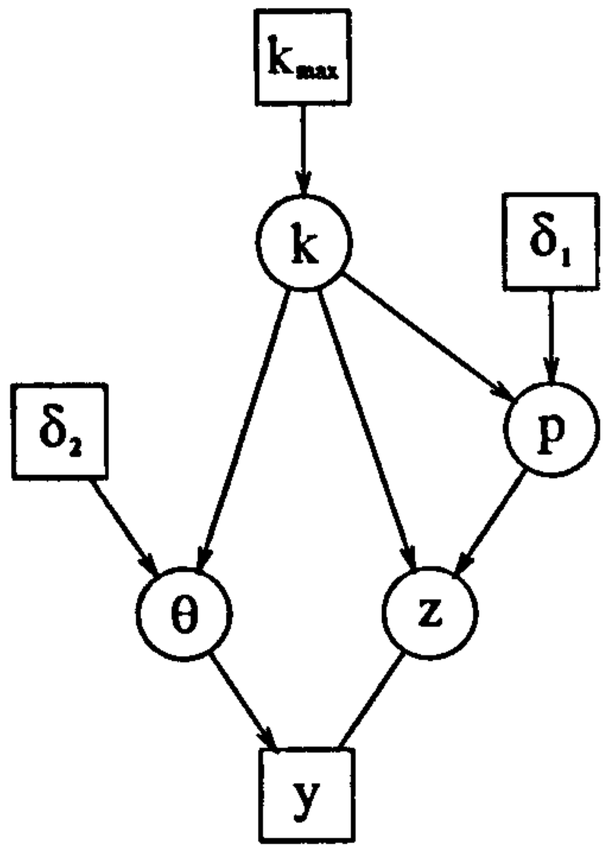


Figure 5.1: DAG specifying the mixture model implemented here.

$$\theta_{i|j} \sim B(\delta_2, \delta_2) \qquad i = 1, \ldots, n \;\; ; \;\; j = 1, \ldots, k,$$

and

$$P(k = j) = \frac{1}{k_{max}} \qquad j = 1, \ldots, k_{max},$$

where k_{max} is some predetermined constant and $\delta_1 > 0$, $\delta_2 > 0$. Hence our joint distribution is given by

$$\begin{aligned} P(\delta_1, \delta_2, k, \boldsymbol{p}, \boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{y}) \;\; = \;\; & P(\delta_1) P(\delta_2) P(k) P(\boldsymbol{p} \mid k, \delta_1) \\ & \times P(\boldsymbol{\theta} \mid k, \delta_2) P(\boldsymbol{z} \mid \boldsymbol{p}, k) P(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{z}) \,. \end{aligned} \tag{5.7}$$

This joint distribution is displayed as a directed acyclic graph (DAG) in Figure 5.1.

5.2.1 The Reversible Jump MCMC Algorithm for Mixture Distributions

Following the method of Richardson and Green [59] [60] we consider an MCMC scheme in which there are 5 possible move types:

(a) updating the weights $\boldsymbol{p}$;

(b) updating the parameters $\boldsymbol{\theta}$;

(c) updating the allocation $\boldsymbol{z}$;

(d) splitting a component into two or combining two components into one;

(e) the birth or death of an empty component.

A complete pass through all 5 steps is considered to be a *sweep* of the algorithm. The first three moves are straightforward and do not alter the dimension of our parameter space; we determine the new values for these unknowns by sampling values from their posterior distributions, which are given by respectively

$$\begin{aligned} \boldsymbol{p} \mid \cdots &\sim Di\left(\delta_1 + N_1, \ldots, \delta_1 + N_k\right), \\ \theta_{i|j} \mid \cdots &\sim B\left(\delta_2 + N_{ij1}, \delta_2 + N_{ij0}\right) \end{aligned}$$

and

$$P\left(z_l = j \mid \cdots\right) \propto p_j f\left(y_l \mid \theta_j\right),$$

where N_{ijk} is the number of times we observe $y_i = k$ and $z = j$, and N_j is the number of times we observe $z = j$.

The last two steps are more complicated, involving a change in the dimension of our parameter space. In the split/combine move we make a random choice

between a split or combine move with probabilities

$$P(\quad \text{attempt a split move} \quad) = b_k$$
$$P(\quad \text{attempt a combine move} \quad) = d_k = 1 - b_k.$$

We define $b_0 = 1$, $d_0 = 0$, $b_{k_{max}} = 0$, $d_{k_{max}} = 1$ and $b_i = d_i = 0.5$ otherwise, where k_{max} is the maximum value allowed for k.

For a combine move, using j_1 and j_2 to denote the two mixture components being combined and j_* to denote the new mixture component, we propose

$$\begin{aligned} p_{j_*} &= p_{j_1} + p_{j_2} \\ \theta_{i|j_*} &= \frac{p_{j_1}\theta_{i|j_1} + p_{j_2}\theta_{i|j_2}}{p_{j_1} + p_{j_2}}. \end{aligned}$$

We also produce an updated allocation variable, z', by simply assigning any observations placed in population j_1 or j_2 to j_*.

This definition of our combine move now determines the split move. Clearly in increasing k by 1 we are increasing the number of parameters in the model by $n+1$, and to determine their values we generate $n+1$ supplementary independent variables $u_1, u_{2,1}, u_{2,2}, \ldots, u_{2,n}$ with distributions

$$u_1 \sim B(1,1) \quad \text{and} \quad u_{2,i} \sim B(1,1), \quad i = 1, \ldots, n.$$

Our split moves are then defined by

$$\begin{aligned} p_{j_1} = u_1 p_{j_*} \quad &, \quad p_{j_2} = (1 - u_1)\, p_{j_*} \\ \theta_{i|j_1} = \tfrac{u_{2,i}}{u_1}\theta_{i|j_*} \quad &, \quad \theta_{i|j_2} = \tfrac{1-u_{2,i}}{1-u_1}\theta_{i|j_*}. \end{aligned}$$

We also produce an updated allocation variable, $\boldsymbol{z}'$, in a manner analogous to the update undertaken in step (c) of our MCMC sweep. In each case that $z_i = j_*$ we assign its new value according to

$$P\left(z_i' = j_1\right) \propto p_{j_1} f\left(y_i \mid \boldsymbol{\theta}_j\right)$$
$$P\left(z_i' = j_2\right) \propto p_{j_2} f\left(y_i \mid \boldsymbol{\theta}_j\right).$$

Since the $\theta_{i|j_1}$ and $\theta_{i|j_2}$ involve a ratio of two independent and identically distributed (iid) random variables there will be instances in which the value of one or more of the $\theta_{i|j_1}$ and $\theta_{i|j_2}$ is greater than 1; when this happens we abandon our split proposal and move on to the next stage of the algorithm. The acceptance probabilities for our split and combine moves are given by $\min\left(1, A_S\right)$ and $\min\left(1, A_C\right)$ respectively, where

$$\begin{aligned} A_S \;=\; & \frac{P\left(\boldsymbol{y} \mid \boldsymbol{z}', \boldsymbol{\theta}'\right)}{P\left(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\theta}\right)} \times \frac{P\left(k+1\right)}{P\left(k\right)} \times \frac{\Gamma\left(\left(k+1\right)\delta_1\right)}{\Gamma\left(\delta_1\right)\Gamma\left(k\delta_1\right)} \frac{p_{j_1}^{\delta_1-1+l_1} p_{j_2}^{\delta_1-1+l_2}}{p_{j_*}^{\delta_1-1+l_1+l_2}} \\ & \times \prod_{i=1}^{n} \frac{\Gamma\left(2\delta_2\right)}{\Gamma\left(\delta_2\right)^2} \frac{\theta_{i|j_1}^{\delta_2-1}\left(1-\theta_{i|j_1}\right)^{\delta_2-1} \theta_{i|j_2}^{\delta_2-1}\left(1-\theta_{i|j_2}\right)^{\delta_2-1}}{\theta_{i|j_*}^{\delta_2-1}\left(1-\theta_{i|j_*}\right)^{\delta_2-1}} \\ & \times \frac{d_{k+1}}{b_k P_{\text{alloc}}} \left(g_{1,1}\left(u_1\right) \prod_{i=1}^{n} g_{1,1}\left(u_{2,i}\right) \right)^{-1} \\ & \times p_{j_*} \prod_{i=1}^{n} \frac{\theta_{i|j_*}}{u_1\left(1-u_1\right)}, \end{aligned} \tag{5.8}$$

and

$$\begin{aligned} A_C \;=\; & \frac{P\left(\boldsymbol{y} \mid \boldsymbol{z}', \boldsymbol{\theta}'\right)}{P\left(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\theta}\right)} \times \frac{P\left(k-1\right)}{P\left(k\right)} \times \frac{\Gamma\left(\delta_1\right)\Gamma\left(\left(k-1\right)\delta_1\right)}{\Gamma\left(k\delta_1\right)} \frac{p_{j_*}^{\delta_1-1+l_1+l_2}}{p_{j_1}^{\delta_1-1+l_1} p_{j_2}^{\delta_1-1+l_2}} \\ & \times \prod_{i=1}^{n} \frac{\Gamma\left(\delta_2\right)^2}{\Gamma\left(2\delta_2\right)} \frac{\theta_{i|j_*}^{\delta_2-1}\left(1-\theta_{i|j_*}\right)^{\delta_2-1}}{\theta_{i|j_1}^{\delta_2-1}\left(1-\theta_{i|j_1}\right)^{\delta_2-1} \theta_{i|j_2}^{\delta_2-1}\left(1-\theta_{i|j_2}\right)^{\delta_2-1}} \end{aligned}$$

$$\times \frac{b_{k-1} P_{\text{alloc}}}{d_k} \left(g_{1,1}(u_1) \prod_{i=1}^{n} g_{1,1}(u_{2,i}) \right)$$
$$\times \left(p_{j_*} \prod_{i=1}^{n} \frac{\theta_{i|j_*}}{u_1 (1-u_1)} \right)^{-1}, \tag{5.9}$$

in which l_1 and l_2 are the numbers of observations assigned to j_1 and j_2 respectively, P_{alloc} is the probability that that particular allocation is made, $g_{p,q}$ denotes the $B(p,q)$ density, and for the combine move the values of u_1 and $u_{2,1}, \ldots, u_{2,n}$ are given by

$$u_1 = \frac{p_{j_1}}{p_{j_*}} \quad \text{and} \quad u_{2,i} = u_1 \frac{\theta_{i|j_1}}{\theta_{i|j_*}} \quad i = 1, \ldots, n.$$

The first two lines of A_S and A_C correspond to the ratio $\frac{p(\psi'|y)}{p(\psi|y)}$, the third line corresponds to the proposal ratio $\frac{r_m(\psi')}{r_m(\psi) q(u)}$ and the final line corresponds to the Jacobian of the transformation between the two parameter spaces. The derivation of A_S and A_C is explained in greater detail in Appendix D.

In the final stage we choose between the birth and death of an empty component, using

$$P(\text{ attempt a birth move }) = b_k$$
$$P(\text{ attempt a death move }) = d_k = 1 - b_k,$$

with b_k and d_k defined as before. When proposing a birth move we draw our new parameters using

$$p_{j_*} \sim B(1,k)$$
$$\theta_{i|j_*} \sim B(\delta_2, \delta_2) \quad i = 1, \ldots, n.$$

We then have to re-scale the existing weights $p_1, \ldots, p_k$ by multiplying them by $(1 - p_{j_*})$ in order that our new weights will sum to 1. No other change is made; in particular the allocation variable $\boldsymbol{z}$ is left unaltered. For a death move we pick an empty component at random, delete it and re-scale the remaining weights accordingly. If no empty mixture component exists the death move is rejected immediately, and we move on to the next stage of the sweep. The acceptance probabilities for the birth and death moves are simplified by the fact that we sample new values for the $\theta_{i|j}$'s from their prior distribution, and are given by $\min(1, A_B)$ and $\min(1, A_D)$ respectively, where

$$\begin{aligned} A_B &= \frac{P(k+1)}{P(k)} \times \frac{\Gamma((k+1)\delta_1)}{\Gamma(k\delta)\Gamma(\delta_1)} p_{j_*}^{\delta_1 - 1} (1 - p_{j_*})^{N + k(\delta_1 - 1)} \\ &\quad \times \frac{d_{k+1}}{b_k} \times \frac{1}{g_{1,k}(p_{j_*})} (1 - w_{j_*})^{k-1} \\ &= \frac{P(k+1)}{P(k)} \times \frac{\Gamma((k+1)\delta_1)}{\Gamma(k\delta_1)\Gamma(\delta_1)} p_{j_*}^{\delta_1 - 1} (1 - p_{j_*})^{N + k(\delta_1 - 1)} \\ &\quad \times \frac{d_{k+1}}{b_k} \times \frac{1}{k} \end{aligned} \tag{5.10}$$

and

$$\begin{aligned} A_D &= \frac{P(k-1)}{P(k)} \times \frac{\Gamma(k - 1\delta_1)\Gamma(\delta_1)}{\Gamma((k)\delta_1)} \times \frac{1}{p_{j_*}^{\delta_1 - 1} (1 - p_{j_*})^{N + k(\delta_1 - 1)}} \\ &\quad \times \frac{b_{k-1}}{d_k} \times (k-1). \end{aligned} \tag{5.11}$$

Again, a more detailed explanation of the derivation of these quantities may be found in Appendix D.

5.3 Sensitivity of Results to Prior Assumptions

It is important to consider how the assumptions we make about our prior distributions can affect the outcome of this methodology. Specifically, in the context of deciding which model provides the best explanation of our data, we are interested in how our priors can affect the posterior distribution of k.

A dataset of 200 observations was generated from a network with 3 latent states and 4 observed variables in which the network parameters were drawn from a uniform distribution. Reversible jump MCMC was then applied to this artificial dataset, involving a burn in of 100000 sweeps followed by a run of 100000 sweeps. Figure 5.2 shows how varying the value of the parameter δ_2, while keeping δ_1 fixed, has affected the posterior distribution of k. It clearly shows that this effect can be considerable, and it therefore follows that it will be difficult to select a value of δ_2 in order to be only weakly informative. However, we would argue that a sensible choice for δ_2 would be 1. Choosing δ_2 to be much smaller or greater than this restricts the values of the parameters θ_{ij}, smaller δ_2 tending to keep the θ_{ij} close to 0 or 1 and larger δ_2 tending to keep them close to 0.5. This restriction means not only that our mixture model might not fit the data so well, but also that the components of our mixture tend to be more similar to each other than they might otherwise be and hence there will be a preference for models with fewer components. Allowing the θ_{ij} freer range can be expected to result in a better fit, and will hopefully lead to better estimation of the number of mixture components required. This choice of δ_2 also has the advantage of greatly simplifying the calculation of the acceptance probabilities for our dimension-altering moves and will speed up the reversible jump MCMC algorithm. It is

also interesting that this value of δ_2 gives the highest level of mixing across k in the example considered here, which in turn can be expected to improve the 'within-k' mixing.

Figure 5.3 shows the effect, on the posterior for k, of altering δ_1 while keeping δ_2 fixed. Perhaps unsurprisingly its effect is even greater than that attributable to δ_2. Again we would argue that $\delta_1 = 1$ is a sensible choice; smaller values of δ_1 will tend to encourage situations in which we have either a single mixture component or many mixture components all with small weights, while larger values of δ_1 will tend to lead to mixture models in which the components are equally weighted, in which case we may well need more mixture components than we might otherwise have done in order to give a good fit to our data. Again this choice has the advantage of reducing the computational complexity of the algorithm and maximising the degree of mixing across k, which in turn can be expected to improve the rate of within k mixing.

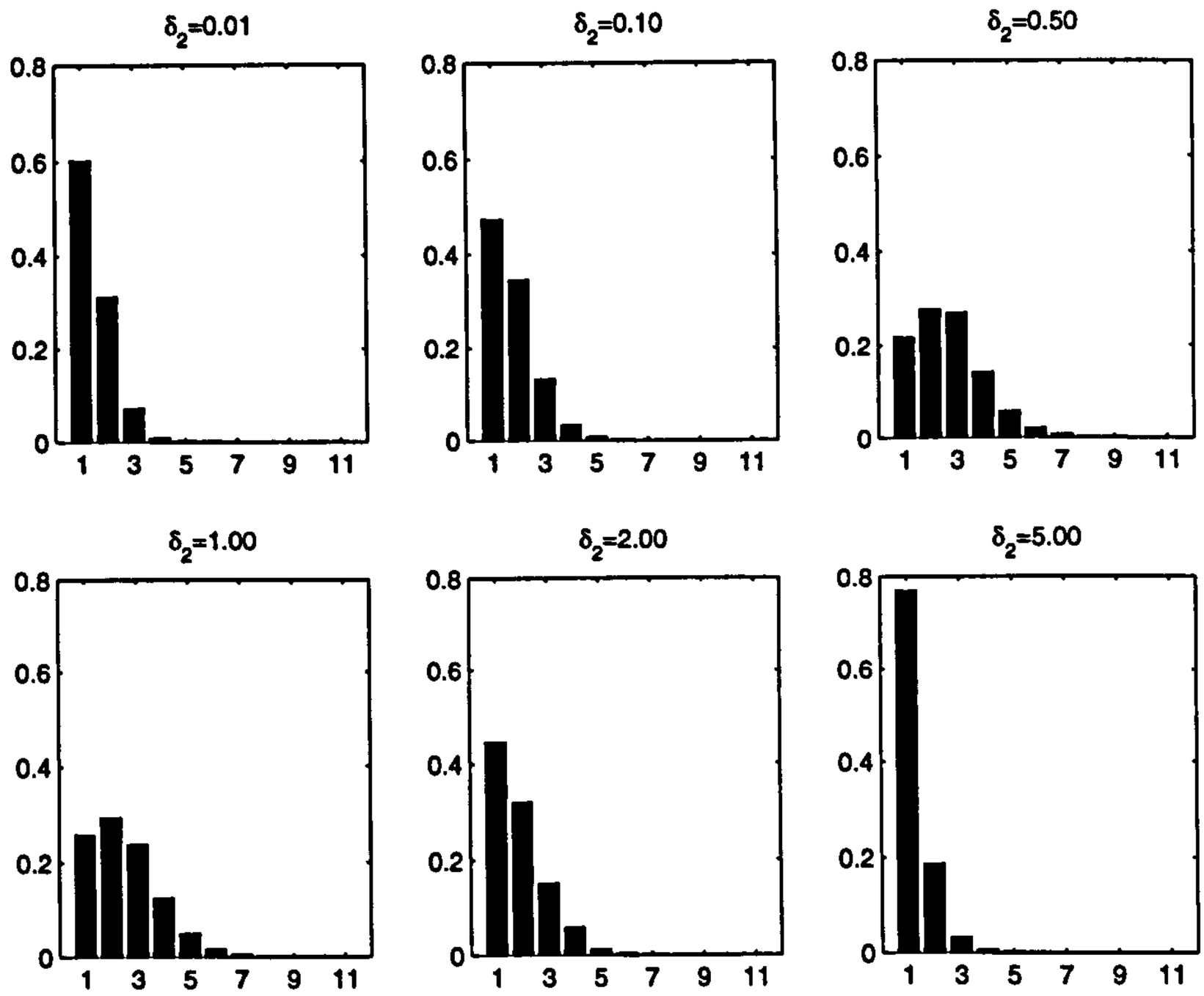


Figure 5.2: The relationship between the posterior distribution of k and the value of the parameter δ_2. Here the value of δ_1 has been kept fixed at 0.5.

	type of move				
δ_2	birth	death	split	combine	overall
0.01	1.1	4.5	0.0	0.0	0.9
0.10	1.4	4.0	0.5	1.0	1.3
0.50	2.1	5.1	3.4	3.3	3.3
1.00	1.9	5.5	4.1	4.4	3.7
2.00	1.4	5.4	3.1	6.2	3.3
5.00	1.0	8.6	1.0	6.0	1.7

Table 5.1: Percentage acceptance rates for the four dimension altering moves for different values of δ_2. Here δ_1 is kept fixed at 0.5.

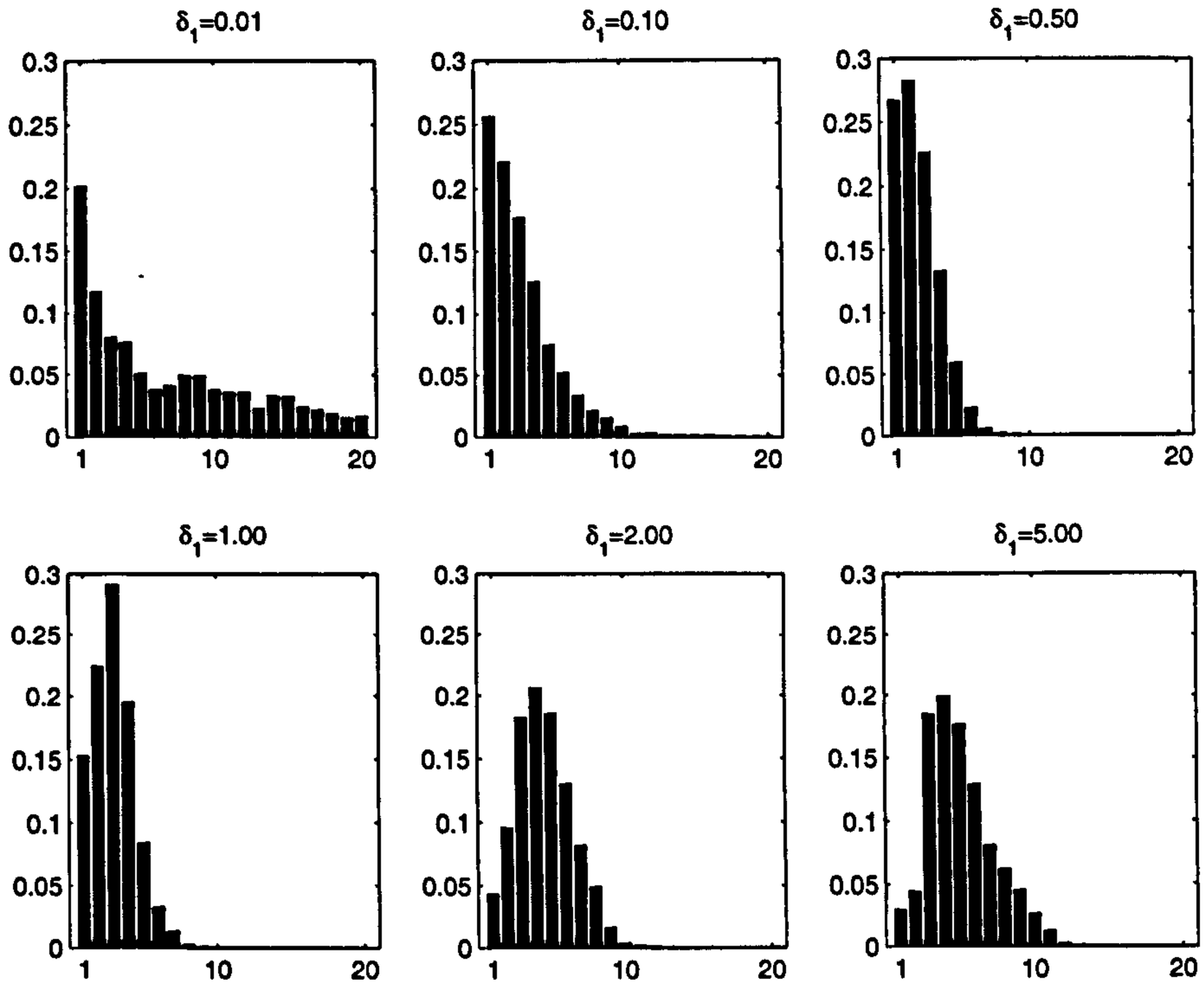


Figure 5.3: The relationship between the posterior distribution of k and the value of the parameter δ_1. Here the value of δ_2 has been kept fixed at 0.5.

	type of move				
δ_1	birth	death	split	combine	overall
0.01	1.3	1.9	0.8	0.9	1.2
0.10	2.1	4.7	2.3	2.9	2.8
0.50	2.0	5.1	3.4	3.7	3.3
1.00	1.3	3.5	3.7	3.3	2.9
2.00	0.2	1.0	3.8	3.5	2.1
5.00	0.0	0.0	3.2	3.3	1.7

Table 5.2: Percentage acceptance rates for the four dimension altering moves for different values of δ_1. Here δ_2 is kept fixed at 0.5.

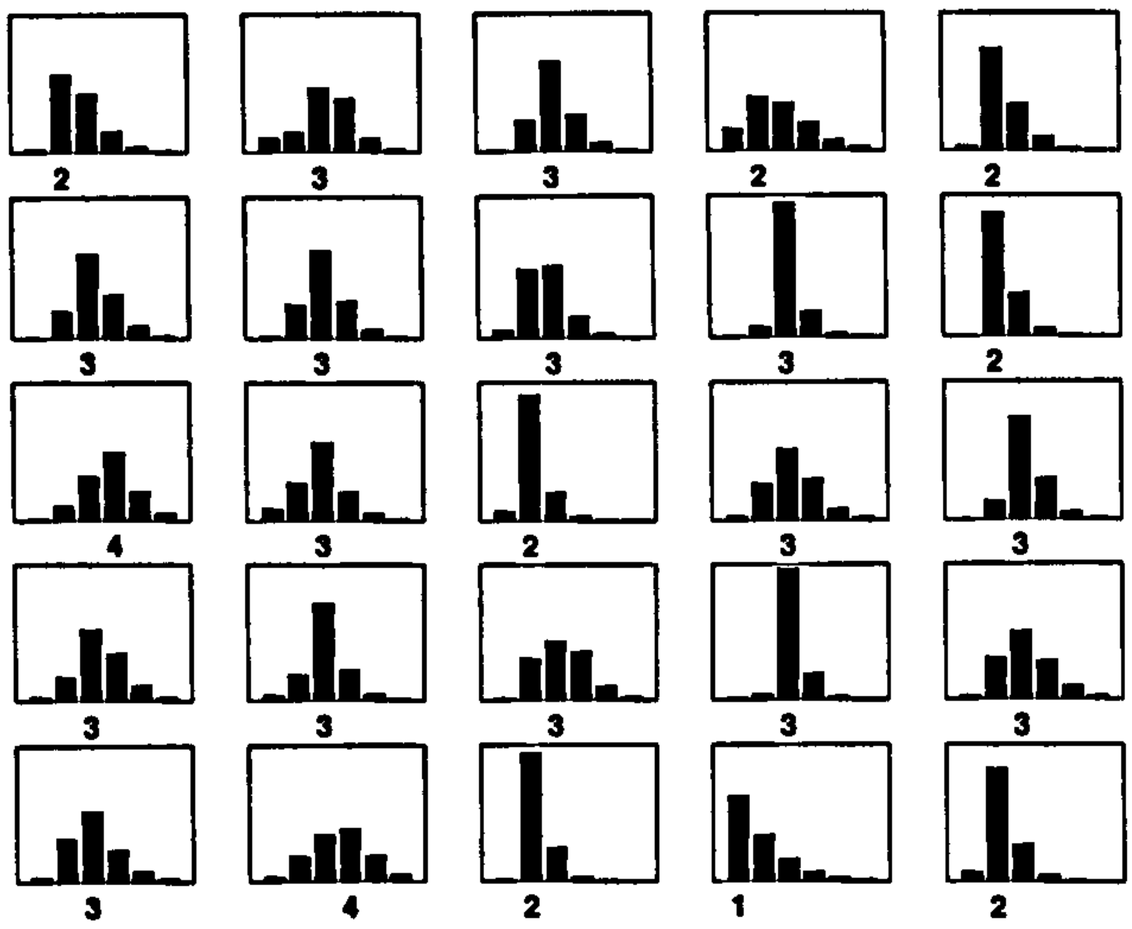


Figure 5.4: Estimates of the posterior distribution from the Reversible Jump MCMC algorithm when applied to datasets of size $N = 100$ generated from 25 different naive Bayesian networks with 4 latent states and 8 binary observable variables. The figures in bold show the most likely number of latent states for each example.

We can see how this choice of δ_1 and δ_2 affects the performance of the Reversible Jump algorithm in Figures 5.4, 5.5 and 5.6 which show the estimates of the posterior distribution of k for networks with 4 latent states and 8 binary observable datasets when we have a dataset of sizes $N = 100$, 500 and 1000; twenty-five examples of each size of dataset are shown and the results are summarised in Table 5.3. For a dataset of size 100 we would not expect there to be sufficient support for a model with 4 latent states, so unsurprisingly the results from the reversible jump algorithm tend to indicate that there are less than 4 latent states present in the data; if anything it is perhaps surprising that the algorithm indicates 4 latent states in even 2 of the datasets examined. As the dataset size increases we would expect it to become easier for the algorithm to determine the correct number of latent states, and indeed for datasets of size

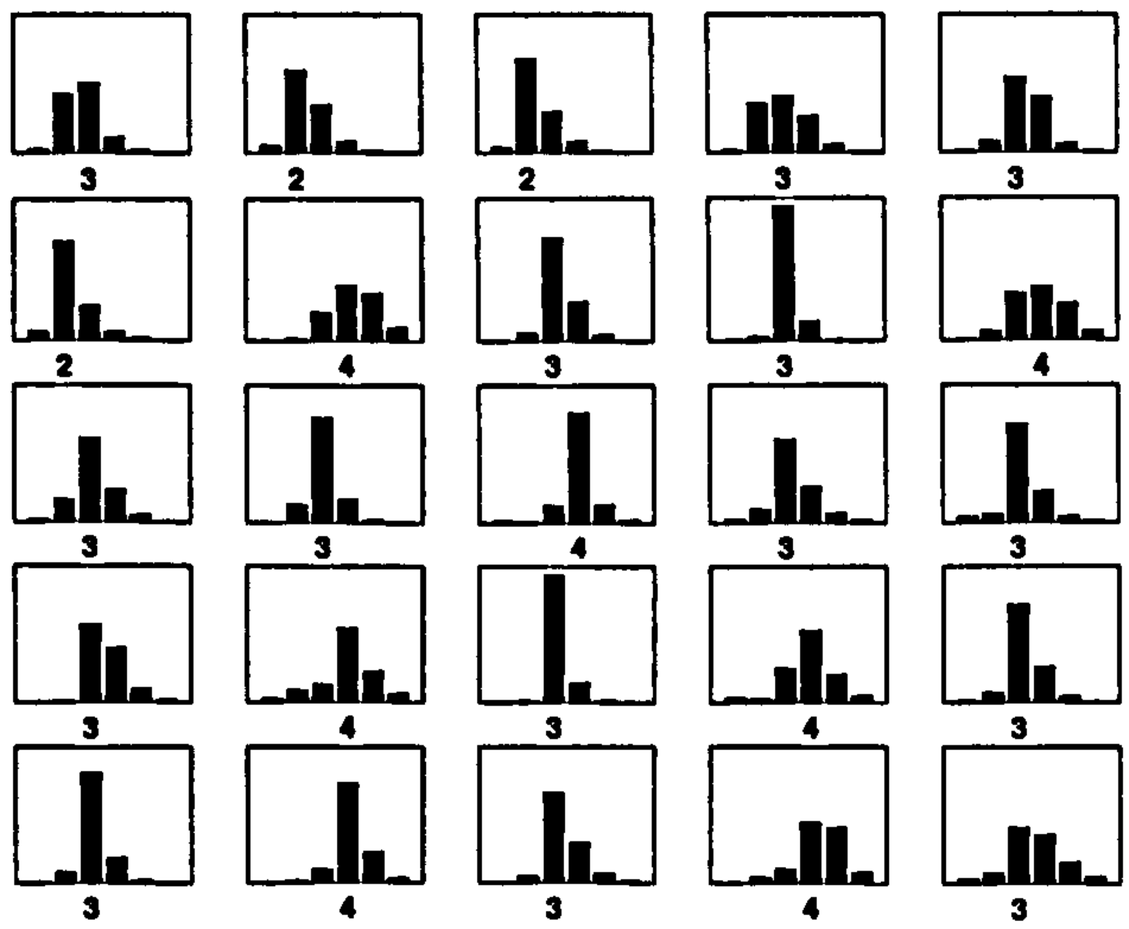


Figure 5.5: Estimates of the posterior distribution from the Reversible Jump MCMC algorithm when applied to datasets of size $N = 100$ generated from 25 different naive Bayesian networks with 4 latent states and 8 binary observable variables. The figures in bold show the most likely number of latent states for each example.

1000 the algorithm identifies the correct number of states in just over half of the examples looked at. In the remaining 12 cases it identifies 3 rather than 4 latent classes; it would seem that in these examples even this dataset is not large enough to support 4 latent classes.

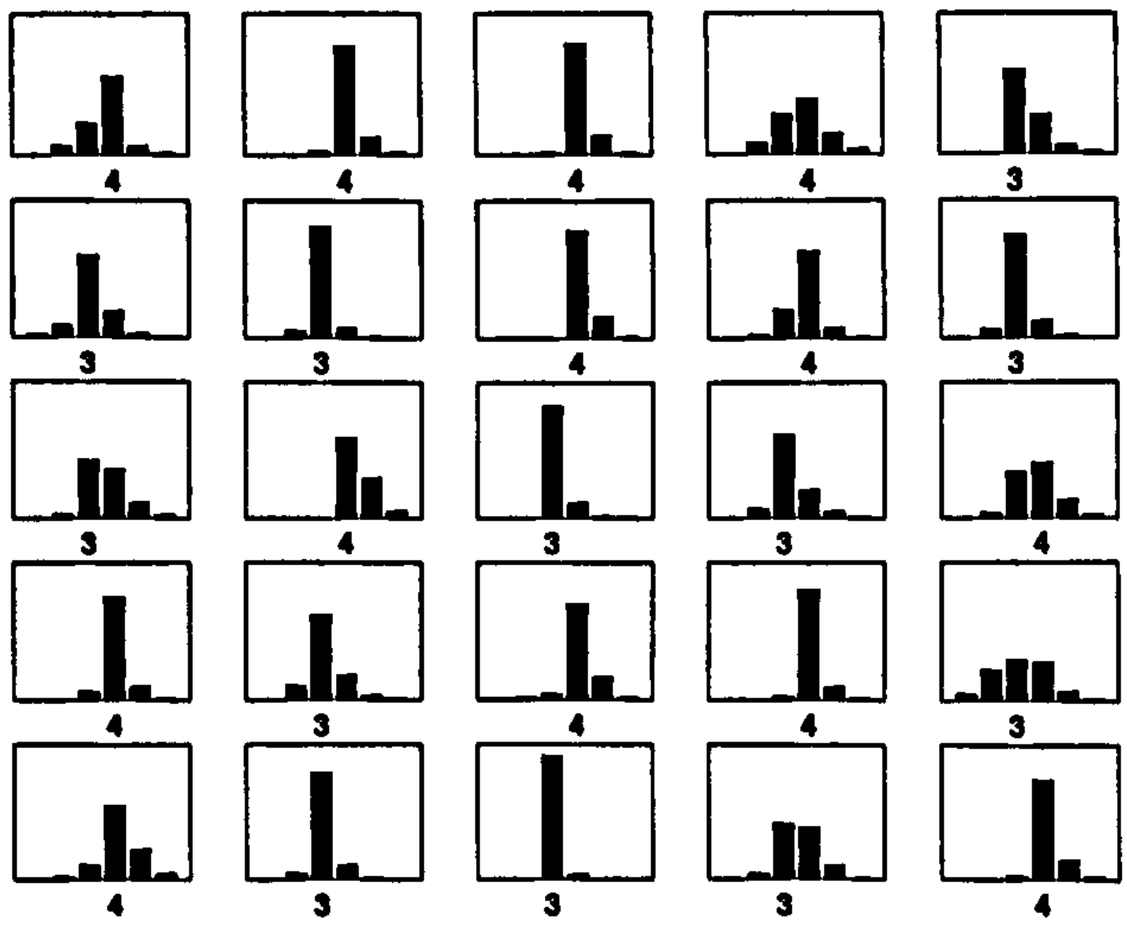


Figure 5.6: Estimates of the posterior distribution from the Reversible Jump MCMC algorithm when applied to datasets of size $N = 100$ generated from 25 different naive Bayesian networks with 4 latent states and 8 binary observable variables. The figures in bold show the most likely number of latent states for each example.

Number of Latent States	Size of dataset		
	100	500	1000
1	1	0	0
2	7	3	0
3	15	15	12
4	2	7	13

Table 5.3: The number of latent states chosen as most likely by the Reversible Jump MCMC algorithm when applied to datasets generated from 25 different models with 4 latent states and 8 binary observable variables. The datasets used are of size 100, 500 and 1000.

5.3.1 Post-Processing of MCMC Output

Here we consider the reversible jump MCMC algorithm with $\delta_1 = \delta_2 = 1$ and $P(k) = \frac{1}{k_{max}}$ for all k, for which the quantities which must be calculated in order to determine our acceptance probabilities for our birth, death, split and combine moves are simplified to

$$A_S = \frac{P(\boldsymbol{y} \mid \boldsymbol{z}', \boldsymbol{\theta}')}{P(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\theta})} \times k \frac{p_{j_1}^{l_1} p_{j_2}^{l_2}}{p_{j_*}^{l_1+l_2}} \times \frac{d_{k+1}}{b_k P_{\text{alloc}}} \times p_{j_*} \prod_{i=1}^{n} \frac{\theta_{i|j_*}}{u_1(1-u_1)}, \tag{5.12}$$

$$A_C = \frac{P(\boldsymbol{y} \mid \boldsymbol{z}', \boldsymbol{\theta}')}{P(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\theta})} \times k \frac{p_{j_*}^{l_1+l_2}}{p_{j_1}^{l_1} p_{j_2}^{l_2}} \times \frac{b_{k-1} P_{\text{alloc}}}{d_k} \times \left(p_{j_*} \prod_{i=1}^{n} \frac{\theta_{i|j_*}}{u_1(1-u_1)} \right)^{-1} \tag{5.13}$$

$$A_B = (1-p_{j_*})^N \times \frac{d_{k+1}}{b_k} \tag{5.14}$$

$$A_D = (1-p_{j_*})^{-N} \times \frac{b_{k-1}}{d_k} \tag{5.15}$$

A new dataset of size 200 was generated from a network with 4 observable variables and 3 latent states, and the reversible jump MCMC algorithm was applied to the resulting dataset using a burn-in period of 100000 sweeps followed by 100000 sweeps in which values for the unknown parameters were collected. As can be seen in Figure 5.7, the algorithm correctly identified, through the posterior mode, the number of latent classes present in this dataset.

In practice when using reversible jump MCMC it will not be enough to extract only information about the number of subclasses present. We will also be interested in producing information about the parameters of the models within each parameter space. This inevitably leads to issues of identifiability and labelling, which can be difficult to overcome. In order to interpret the output of

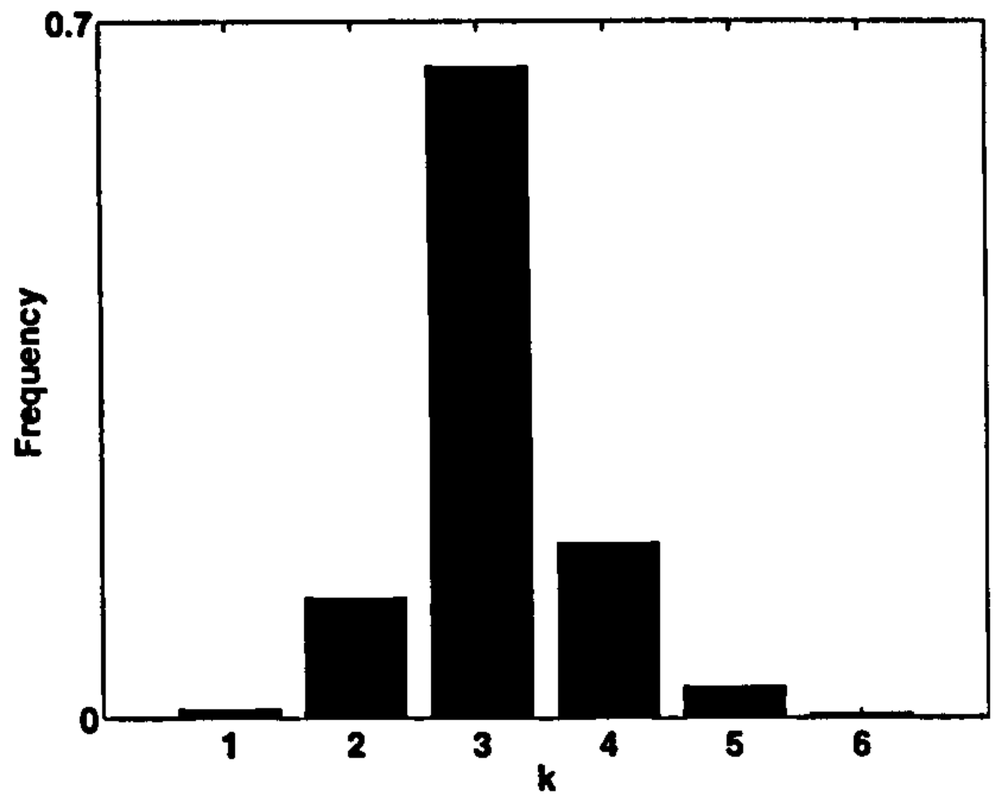


Figure 5.7: Estimate of the posterior distribution of k. The reversible jump MCMC algorithm correctly identifies the number of latent classes in the dataset as 3.

the MCMC algorithm we need to impose some sort of labelling on the components of our mixture. For the example here one obvious way of imposing this labelling is to order the components of the mixture according to their weights; we can then estimate the parameters of our network by taking the simple ergodic average of our sample, and we term these the 'naive' parameter estimates. Figure 5.8 shows estimates of the posterior distribution of the weights of the mixture components in the case $k = 3$, and compares the MAP estimates from the MCMC output to the true values. Clearly many of the posterior distributions for the network parameters are bimodal; this can be the result of there being two competing three component mixture models, but a more likely explanation is that 'label switching' has occurred. This means that we are assigning some of our realisations to the wrong mixture component. This problem is particularly marked in this example because two of the mixture components have very similar weights. The result of this is that in a good many instances we are assigning observations drawn from the component with the larger weight to the component with the smaller

weight and vice versa. The effect of this is to introduce bias into our estimates of these weights, tending to decrease the smaller weight and increase the larger one. This label switching also means that our estimates for the parameters of each mixture component are very inaccurate. As can be seen in Figure 5.10 the estimates of the parameters associated with the first two components are quite different from the true values. However, it can also be seen that the modes of the estimated posterior distributions lie close to the true values, the implication being that if we could correctly assign our observations to the components of the mixture we could produce fairly accurate estimates. Another problem that is illustrated by this figure is that the posterior distributions of the parameters associated with low weight components tend to be very diffuse. The low weighting of these components means that we gain less information about the true values of these parameters from the MCMC output as a result of fewer observations being attributed to these components.

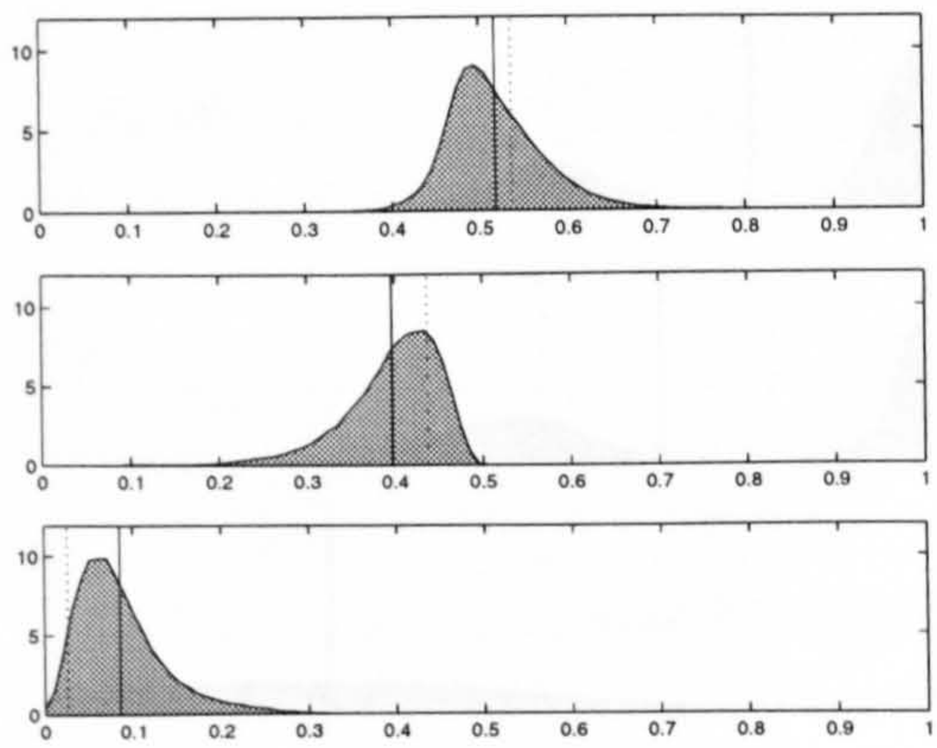


Figure 5.8: Estimates of the posterior distributions of the weights of the three mixture components. For convenience we label the components 1, 2, 3, with 1 being the component with the largest weight, 3 the one with the smallest weight. The solid vertical line indicates the average value of the parameter from the 100000 MCMC runs and the dotted vertical line indicates the MAP estimates of the parameters.

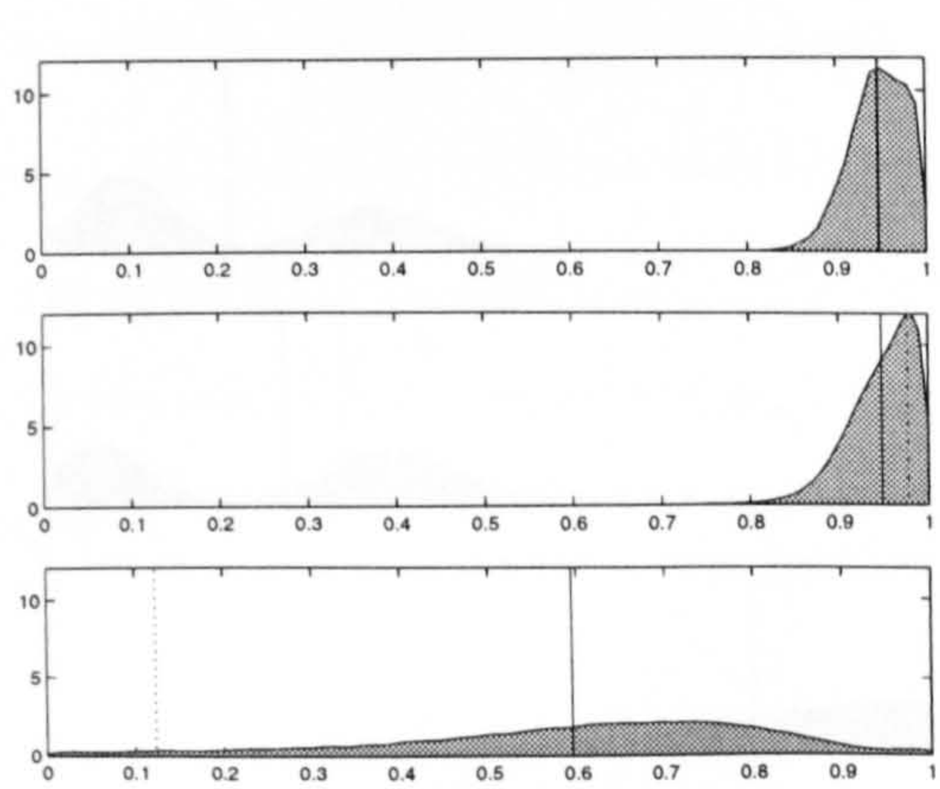


Figure 5.9: Estimates of the posterior distributions of the probability of the first node being 1 given membership of each of the three components. The graphs correspond to membership of components 1, 2 and 3 going from top to bottom. The solid vertical line indicates the average value of the parameter from the 100000 MCMC runs and the dotted vertical line indicates the MAP estimates of the parameters.

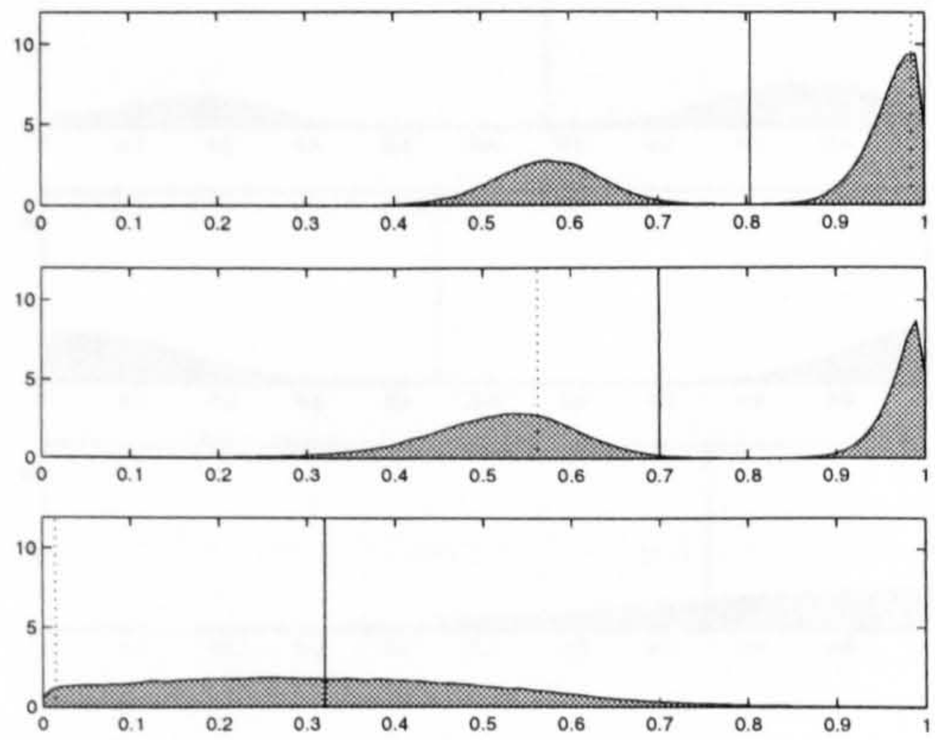


Figure 5.10: Estimates of the posterior distributions of the probability of the second node being 1 given membership of each of the three components. The graphs correspond to membership of components 1, 2 and 3 going from top to bottom. The solid vertical line indicates the average value of the parameter from the 100000 MCMC runs and the dotted vertical line indicates the MAP estimates of the parameters.

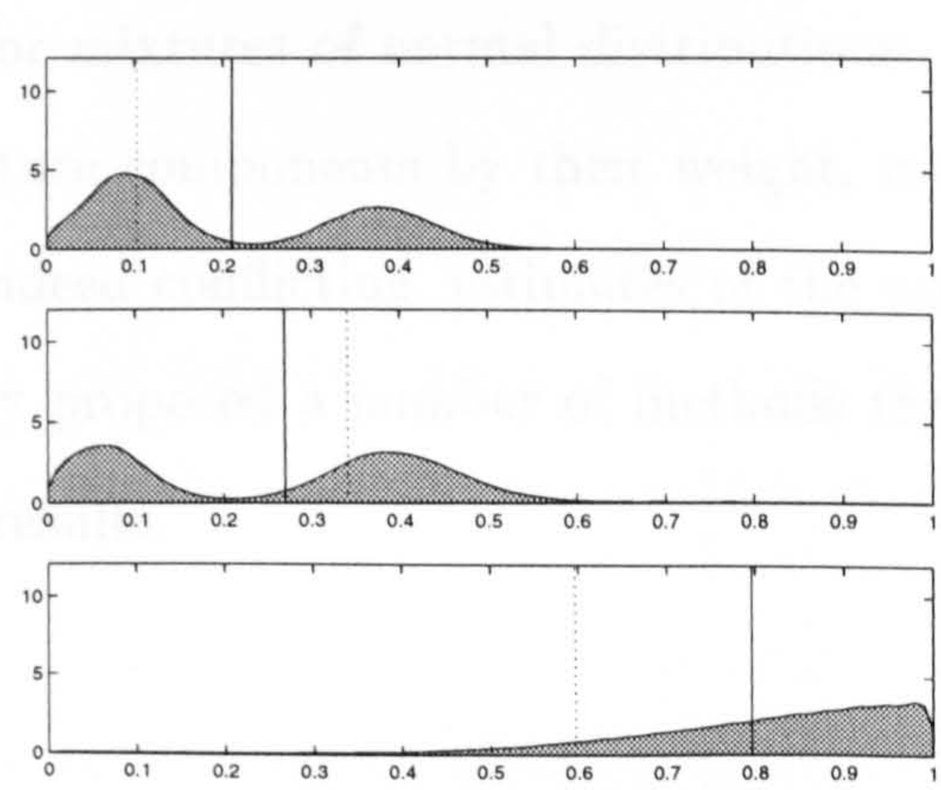


Figure 5.11: Estimates of the posterior distributions of the probability of the third node being 1 given membership of each of the three components. The graphs correspond to membership of components 1, 2 and 3 going from top to bottom. The solid vertical line indicates the average value of the parameter from the 100000 MCMC runs and the dotted vertical line indicates the MAP estimates of the parameters.

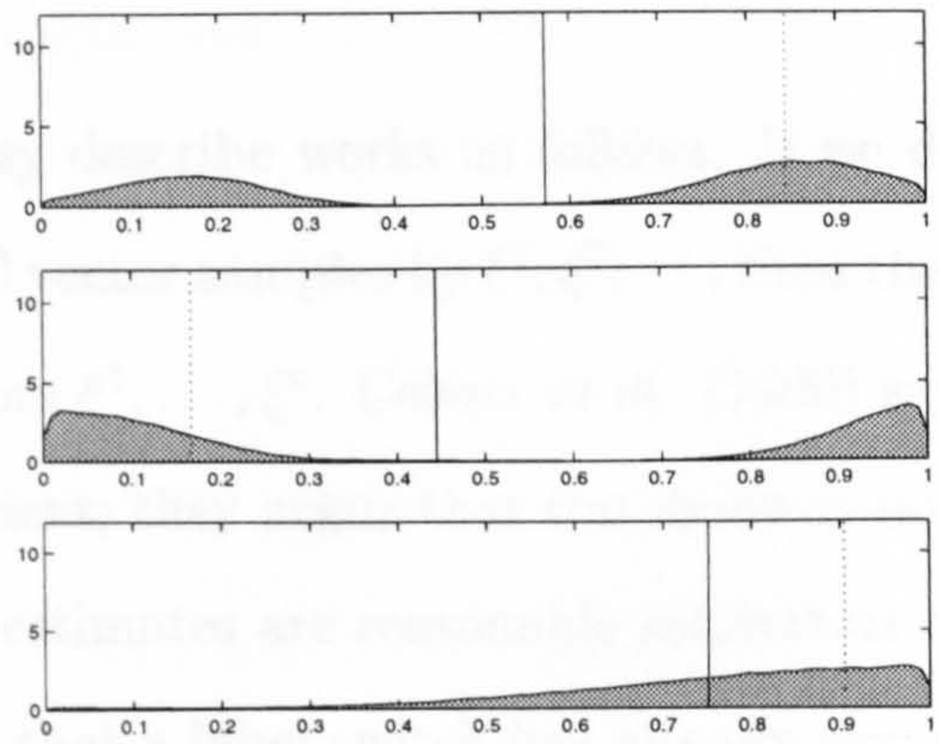


Figure 5.12: Estimates of the posterior distributions of the probability of the fourth node being 1 given membership of each of the three components. The graphs correspond to membership of components 1,2 and 3 going from top to bottom. The solid vertical line indicates the average value of the parameter from the 100000 MCMC runs and the dotted vertical line indicates the MAP estimates of the parameters.

Celeux *et al.* [11] considered the problem of post-processing the output of an MCMC sampler for mixtures of normal distributions. They also found that constraining the mixture components by their weight, mean or variance led to unsatisfactory, and indeed conflicting, estimates of the posterior distribution of the parameters. They proposed a number of methods that they felt could lead to more satisfactory results.

Classifying

Their first proposal was the use of a classification algorithm to allocate the MCMC samples to the different mixture components. The idea is that if each sample is permuted so that they all come from the same mode of the posterior distribution then their mean will form a suitable estimate of the position of that mode from which the position of all other modes can be derived by permutation of the

component labels.

The algorithm they describe works as follows. If we denote the sequence of d-dimensional MCMC vector samples by $\xi^1, \xi^2, \ldots$, then the procedure is initiated using the first v vectors $\xi^1, \ldots, \xi^v$. Celeux *et al.* (1999) say that a value for v of 100 is generally sufficient; they argue that the choice is not vital but must be so large that the initial estimates are reasonable estimators of the posterior means but not large enough that a label switch has already occurred. These m vectors are used to define reference centres for the d parameters in the output,

$$\bar{\xi}_i = \frac{1}{v} \sum_{j=1}^{v} \xi_i^j,$$

along with component-wise variances

$$s_i = \frac{1}{v} \sum_{j=1}^{v} \left(\xi_i^j - \bar{\xi}_i\right)^2.$$

We then set $s_i^{[0]} = s_i$, $i = 1, \ldots, d$. If we write $\bar{\xi}_1^{[0]} = \bar{\xi}$, then the other $(k! - 1)$ centres $\bar{\xi}_2^{[0]}, \ldots, \bar{\xi}_{k!}^{[0]}$ can be deduced from $\bar{\xi}_1^{[0]}$ by permutation of the component labels. After this initialisation stage the r^{th} iteration of the clustering algorithm proceeds as follows.

1. Allocate ξ^{v+r} to the cluster j^* which minimises the normalised squared distance

$$\|\xi^{v+r} - \bar{\xi}_j^{[r-1]}\|^2 = \sum_{i=1}^{d} \frac{\left(\xi_i^{v+r} - \bar{\xi}_{ij}^{[r-1]}\right)^2}{s_i^{[r-1]}},$$

   where $\bar{\xi}_{ij}^{[r-1]}$ is the i^{th} coordinate of $\bar{\xi}_j^{[r-1]}$.

2. If $j^* \neq 1$ we permute the coordinates of ξ^{v+r} to get $j^* = 1$.

3. Update the $k!$ centres and the d normalising coefficients as follows

   (a) Compute

$$\bar{\xi}_1^{[r]} = \frac{v+r-1}{v+r}\bar{\xi}_1^{[r-1]} + \frac{1}{v+r}\xi^{v+r}.$$

   (b) Derive the other $(k! - 1)$ centres by permuting the component labels.

   (c) Update the component-wise variances using

$$\begin{aligned} s_i^{[r]} &= \frac{v+r-1}{v+r}s_i^{[r-1]} + \frac{v+r-1}{v+r}\left(\bar{\xi}_{i1}^{[r-1]} - \bar{\xi}_{i1}^{[r]}\right)^2 \\ &\quad + \frac{1}{v+r}\left(\xi_i^{v+r} - \bar{\xi}_{i1}^{[r]}\right)^2. \end{aligned}$$

In this way the mode of reference corresponds to $j = 1$ at each iteration.

A Loss Function for the Parameters

Celeux *et al.* suggest a second approach to the estimation of the mixture parameters. This approach is motivated by thinking of the components of the mixture distribution as points in a fixed dimension point process. So in the example here a mixture model would be represented by k points in (p, θ) space. This suggests that an appropriate loss function could be formulated by considering ways of measuring the 'distance' between two point configurations. The approach they suggest is based upon the Baddeley Δ metric. This metric was initially suggested by Baddeley [2] as a method for measuring the distance between two binary images and is based upon the distance from every pixel to the closest foreground pixel. Celeux *et al.*'s adaptation of this metric begins by selecting a set of points $t_1, \ldots, t_n$ in the same space as the mixture components $\xi = (\xi_j)$. They define $d(t_i, \xi)$ to be the distance from t_i to the closest of the ξ_j,

$j = 1, \ldots, k$, which they then use to define the loss function

$$L\left(\xi, \hat{\xi}\right) = \sum_{i=1}^{n} \left(d\left(t_i, \xi\right) - d\left(t_i, \hat{\xi}\right)\right)^2.$$

Thus, for each of the t_i there is a contribution to the loss function if the distance to the nearest ξ_j is not the same as to the nearest ξ_j. Celeux *et al.* make two points about the choice of the t_i, first that ideally we should have $L\left(\xi, \hat{\xi}\right) = 0$ if and only if $\xi = \hat{\xi}$, and secondly that the loss function should respond appropriately to changes in the two point configurations. They argue that both these points can be answered by ensuring that the t_i are both sufficiently numerous and lie in high posterior density regions of the ξ_j's space. They aim to satisfy both conditions by allocating the first half of their simulations from the posterior to the allocation of the t_i. They do this by randomly selecting one of the components of each realisation as a t_i. Following the two step procedure proposed by Rue [64], the remaining realisations are then used to estimate the corresponding quantities $\gamma_i = \mathbb{E}_{\xi|x}\left[d\left(t_i, \xi\right)\right]$. The estimates $\hat{\gamma}_i$ are simply found by taking the ergodic average of the samples of the quantity γ_i calculated by our remaining realisations from the posterior distribution. Then, noting that it is possible to write

$$\begin{aligned}
&\mathbb{E}_{\xi|x}\left[\sum_{i=1}^{n}\left(d\left(t_i, \xi\right) - d\left(t_i, \hat{\xi}\right)\right)^2\right] \\
&\quad = \sum_{i=1}^{n}\left(\mathbb{E}_{\xi|x}\left[d\left(t_i, \xi\right)^2\right] - 2d\left(t_i, \xi\right)\mathbb{E}_{\xi|x}\left[d\left(t_i, \xi\right)\right] + d\left(t_i, \xi\right)^2\right)
\end{aligned}$$

and using $\hat{\gamma}_i$ as an approximation for $\mathbb{E}_{\xi|x}\left[d\left(t_i,\xi\right)\right]$, the problem of minimising the loss function becomes that of finding $\hat{\xi}^*$ such that

$$\hat{\xi}^* = \arg\min_{\hat{\xi}} h\left(\hat{\xi}\right) = \arg\min_{\hat{\xi}} \sum_{i=1}^{n} \left(-2\hat{\gamma}_i d\left(t_i,\hat{\xi}\right) + d\left(t_i,\xi\right)^2\right).$$

This minimisation can then be carried out by using simulated annealing to search for the modes of the distribution proportional to $\exp\left(-h\left(\hat{\xi}\right)\right)$.

A Loss Function for the Predictive Distribution

Celeux et al's final suggestion is based upon a more global loss function that measures distributional discrepancies. They suggest the use of the integrated square difference

$$L\left(\xi,\hat{\xi}\right) = \int_{\mathcal{R}} \left(f_\xi\left(y\right) - f_{\hat{\xi}}\left(y\right)\right)^2 dy.$$

The analogue for this for discrete data would be the summed square difference

$$L\left(\xi,\hat{\xi}\right) = \sum_{y\in\mathcal{R}} \left(f_\xi\left(y\right) - f_{\hat{\xi}}\left(y\right)\right)^2,$$

where $\mathcal{R}$ is the set of all possible values of y. This loss function again lends itself to a two step approach. If we assume the integration and summation can be interchanged, the posterior expected loss can be decomposed as

$$\begin{aligned}&\mathbb{E}_{\xi|x}\left[\textstyle\sum_{y\in\mathcal{R}}\left(f_\xi\left(y\right) - f_{\hat{\xi}}\left(y\right)\right)^2\right]\\ &= \textstyle\sum_{y\in\mathcal{R}}\left(\mathbb{E}_{\xi|x}\left[f_\xi\left(y\right)^2\right] - 2f_{\hat{\xi}}\left(y\right)\mathbb{E}_{\xi|x}\left[f_\xi\left(y\right)\right] + f_{\hat{\xi}}\left(y\right)^2\right).\end{aligned}$$

We can then use our realisations from the posterior distribution to estimate the quantities $\delta(y) = \mathbb{E}_{\xi|x}[f_\xi(y)]$, $y \in \mathcal{R}$, by calculating these quantities for each of our realisations and taking the ergodic average. Minimising the posterior loss function then becomes a matter of finding $\hat{\xi}^*$ such that

$$\hat{\xi}^* = \arg\min_{\hat{\xi}} h\left(\hat{\xi}\right) = \arg\min_{\hat{\xi}} \sum_{y \in \mathcal{R}} \left(-2\hat{\delta}(y) f_{\hat{\xi}}(y) + f_{\hat{\xi}}(y)^2\right).$$

Again, we can use simulated annealing to tackle this minimisation problem. One problem that might be encountered in the use of this method is the sheer size of $\mathcal{R}$; if our set of observable variables contains just 10 binary variables then $\mathcal{R}$ contains 1024 items.

Application

Applying these three methods to the Reversible Jump MCMC output produced mixed results. The clustering method performed poorly, giving results identical to the naive method. The problem with this method appeared to result from the use of component-wise variances to normalise the distances between observations and the current cluster centres; variances for the weights tended to be much lower than for other parameters, generally by an order of magnitude or more. This means that more importance is placed on the close matching of weights than of any other parameter, resulting in allocations identical (up to a permutation) to those given by the naive method. The method based on Baddeley's Δ metric also performed poorly. Figure 5.13 compares this method to the naive method. The 72 points on the diagram show how the distances from the estimates from the Δ metric method to the true values compared to the true values, and how

the distance from the true and estimated observation probabilities compared. It shows that the parameter estimates were improved in only 21% of cases, and that in only 2 cases (3%) were the observed probability estimates improved. This method also proved to be computationally expensive, requiring a far greater run time than any of the other methods looked at here. The final method based on the loss function for the predictive distribution proved much better; Figure 5.14 compares this method to the naive method. Perhaps unsurprisingly the estimate for the predictive distribution is improved in 85% of cases. It is also notable that in just over half the cases observed (54%) this method also improved the parameter estimates.

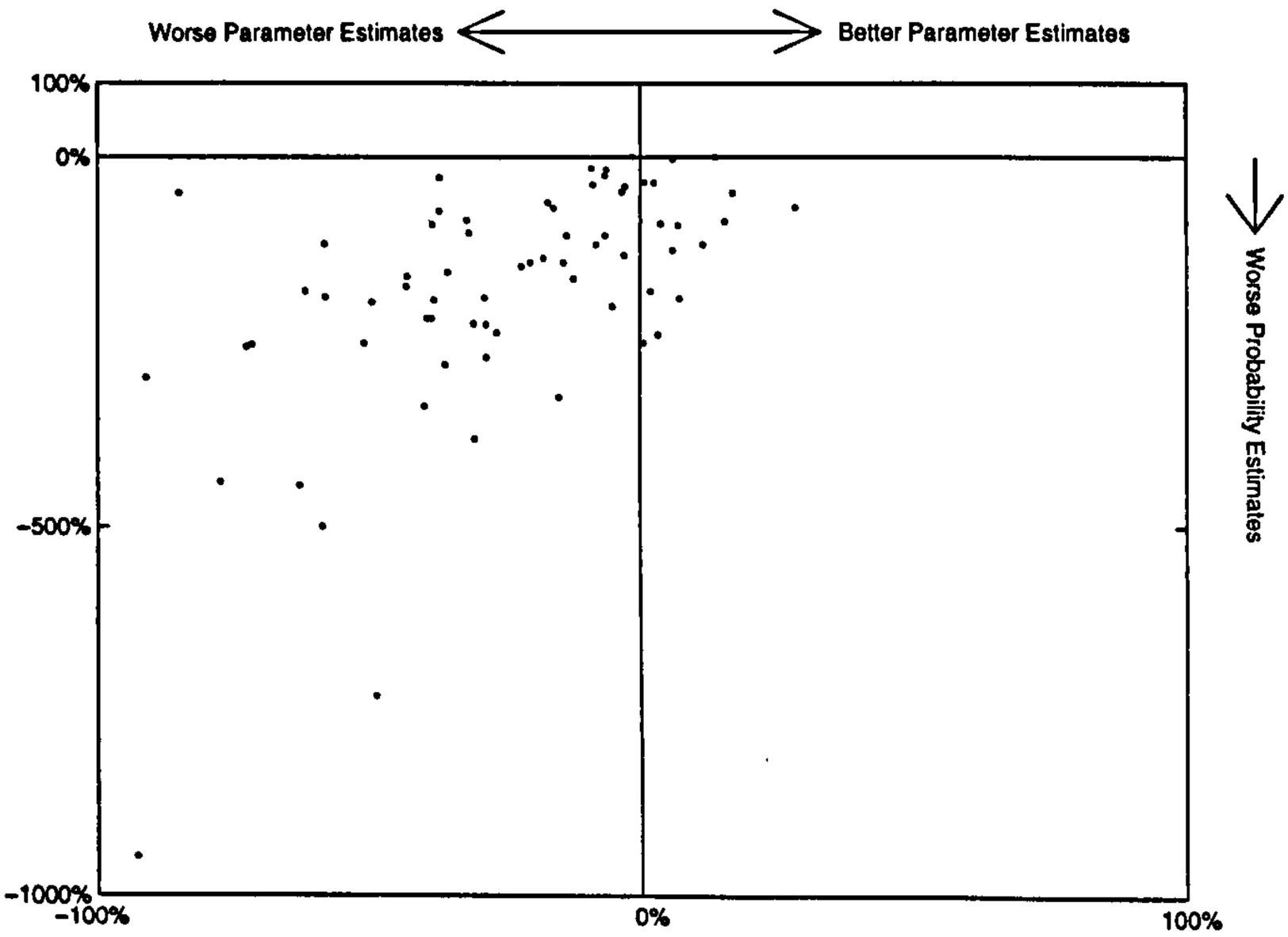


Figure 5.13: Comparison of parameter estimates and distribution estimates given by the method based on Baddeley's Δ metric compared to allocating observations to components based merely on their weights.

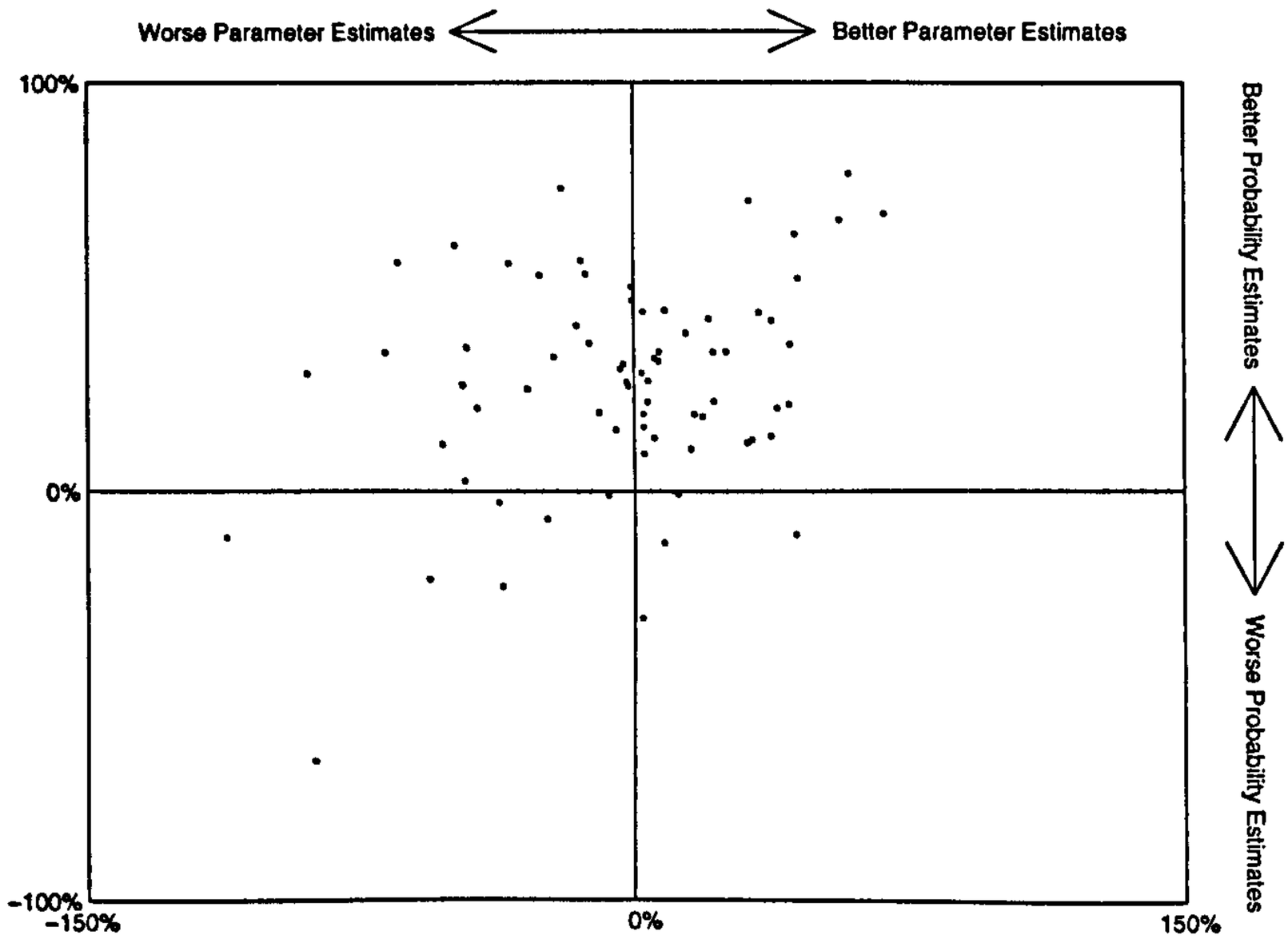


Figure 5.14: Comparison of parameter estimates and distribution estimates given by the method based on the loss function for the predictive distribution compared to allocating observations to components based merely on their weights.

Chapter 6

Discussion and Future Work

In this thesis we have reviewed the current state of the art for a number of different aspects of Bayesian belief networks. Clearly many issues need to be considered only some of which we have looked at.

In Chapter 2 we have surveyed some of the recently and not so recently proposed parameter estimation algorithms and compared their effectiveness in finding the parameters of a naive Bayesian network. Many of these techniques have been proposed with the specific intention of improving on the less desirable properties of that old workhorse, the EM algorithm. Of those methods aimed at improving the rate of convergence we have found that finding the mixing proportions of known mixture components is a problem which is successfully tackled by almost all of the algorithms we considered. However, we have also seen that the usefulness of these algorithms often does not extend beyond this to more complicated problems in which the components of the mixture are also unknown. We have also considered recent attempts to solve the more difficult problem of finding the global maximum of our likelihood surface rather than a

local stationary point. Unfortunately it became apparent that the little progress that has been made in this area has little application to models of the type we are considering.

We have seen in Chapter 3 that although useful tools for determining model identifiability exist that there is still a need for a 'general theorem of identifiability'. In this thesis we have gone some way towards producing a general result but this result only applies to specific types of naive Bayesian networks, which themselves form only a small subset of Bayesian network structures. The techniques we used in this proof do not lend themselves to similar proofs for more complicated models, and a different approach will be necessary if a more generally applicable result is to be derived.

In Chapter 4 we have surveyed some of the currently available methods for model selection and made some attempt to make recommendations about their effectiveness. Given that we are unable to visualise the multidimensional binary data we have been considering in this thesis it is difficult to assess how well the models selected by these criterion really fit the data. However,the study we undertook on the ICU admissions data would seem to indicate that at least some of these methods are doing a reasonable job of selecting an effective model.

Finally in Chapter 5 we looked at a method which combines the twin problems of model selection and parameter estimation. From a Bayesian point of view it is highly desirable that we tackle these problems simultaneously rather than adapting the usual false dichotomy and dealing with these problems as separate entities. Unfortunately we have seen that although this method shows promise as a model selection tool there still remains a great deal of scope for improving the post-processing of the output from the algorithm in order to produce satisfactory

parameter estimates.

Appendix A

Implementation of Parameter Estimation Algorithms

In this appendix we present the algorithms of Chapter 2 as they would be when applied to a naive Bayesian network with a single hidden node, Z, with r_z states and n observable nodes, $Y_1, \ldots, Y_n$, each with $r_i = 2$ states. The parameters of this network are given by

$$\begin{aligned} p_j &= P(Z = j) \\ \theta_{i|j} &= P(Y_i = 1 \mid Z = j) \end{aligned}$$

A.1 EM Algorithm

A.1.1 Mixtures of Known Components

We iterate between the following two steps until convergence is reached.

E Step

Calculate the quantities

$$P_{lj} = P(y_l \mid Z = j)\, j = 1, \ldots, r_z \text{ and } l = 1, \ldots, N$$

M Step

$$p_j = \frac{1}{N} \sum_{l=1}^{N} P_{lj}$$

A.1.2 Mixtures of Unknown Components

E Step

Calculate the quantities

$$P_{lj} = P(y_l \mid Z = j)\, j = 1, \ldots, r_z \text{ and } l = 1, \ldots, N$$

M Step

$$\begin{aligned} p_j &= \frac{1}{N} \sum_{l=1}^{N} P_{lj} \\ \theta_{i|j} &= \frac{1}{N p_j} \sum_{l=1}^{N} y_{li} P_{lj} \end{aligned}$$

A.2 IEM Algorithm

A.2.1 Mixtures of Known Components

Begin with one step of the EM algorithm to determine the quantities

$$P_{lj}^0 = P(y_l \mid Z = j)\, j = 1, \ldots, r_z \text{ and } l = 1, \ldots, N$$

We then iterate between the following two steps until convergence is reached.

E Step

Choose an observation y_l and calculate

$$P_{lj}^t = P(y_l \mid Z = j)\, j = 1, \ldots, r_z$$

M Step

$$p_j^t = \frac{1}{N}\left(N p_j^{t-1} - P_{lj}^{t-1} + P_{lj}^t\right)$$

A.2.2 Mixtures of Unknown Components

Begin with one step of the EM algorithm to determine the quantities

$$P_{lj}^0 = P(y_l \mid Z = j)\, j = 1, \ldots, r_z \text{ and } l = 1, \ldots, N$$

We then iterate between the following two steps until convergence is reached.

E Step

Choose an observation y_l and calculate

$$P_{lj}^t = P(y_l \mid Z = j)\, j = 1, \ldots, r_z$$

M Step

$$\begin{aligned} p_j^t &= \frac{1}{N}\left(N p_j^{t-1} - P_{lj}^{t-1} + P_{lj}^t\right) \\ \theta_{i|j}^t &= \frac{1}{N p_j^t}\left(N p_j^{t-1}\theta_{i|j}^{t-1} - y_{li} P_{lj}^{t-1} + y_{li} P_{lj}^t\right) \end{aligned}$$

A.3 Helmbold's Methods

For an explanation of the derivation of Helmbold et al's algorithms when applied to the proportion vector problem see their 1997 paper, "A comparison of new and old algorithms for a mixture estimation problem" [34]. Here we consider the application of their methods to naive Bayesian networks in which the mixture components are also unknown.

A.3.1 Mixtures of Unknown Components

Recall that for a general network we wish to find to find $\boldsymbol{\theta}_m = \boldsymbol{\theta}_m^{t+1}$ to maximise

$$F'\left(\boldsymbol{\theta}_m, \gamma\right) = \eta \Delta \mathcal{L}\left(\boldsymbol{\theta}_m^t\right)^T \boldsymbol{\theta}_m - d\left(\boldsymbol{\theta}_m, \boldsymbol{\theta}_m^t\right) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \gamma_{ij} \left(\sum_{k=1}^{r_i} \theta_{ijk} - 1 \right),$$

where $d\left(\boldsymbol{\theta}_m, \boldsymbol{\theta}_m^t\right)$ is a distance metric, one of the Euclidean, relative entropy and chi squared metrics given by

$$\begin{aligned} d_{EUC}\left(\boldsymbol{u} \parallel \boldsymbol{v}\right) &= \frac{1}{2} \parallel \boldsymbol{u} - \boldsymbol{v} \parallel_2^2 = \frac{1}{2} \sum_{i=1}^{N} \left(u_i - v_i\right)^2 \\ d_{RE}\left(\boldsymbol{u} \parallel \boldsymbol{v}\right) &= \sum_{i=1}^{N} u_i \ln \frac{u_i}{v_i} \\ d_{\chi^2}\left(\boldsymbol{u} \parallel \boldsymbol{v}\right) &= \frac{1}{2} \sum_{i=1}^{N} \frac{\left(u_i - v_i\right)^2}{v_i}. \end{aligned}$$

For our naive Bayesian network with binary observable variables the problem becomes one of finding $\boldsymbol{\theta}_m = \boldsymbol{\theta}_m^{t+1}$ to maximise

$$F'\left(\boldsymbol{\theta}_m, \gamma\right) = \eta \Delta \mathcal{L}\left(\boldsymbol{\theta}_m^t\right)^T \boldsymbol{\theta}_m - d\left(\boldsymbol{\theta}_m, \boldsymbol{\theta}_m^t\right) + \gamma \left(\sum_{j=1}^{r_z} p_j - 1 \right) + \sum_{j=1}^{r_z} \sum_{i=1}^{n} \delta_{ij} \left(\theta_{ij0} + \theta_{ij1} - 1\right).$$

Here we have altered our notation slightly so that

$$\theta_{ij1} = \theta_{i|j}$$

$$\theta_{ij0} = 1 - \theta_{i|j}$$

$$\frac{\partial F'(\boldsymbol{\theta}_m, \gamma)}{\partial p_k} = \eta \frac{\partial \mathcal{L}(\boldsymbol{\theta}_m^t)}{\partial p_k} - \frac{\partial d(\boldsymbol{\theta}_m^t, \boldsymbol{\theta}_m^{t+1})}{\partial p_k} + \gamma \tag{A.1}$$

$$\frac{\partial F'(\boldsymbol{\theta}_m, \gamma)}{\partial \theta_{hki}} = \eta \frac{\partial \mathcal{L}(\boldsymbol{\theta}_m^t)}{\partial \theta_{hki}} - \frac{\partial d(\boldsymbol{\theta}_m^t, \boldsymbol{\theta}_m^{t+1})}{\partial \theta_{hki}} + \delta_{hk} \tag{A.2}$$

Hence for the Euclidean distance metric we have

$$\frac{\partial F'(\boldsymbol{\theta}_m, \gamma)}{\partial p_k} = \eta \frac{\partial \mathcal{L}(\boldsymbol{\theta}_m^t)}{\partial p_k} - \left(p_k^t - p_k^{t-1}\right) + \gamma \tag{A.3}$$

$$\frac{\partial F'(\boldsymbol{\theta}_m, \gamma)}{\partial \theta_{h|k}} = \eta \frac{\partial \mathcal{L}(\boldsymbol{\theta}_m^t)}{\partial \theta_{h|k}} - \left(\theta_{h|k}^t - \theta_{h|k}^{t-1}\right) \tag{A.4}$$

Putting (A.3) equal to zero, summing over k and applying the constraints $\sum_k p_k^t = \sum_k p_k^{t+1} = 1$ we have

$$\gamma = \eta \sum_{k=1}^{r_z} \frac{\partial \mathcal{L}(\boldsymbol{\theta}_m^t)}{\partial p_k}.$$

Similarly

$$\delta_{ij} = \eta \left(\frac{\partial \mathcal{L}(\boldsymbol{\theta}_m^t)}{\partial \theta_{ij0}} + \frac{\partial \mathcal{L}(\boldsymbol{\theta}_m^t)}{\partial \theta_{ij1}} \right)$$

Hence the GP_η updates are given by

$$\begin{aligned} p_k^{t+1} &= p_k^t + \eta \left(\frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial p_k} - \frac{1}{r_z} \sum_{j=1}^{r_z} \frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial p_j} \right) \\ \theta_{ij1}^{t+1} &= \theta_{ij1}^t + \eta \left(\frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial \theta_{ij1}} - \left(\frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial \theta_{ij0}} + \frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial \theta_{ij1}} \right) \right). \end{aligned}$$

Since

$$\mathcal{L}\left(\theta_m^t\right) = \sum_{l=1}^{N} \sum_{j=1}^{r_z} p_j \prod_{i=1}^{n} \theta_{ij1}^{y_{li}} \theta_{ij0}^{1-y_{li}}$$

and hence

$$\frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial p_k} = \sum_{l=1}^{N} \prod_{i=1}^{n} \theta_{ik1}^{y_{li}} \theta_{ik0}^{1-y_{li}} \tag{A.5}$$

$$\frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial \theta_{ij1}} = \sum_{l=1}^{N} p_j y_{li} \theta_{ij1}^{-1} \prod_{i=1}^{n} \theta_{ij1}^{y_{li}} \theta_{ij0}^{1-y_{li}} \tag{A.6}$$

$$\frac{\partial \mathcal{L}\left(\theta_m^t\right)}{\partial \theta_{ij0}} = \sum_{l=1}^{N} p_j \left(1 - y_{li}\right) \theta_{ij0}^{-1} \prod_{i=1}^{n} \theta_{ij1}^{y_{li}} \theta_{ij0}^{1-y_{li}} \tag{A.7}$$

we can write our updates quite simply as

$$\begin{aligned} p_k^{t+1} &= p_k^t + \eta \left(\sum_{l=1}^{N} \prod_{i=1}^{n} \theta_{ik1}^{y_{li}} \theta_{ik0}^{1-y_{li}} - \frac{1}{r_z} \sum_{j=1}^{r_z} \sum_{l=1}^{N} \prod_{i=1}^{n} \theta_{ij1}^{y_{li}} \theta_{ij0}^{1-y_{li}} \right) \\ &= p_k^t + \eta \left(\sum_{l=1}^{N} \prod_{i=1}^{n} \theta_{i|k}^{y_{li}} \left(1 - \theta_{i|k}\right)^{1-y_{li}} - \frac{1}{r_z} \sum_{j=1}^{r_z} \sum_{l=1}^{N} \prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1 - \theta_{i|j}\right)^{1-y_{li}} \right) \end{aligned} \tag{A.8}$$

$$\begin{aligned} \theta_{ij1}^{t+1} &= \theta_{ij1}^t + \eta \left(\sum_{l=1}^{N} p_j y_{li} \theta_{ij1}^{-1} \prod_{i=1}^{n} \theta_{ij1}^{y_{li}} \theta_{ij0}^{1-y_{li}} \right. \\ &\quad \left. - \frac{1}{2} \left(\sum_{l=1}^{N} p_j y_{li} \theta_{ij1}^{-1} \prod_{i=1}^{n} \theta_{ij1}^{y_{li}} \theta_{ij0}^{1-y_{li}} + \sum_{l=1}^{N} p_j \left(1 - y_{li}\right) \theta_{ij0}^{-1} \prod_{i=1}^{n} \theta_{ij1}^{y_{li}} \theta_{ij0}^{1-y_{li}} \right) \right) \end{aligned}$$

$$= \theta^t_{i|j} + \eta p_j \left(\sum_{l=1}^{N} y_{li} \theta^{-1}_{i|j} \prod_{i=1}^{n} \theta^{y_{li}}_{i|j} \left(1 - \theta_{i|j}\right)^{1-y_{li}} \right.$$
$$\left. -\frac{1}{2} \sum_{l=1}^{N} \left(\left(y_{li} \theta^{-1}_{i|j} + (1 - y_{li}) \left(1 - \theta_{i|j}\right)^{-1} \right) \prod_{i=1}^{n} \theta^{y_{li}}_{i|j} \left(1 - \theta_{i|j}\right)^{1-y_{li}} \right) \right) \tag{A.9}$$

Using the relative entropy metric we can derive the EG_η update in a similar fashion.

$$\frac{\partial d_{RE}\left(\theta^t_m, \theta^{t+1}_m\right)}{\partial p^{t+1}_k} = \log\left(\frac{p^{t+1}_k}{p^t_k}\right) + 1 \tag{A.10}$$

$$\frac{\partial d_{RE}\left(\theta^t_m, \theta^{t+1}_m\right)}{\partial \theta_{hki}} = \log\left(\frac{\theta^{t+1}_{hki}}{\theta^t_{hki}}\right) + 1 \tag{A.11}$$

and hence

$$p^{t+1}_k = p^t_k \exp\left(\eta \frac{\partial \mathcal{L}\left(\theta^t_m\right)}{\partial p^t_k}\right) \Big/ \sum_{j=1}^{r_z} p^t_j \exp\left(\eta \frac{\partial \mathcal{L}\left(\theta^t_m\right)}{\partial p^t_j}\right)$$
$$= p^t_k \exp\left(\eta \sum_{l=1}^{N} \prod_{i=1}^{n} \theta^{y_{li}}_{i|k} \left(1 - \theta_{i|k}\right)^{1-y_{li}}\right) \Big/ \sum_{j=1}^{r_z} p^t_j \exp\left(\eta \sum_{l=1}^{N} \prod_{i=1}^{n} \theta^{y_{li}}_{i|j} \left(1 - \theta_{i|j}\right)^{1-y_{li}}\right) \tag{A.12}$$

$$\theta_{h|k} = \theta_{h|k} \exp\left(\eta \frac{\partial \mathcal{L}\left(\theta^t_m\right)}{\partial \theta^t_{h|k}}\right) \Big/ \left(\theta_{h|k} \exp\left(\eta \sum_{l=1}^{N} p_k y_{li} \theta^{-1}_{h|k} \prod_{i=1}^{n} \theta^{y_{li}}_{h|k} \left(1 - \theta_{h|k}\right)^{1-y_{li}}\right) \right.$$
$$\left. + \left(1 - \theta_{h|k}\right) \exp\left(\eta \sum_{l=1}^{N} p_k \left(1 - y_{li} \left(1 - \theta_{h|k}\right)^{-1} \prod_{i=1}^{n} \theta^{y_{li}}_{h|k} \left(1 - \theta_{h|k}\right)^{1-y_{li}}\right)\right)\right) \tag{A.13}$$

Finally we use the χ^2 metric to derive the EM_η update

$$\frac{\partial d_{\chi^2}\left(\theta_m^t, \theta_m^{t+1}\right)}{\partial p_k^{t+1}} = \frac{p_k^{t+1}}{p_k^t} - 1 \tag{A.14}$$

$$\frac{\partial d_{\chi^2}\left(\theta_m^t, \theta_m^{t+1}\right)}{\partial \theta_{hki}} = \frac{\theta_{hki}^{t+1}}{\theta_{hki}^t} - 1 \tag{A.15}$$

and hence

$$\begin{aligned} p_k^{t+1} &= p_k^t \left(\eta \left(\sum_{l=1}^{N} \prod_{i=1}^{n} \theta_{i|k}^{y_{li}} \left(1-\theta_{i|k}\right)^{1-y_{li}} - \sum_{j=1}^{r_x} p_j \sum_{l=1}^{N} \prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1-\theta_{i|j}\right)^{1-y_{li}} \right) + 1 \right) \\ &= p_k^t \left(\eta \left(\sum_{l=1}^{N} \prod_{i=1}^{n} \theta_{i|k}^{y_{li}} \left(1-\theta_{i|k}\right)^{1-y_{li}} - 1 \right) + 1 \right) \end{aligned} \tag{A.16}$$

$$\begin{aligned} \theta_{h|k}^{t+1} = \theta_{h|k}^t \Bigg(\eta \Bigg(& p_k^t y_{li} \theta_{h|k}^{-1} \prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1-\theta_{i|j}\right)^{1-y_{li}} \\ & - \left(p_k^t y_{li} \prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1-\theta_{i|j}\right)^{1-y_{li}} + p_k^t \left(1-y_{li}\right) \prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1-\theta_{i|j}\right)^{1-y_{li}} \right) \Bigg) + 1 \Bigg) \end{aligned} \tag{A.17}$$

A.4 ACG_ϕ , ACG_θ and ACG_θ^χ Algorithms

The formulae for the gradient needed by the ACG_θ and ACG_θ^χ algorithms are as detailed for the conjugate gradient algorithms; the calculations for the ACG_ϕ algorithm are different due to operating in a transformed parameter space. We have made use of the logit transformation, so our new parameters are

$$\begin{aligned} \zeta_j &= \log \frac{p_j}{p_1} \\ \phi_{i|j} &= \log \frac{\theta i \mid j}{1-\theta_{i|j}} \end{aligned}$$

The first order partial derivatives we require are then

$$\begin{aligned}
\frac{\partial \mathcal{L}(\theta_m)}{\partial \zeta_k} &= \frac{\partial}{\partial \zeta_k} \sum_{l=1}^{N} \log P(y_l \mid \theta_m) \\
&= \sum_{l=1}^{N} \frac{1}{P(y_l \mid \theta_m)} \frac{\partial}{\partial \zeta_k} P(y_l \mid \theta_m) \\
\frac{\partial \mathcal{L}(\theta_m)}{\partial \phi_{h|k}} &= \sum_{l=1}^{N} \frac{1}{P(y_l \mid \theta_m)} \frac{\partial}{\partial \phi_{h|k}} P(y_l \mid \theta_m)
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial}{\partial \zeta_k} P(y_l \mid \theta_m) &= p_k(\phi)(1 - p_k(\phi)) \prod_{i=1}^{n} \theta_{i|k}(\phi)^{y_{li}} \left(1 - \theta_{i|k}(\phi)\right)^{1-y_{li}} \\
&\quad - \sum_{j \neq k} p_j(\phi)\, p_k(\phi) \prod_{i=1}^{n} \theta_{i|j}(\phi)^{y_{li}} \left(1 - \theta_{i|j}(\phi)\right)^{1-y_{li}} \\
\frac{\partial}{\partial \phi_{h|k}} P(y_l \mid \theta_m) &= p_k(\phi)\left(y_{li} - \theta_{h|k}(\phi)\right) \prod_{i=1}^{n} \theta_{i|k}(\phi)^{y_{li}} \left(1 - \theta_{i|k}(\phi)\right)^{1-y_{li}}
\end{aligned}$$

A.5 Conjugate Gradient Algorithms (CG and GCG)

A.5.1 Calculating the Gradient

We need to eliminate p_{r_z} before calculating the derivative of $\mathcal{L}(\theta_m)$ using $p_{r_z} = 1 - p_1 - \ldots - p_{r_z-1}$, hence

$$\mathcal{L}(\theta_m) = \sum_{l=1}^{N} \log P(y_l \mid \theta)$$

where

$$\begin{aligned} P(y_l \mid \theta) &= \sum_{j=1}^{r_z} p_j \prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1 - \theta_{i|j}\right)^{1-y_{li}} \\ &= \sum_{j=1}^{r_z-1} p_j \prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1 - \theta_{i|j}\right)^{1-y_{li}} + \left(1 - \sum_{j=1}^{r_z-1} p_j\right) \prod_{i=1}^{n} \theta_{i|r_z}^{y_{li}} \left(1 - \theta_{i|r_z}\right)^{1-y_{li}} \end{aligned} \tag{A.18}$$

First order partial derivatives are then given by

$$\begin{aligned} \frac{\partial \mathcal{L}(\theta_m)}{\partial p_k} &= \sum_{l=1}^{N} \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial p_k} P(y_l \mid \theta) \\ \frac{\partial \mathcal{L}(\theta_m)}{\partial \theta_{h|k}} &= \sum_{l=1}^{N} \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial \theta_{h|k}} P(y_l \mid \theta) \end{aligned}$$

where

$$\frac{\partial}{\partial p_k} P(y_l \mid \theta) = \prod_{i=1}^{n} \theta_{i|k}^{y_{li}} \left(1 - \theta_{i|k}\right)^{1-y_{li}} - \prod_{i=1}^{n} \theta_{i|r_z}^{y_{li}} \left(1 - \theta_{i|r_z}\right)^{1-y_{li}} \tag{A.19}$$

$$\frac{\partial}{\partial \theta_{h|k}} P\left(y_l \mid \theta\right) \;=\; p_k \left(y_{li}\theta_{h|k}^{-1} + \left(1 - y_{il}\right)\left(1 - \theta_{h|k}\right)^{-1}\right) \prod_{i=1}^{n} \theta_{i|r_x}^{y_{li}} \left(1 - \theta_{i|r_x}\right)^{1-y_{li}} \tag{A.20}$$

A.5.2 Line Search Routine

The line search algorithm implemented was a simple golden ratio search. Here we outline it's application for the proportion vector problem; its extension to more complicated maximisation problems is simple. To begin with a maximum step length was determined; this is the largest value α can take without crossing the boundary of the parameter space. To find this value, denoted by α^{max}, we calculate the quantities α_i^{max} and take $\alpha^{max} = \max\left\{\alpha_i^{max}\right\}$ where

$$\alpha_i^{max} = \begin{cases} -\frac{p_i}{d_i} & \text{if } d_i < 0 \\ \frac{1-p_i}{d_i} & \text{if } d_i > 0 \end{cases} \tag{A.21}$$

The golden ratio search then proceeds as follows.

If we define

$$\mathcal{L}_\alpha\left(\theta_m^t\right) = \mathcal{L}\left(\theta_m^t + \alpha d^t\right)$$

then we begin by evaluating

$$\begin{aligned} \tau &= \frac{2}{1+\sqrt{5}} \\ \alpha_0^0 &= 0 \\ \alpha_1^0 &= 1 - \alpha^{max} \times \tau \\ \alpha_2^0 &= \alpha^{max} \times \tau \\ \alpha_3^0 &= \alpha^{max} \end{aligned}$$

We then repeat the following until our convergence criteria is reached

if $\mathcal{L}_{\alpha_1^t} > \mathcal{L}_{\alpha_2^t}$

$$\begin{aligned}
\alpha_3^{t+1} &= \alpha_2^t \\
\alpha_2^{t+1} &= \alpha_1^t \\
\alpha_1^{t+1} &= \alpha_3^t - \tau\left(\alpha_3^t - \alpha_0^t\right) \\
\alpha_0^{t+1} &= \alpha_0^t
\end{aligned}$$

else if $\mathcal{L}_{\alpha_1^t} < \mathcal{L}_{\alpha_2^t}$

$$\begin{aligned}
\alpha_3^{t+1} &= \alpha_3^t \\
\alpha_2^{t+1} &= \alpha_0^t + \tau\left(\alpha_3^t - \alpha_0^t\right) \\
\alpha_1^{t+1} &= \alpha_2^t \\
\alpha_0^{t+1} &= \alpha_1^t
\end{aligned}$$

We took our convergence criterion to be $\alpha_3^t - \alpha_0^t$ less than some predetermined value. Decreasing this value would increase the accuracy of the search at the cost of greater computation time; a value of 0.0001 was found to work well.

Appendix B

Proofs of Theorems from Chapter 2

Theorem 2.1 *If the EM algorithm is started at any independence model, that is to say either*

$$\theta^0_{i1k} = \ldots = \theta^0_{ir_zk} \quad \forall i,k$$

or

$$p^0_h = 1 \quad and \quad p^0_j = 0 \quad \forall j \neq h$$

then the EM algorithm will converge to the maximum likelihood independence model in just one iteration. In the first case it converges to

$$p^1_j = p^0_j \quad \forall j \qquad and \qquad \theta_{ijk} = \frac{1}{n}\sum_{m=1}^{N} I_k\left(y_{mi}\right) \quad \forall i,j,k$$

and in the second case to

$$p_h^1 = 1 \quad p_j^1 = 0 \quad \forall j \neq k \qquad \textit{and} \qquad \theta_{ihk} = \frac{1}{n} \sum_{m=1}^{N} I_k \left(y_{mi} \right) \quad \forall i, j, k.$$

Proof. We update p_j^t using

$$p_j^{t+1} = \frac{1}{N} \sum_{m=1}^{N} r_{mj}^t$$

and θ_{ijk} using

$$\begin{aligned} \theta_{ijk}^{t+1} &= \sum_{m=1}^{N} I_k \left(y_{mi} \right) r_{mj}^t \Big/ \sum_{m=1}^{N} r_{mj}^t \\ &= \sum_{m=1}^{N} I_k \left(y_{mi} \right) r_{mj}^t \Big/ N p_j^{t+1}, \end{aligned}$$

where

$$\begin{aligned} r_{mj}^t &= P \left(Z_m = j \mid y_m, \theta^t \right) \\ &= p_j^t P^t \left(y_m \mid Z = j \right) \Big/ \sum_{k=1}^{r_z} p_k^t P^t \left(y_m \mid Z = k \right) \qquad \forall m, j. \end{aligned}$$

We look at the two possible starting configurations separately

Case 1: $p_k^0 = 1$, $p_j^0 = 0$ for all $j \neq k$.

In this case it can be shown that

$$r_{mk}^0 = p_k^t P^t \left(y_m \mid Z = j \right) \Big/ \sum_{j=1}^{r_z} p_j^t P^t \left(y_m \mid Z = j \right)$$

$$= \; p_k^t P^t\left(y_m \mid Z = j\right) \Big/ p_k^t P^t\left(y_m \mid Z = j\right)$$

$$= \; 1$$

and

$$r_{mj}^0 \;=\; p_j^t P^t\left(y_m \mid Z = j\right) \Big/ \sum_{j=1}^{r_z} p_j^t P^t\left(y_m \mid Z = j\right)$$

$$= \; 0 \Big/ p_k^t P^t\left(y_m \mid Z = j\right)$$

$$= \; 0 \quad \forall j \neq k$$

and hence

$$p_k^1 = \frac{1}{N} \sum_{m=1}^{N} r_{mk}^0 = \frac{1}{N} \sum_{m=1}^{N} 1 = 1$$

and

$$p_j^1 = \frac{1}{N} \sum_{m=1}^{N} r_{mj}^0 = \frac{1}{N} \sum_{m=1}^{N} 0 = 0 \quad \forall j \neq k.$$

Also

$$\theta_{ijk}^1 \;=\; \sum_{m=1}^{N} I_k\left(y_{mi}\right) r_{mk}^t \Big/ N p_k^{t+1}$$

$$= \; \frac{1}{N} \sum_{m=1}^{N} I_k\left(y_{mi}\right).$$

Case 2: $\theta_{i1k}^0 = \ldots = \theta_{ir_zk}^0$ for all i, k.

In this case it can be shown that

$$\begin{aligned} r_{mk}^0 &= p_k^0 P^0(y_m \mid Z = k) \Big/ \sum_{j=1}^{r_z} p_j^0 P^0(y_m \mid Z = j) \\ &= p_k^0 \Big/ \sum_{j=1}^{r_z} p_j^0 \\ &= p_k^0. \end{aligned}$$

Hence

$$\begin{aligned} p_j^1 &= \frac{1}{N} \sum_{m=1}^{N} p_j^0 \\ &= p_j^0 \end{aligned}$$

and

$$\begin{aligned} \theta_{ijk} &= \sum_{m=1}^{N} I_k(y_{mi})\, p_j^0 \Big/ N p_j^1 \\ &= \frac{1}{N} \sum_{m=1}^{N} I_k(y_{mi}) \end{aligned}$$

So, in each case we reach the maximum likelihood independence model. Using the same procedure to consider subsequent iterations it can be seen that the ML independence model is also a stationary point for the EM algorithm.

□

Although we have shown that this point is a stationary point of the EM algorithm it is also clear that this point lies on a ridge of the maximum likelihood surface, rather than at a local maximum. This is because at this point we have a model identifiability problem. In the first case we are unable to identify

unique values for the θ_{ijk} where p_j is zero, and in the second we can see that it is not possible to uniquely identify the values of the p_k. Hence the ML surface will contain a ridge of equal likelihood corresponding to variation of these unidentifiable parameters.

Corollary 2.2 *The maximum likelihood independence model is a stationary point of the EM algorithm.*

Proof. Proof follows simply from Theorem 2.1. We simply observe that if at some iteration we are at the maximum likelihood independence model then Theorem 2.1 implies that we will remain at that point.

□

Theorem 2.3 *The gradient of $\mathcal{L}_\beta(\theta_m)$ at the point*

$$p_j = \frac{1}{r_z} \quad \forall j \quad and \quad \theta_{ijk} = \frac{1}{N}\sum_{m=1}^{N} I_k(y_{mi}) \quad \forall i,j,k$$

is zero for all $0 < \beta \leq 1$.

Proof. The condition for $\mathcal{L}_\beta(\theta_m)$ to have a stationary point is

$$\mathcal{L}_\beta(\theta_m) + \lambda\left(\sum_{j=1}^{r_z} p_j - 1\right) + \delta_{ij}\left(\sum_{k=1}^{r_i} \theta_{ijk} - 1\right) = 0 \tag{B.1}$$

where λ and δ_{ij} are Lagrange multipliers. Differentiating this with respect to p_j and θ_{ijk} this condition is equivalent to

$$\frac{\partial \mathcal{L}_\beta(\theta_m)}{\partial p_h} = -\lambda = -N \quad \forall h \tag{B.2}$$

and

$$\frac{\partial \mathcal{L}_\beta(\theta_m)}{\partial \theta_{efg}} = -\delta_{ef} = -\frac{N}{r_z} \quad \forall e, f. \tag{B.3}$$

The value of λ is determined by multiplying both sides of equation (B.2) by p_h and summing over h. Similarly the values of the δ_{ij} are determined by multiplying both sides of equation (B.3) by θ_{efg} and summing over g. Since we have

$$\frac{\partial \mathcal{L}_\beta(\theta_m)}{\partial p_h} = -\sum_{m=1}^{N}\left(p_h^{\beta-1}\prod_{i=1}^{n}\sum_{k=1}^{r_i} I_k(y_{mi})\,\theta_{ihk}^{\beta} \Big/ \sum_{j=1}^{r_z} p_j^{\beta}\prod_{i=1}^{n}\sum_{k=1}^{r_i} I_k(y_{mi})\,\theta_{ijk}^{\beta}\right) \tag{B.4}$$

and

$$\frac{\partial \mathcal{L}_\beta(\theta_m)}{\partial \theta_{efg}} = -\sum_{m=1}^{N}\left(\frac{I_g(y_{me})}{\theta_{efg}} p_f^{\beta}\prod_{i=1}^{n}\sum_{k=1}^{r_i} I_k(y_{mi})\,\theta_{ifk}^{\beta} \Big/ \sum_{j=1}^{r_z} p_j^{\beta}\prod_{i=1}^{n}\sum_{k=1}^{r_i} I_k(y_{mi})\,\theta_{ijk}^{\beta}\right) \tag{B.5}$$

it can be seen that the point in question satisfies the two conditions (B.2) and (B.3) and hence it must be a stationary point.

□

Theorem 2.4 *At the point given in (2.26) the Hessian of* $\mathcal{L}_\beta(\theta_m)$*, denoted by* H_β*, is of the form*

$$H_\beta = \begin{pmatrix} H_\beta^1 & 0 \\ 0 & H_\beta^2 \end{pmatrix}$$

Proof. We begin by determining the Hessian of a general naive Bayesian network. Using the fact that

$$p_{r_z} = 1 - \sum_{j=1}^{r_z-1} p_j$$

and

$$\theta_{ijr_i} = 1 - \sum_{k=1}^{r_i - 1} \theta_{ijk}$$

the first-order partial derivatives are then

$$\frac{\partial \mathcal{L}_\beta(\theta_m)}{\partial p_h} = -\sum_{m=1}^{N} \left(p_h^{\beta-1} \prod_{i=1}^{n} \sum_{k=1}^{r_i} I_k(y_{mi}) \theta_{ihk}^{\beta} - p_c^{\beta-1} \prod_{i=1}^{n} \sum_{k=1}^{r_i} I_k(y_{mi}) \theta_{ir_zk}^{\beta} \right) \Big/ \sum_{j=1}^{r_z} p_j^{\beta} \prod_{i=1}^{n} \sum_{k=1}^{r_i} I_k(y_{mi}) \theta_{ijk}^{\beta}$$

$$= -\sum_{m=1}^{N} \left(p_h^{\beta-1} \kappa_{mh} - p_{r_z}^{\beta-1} \kappa_{mr_z} \right) \Big/ \sum_{j=1}^{r_z} p_j^{\beta} \kappa_{mj}$$

$$\frac{\partial \mathcal{L}_\beta(\theta_m)}{\partial \theta_{efg}} = -\sum_{m=1}^{N} \left(\frac{I_g(y_{me})}{\theta_{efg}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}} \right) p_f^{\beta} \prod_{i=1}^{n} \sum_{k=1}^{r_i} I_k(y_{mi}) \theta_{ifk}^{\beta} \Big/ \sum_{j=1}^{r_z} p_j^{\beta} \prod_{i=1}^{n} \sum_{k=1}^{r_i} I(y_{mi}) \theta_{ijk}^{\beta}$$

$$= -\sum_{m=1}^{N} \left(\frac{I_g(y_{me})}{\theta_{efg}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}} \right) p_f^{\beta} \kappa_{mf} \Big/ \sum_{j=1}^{r_z} p_j^{\beta} \kappa_{mj}$$

where

$$\kappa_{mj} = \prod_{i=1}^{n} \sum_{k=1}^{r_i} I(y_{mi}) \theta_{ijk}^{\beta}.$$

The second-order partial derivatives are given by

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial p_h^2} = -\sum_{m=1}^{N} \left(\left((\beta - 1) \left(p_h^{\beta-2} \kappa_{mh} + p_{r_z}^{\beta-2} \kappa_{mr_z} \right) \cdot \sum_{j=1}^{r_z} \zeta_j^{\beta} \kappa_{mj} - \beta \left(p_h^{\beta-1} \kappa_{mh} - p_{r_z}^{\beta-1} \kappa_{mr_z} \right)^2 \right) \Big/ \left(\sum_{j=1}^{r_z} p_j^{\beta} \kappa_{mj} \right)^2 \right)$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial p_h \partial p_w} = -\sum_{m=1}^{N} \Bigg(\Bigg(\Big((\beta-1)\, p_{r_z}^{\beta-2} \kappa_{mr_z} \sum_{j=1}^{r_z} p_j^\beta \kappa_{mj} - \beta \left(p_h^{\beta-1} \kappa_{mh} - p_{r_z}^{\beta-1} \kappa_{mr_z} \right) \left(p_w^{\beta-1} \kappa_{mw} - p_{r_z}^{\beta-1} \kappa_{mr_z} \right) \Big) \Big/ \left(\sum_{j=1}^{r_z} p_j^\beta \kappa_{mj} \right)^2 \Bigg)$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial p_f \partial \theta_{efg}} = -\sum_{m=1}^{N} \Bigg(\Bigg(\beta \left(\frac{I_g(y_{me})}{\theta_{efg}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}} \right) p_f^{\beta-1} \kappa_{mf} \sum_{j=1}^{r_z} p_j^\beta \kappa_{mj} - \beta \left(\frac{I_g(y_{me})}{\theta_{efg}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}} \right) \times p_f^\beta \kappa_{mf} \left(p_f^{\beta-1} \kappa_{mf} - p_{r_z}^{\beta-1} \kappa_{mr_z} \right) \Bigg) \Big/ \left(\sum_{j=1}^{r_z} p_j^\beta \kappa_{mj} \right)^2 \Bigg)$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial p_h \partial \theta_{efg}} = -\sum_{m=1}^{N} -\beta \left(p_h^{\beta-1} \kappa_{mh} - \zeta_{r_z}^{\beta-1} \kappa_{mr_z} \right) \left(\frac{I_g(y_{me})}{\theta_{efg}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}} \right) p_f^\beta \kappa_{mf} \Big/ \left(\sum_{j=1}^{r_z} p_j^\beta \kappa_{mj} \right)^2$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial p_h \partial \theta_{ecg}} = -\sum_{m=1}^{N} \Bigg(\Bigg(-\beta \left(\frac{I_g(y_{me})}{\theta_{ecg}} - \frac{I_{r_e}(y_{me})}{\theta_{ecr_e}} \right) p_{r_z}^{\beta-1} \kappa_{mr_z} \sum_{j=1}^{r_z} p_j^\beta \kappa_{mj} - \beta \left(\frac{I_g(y_{me})}{\theta_{ecg}} - \frac{I_{r_e}(y_{me})}{\theta_{ecr_e}} \right)$$

$$\times p_{r_z}^{\beta} \kappa_{mr_z} \left(p_h^{\beta-1} \kappa_{mh} - p_{r_z}^{\beta-1} \kappa_{mr_z} \right) \Bigg) \Bigg/ \left(\sum_{j=1}^{r_z} p_j^{\beta} \kappa_{mj} \right)^2 \Bigg)$$

$$\frac{\partial^2 \mathcal{L}_\beta(\boldsymbol{\theta_m})}{\partial \theta_{efg}^2} = -\sum_{m=1}^{N} \Bigg(\Bigg((\beta-1) \left(\frac{I_g(y_{me})}{\theta_{efg}^2} + \frac{I_{r_e}(y_{me})}{\theta_{efr_e}^2} \right) p_f^{\beta} \kappa_{mf} \sum_{j=1}^{r_z} p_j^{\beta} \kappa_{mj} - \beta \left(\left(\frac{I_g(y_{me})}{\theta_{efg}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}} \right) p_f^{\beta} \kappa_{mf} \right)^2 \Bigg) \Bigg/ \left(\sum_{j=1}^{r_z} p_j^{\beta} \kappa_{mj} \right)^2 \Bigg)$$

$$\frac{\partial^2 \mathcal{L}_\beta(\boldsymbol{\theta_m})}{\partial \theta_{efg} \partial \theta_{tuv}} = \sum_{m=1}^{N} \beta \left(\frac{I_g(y_{me})}{\theta_{efg}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}} \right) p_f^{\beta} \kappa_{mf} \times \left(\frac{I_v(y_{mt})}{\theta_{tuv}} - \frac{I_{r_t}(y_{mt})}{\theta_{tur_t}} \right) p_u^{\beta} \kappa_{mu} \Bigg/ \left(\sum_{j=1}^{r_z} p_j^{\beta} \kappa_{mj} \right)^2$$

$$\frac{\partial^2 \mathcal{L}_\beta(\boldsymbol{\theta_m})}{\partial \theta_{efg} \partial \theta_{efv}} = -\sum_{m=1}^{N} \Bigg(\Bigg((\beta-1) \frac{I_{r_e}(y_{me})}{\theta_{efr_e}^2} p_f^{\beta} \kappa_{mf} \sum_{j=1}^{r_z} p_j^{\beta} \kappa_{mj} - \beta \left(\frac{I_g(y_{me})}{\theta_{efg}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}} \right) p_f^{\beta} \kappa_{mf} \Bigg) \times \left(\frac{I_v(y_{me})}{\theta_{efv}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}} \right) p_f^{\beta} \kappa_{mf} \Bigg) \Bigg/ \left(\sum_{j=1}^{r_z} p_j^{\beta} \kappa_{mj} \right)^2 \Bigg)$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial\theta_{efg}\partial\theta_{tfv}} = -\sum_{m=1}^{N}\left(\left(\beta\left(\frac{I_g(y_{me})}{\theta_{efg}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}}\right)\right.\right.$$
$$\times\left(\frac{I_v(y_{mt})}{\theta_{tfv}} - \frac{I_{r_t}(y_{mt})}{\theta_{tfr_t}}\right)p_f^\beta\kappa_{mf}\sum_{j=1}^{N}p_j^\beta\kappa_{mj}$$
$$-\beta\left(\frac{I_g(y_{me})}{\theta_{efg}} - \frac{I_{r_e}(y_{me})}{\theta_{efr_e}}\right)p_f^\beta\kappa_{mf}$$
$$\left.\times\left(\frac{I_v(y_{mt})}{\theta_{tfv}} - \frac{I_{r_t}(y_{mt})}{\theta_{tfr_t}}\right)p_f^\beta\kappa_{mf}\right)$$
$$\left.\Big/\left(\sum_{j=1}^{r_z}p_j^\beta\kappa_{mj}\right)^2\right)$$

At the point

$$p_j = \frac{1}{r_z}\quad\forall j \qquad \text{and} \qquad \theta_{ijk} = \frac{1}{n}\sum_{m=1}^{N}I_k(y_{mi})\quad\forall i,j,k$$

these second-order partial derivatives simplify to

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial p_h \partial p_w} = Nr_z(1-\beta)$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial p_h^2} = 2Nr_z(1-\beta)$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial p_f \partial\theta_{efg}} = \frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial p_h \partial\theta_{efg}} = \frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial p_h \partial\theta_{ecg}} = 0$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial\theta_{efg}^2} = \frac{N^2}{r_z^2}(\beta + r_z(1-\beta))\left(\frac{1}{\sum_m I_g(y_{me})} + \frac{1}{\sum_m I_{r_e}(y_{me})}\right)$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial\theta_{efg}\partial\theta_{tuv}} = \frac{N^2}{r_z^2}\beta\sum_{m=1}^{N}\left(\frac{I_g(y_{me})}{\sum_m I_g(y_{me})} - \frac{I_{r_e}(y_{me})}{\sum_m I_{r_e}(y_{me})}\right) \times \left(\frac{I_v(y_{mt})}{\sum_m I_v(y_{mt})} - \frac{I_{r_t}(y_{mt})}{\sum_m I_{r_t}(y_{mt})}\right)$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial\theta_{efg}\partial\theta_{efv}} = \frac{N^2}{r_z^2}(\beta + r_z(1-\beta))\frac{1}{\sum_m I_{r_e}(x_{me})}$$

$$\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial\theta_{efg}\partial\theta_{tfv}} = \frac{N^2}{r_z^2}\beta(1-r_z)\sum_{m=1}^{N}\left(\frac{I_g(y_{me})}{\sum_m I_g(y_{me})} - \frac{I_{r_e}(y_{me})}{\sum_m I_{r_e}(y_{me})}\right) \times \left(\frac{I_v(y_{mt})}{\sum_m I_v(y_{mt})} - \frac{I_{r_t}(y_{mt})}{\sum_m I_{r_t}(y_{mt})}\right)$$

Hence it can be seen that, at the point given in (2.26) the Hessian of $\mathcal{L}_\beta(\theta_m)$, denoted by H_β, is of the form

$$H_\beta = \begin{pmatrix} H_\beta^1 & 0 \\ 0 & H_\beta^2 \end{pmatrix}.$$

□

Theorem 2.5 *The matrix H_β^1 is positive definite for all naive Bayesian networks.*

Proof Using the partial derivatives we have already derived we can see that

$$H_\beta^1 = Nr_z(1-\beta)\begin{pmatrix} 2 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 1 & \cdots & 1 \\ 1 & 1 & 2 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 2 \end{pmatrix}$$

$$= \; Nr_z\,(1-\beta)\left(I_n + 11^T\right).$$

it therefore follows that for any $\boldsymbol{x} \in \mathbb{R}^n$ that

$$\begin{aligned} \boldsymbol{x}^T H_\beta^1 \boldsymbol{x} &= Nr_z\,(1-\beta)\,\boldsymbol{x}^T\left(I + 11^T\right)\boldsymbol{x} \\ &= Nr_z\,(1-\beta)\left\{\boldsymbol{x}^T\boldsymbol{x} + \left(\boldsymbol{x}^T\mathbf{1}\right)^2\right\} \\ &\geq 0 \qquad\qquad \forall 0 < \beta < 1 \quad \text{with equality iff } \boldsymbol{x} = \mathbf{0}, \end{aligned}$$

so H_β^1 is positive definite.

□

Theorem 2.6 *Assuming that our maximum likelihood independence model is correct, the expected value of the matrix H_β^2 is positive definite for naive Bayesian networks with binary observable variables.*

Proof There are four possible values that $\frac{\partial^2 \mathcal{L}_\beta(\boldsymbol{\theta}_m)}{\partial\theta_{efg}\partial\theta_{tuv}}$ can take, and assuming that our ML independence model is correct we can calculate the probability of each of these outcomes

$\frac{\partial^2 \mathcal{L}_\beta(\boldsymbol{\theta}_m)}{\partial\theta_{efg}\partial\theta_{tuv}} =$	with probability
$\frac{N^2\beta}{r_z^2}\left(\sum_m I_g\left(y_{me}\right)\sum_m I_v\left(y_{mt}\right)\right)^{-1}$	$\frac{1}{N^2}\sum_m I_g\left(y_{me}\right)\sum_m I_v\left(y_{mt}\right)$
$-\frac{N^2\beta}{r_z^2}\left(\sum_m I_g\left(y_{me}\right)\sum_m I_{r_t}\left(y_{mt}\right)\right)^{-1}$	$\frac{1}{N^2}\sum_m I_g\left(y_{me}\right)\sum_m I_{r_t}\left(y_{mt}\right)$
$\frac{N^2\beta}{r_z^2}\left(\sum_m I_{r_e}\left(y_{me}\right)\sum_m I_v\left(y_{mt}\right)\right)^{-1}$	$\frac{1}{N^2}\sum_m I_{r_e}\left(y_{me}\right)\sum_m I_v\left(y_{mt}\right)$
$-\frac{N^2\beta}{r_z^2}\left(\sum_m I_{r_e}\left(y_{me}\right)\sum_m I_{r_t}\left(y_{mt}\right)\right)^{-1}$	$\frac{1}{N^2}\sum_m I_{r_e}\left(y_{me}\right)\sum_m I_{r_t}\left(y_{mt}\right)$

Hence

$$\mathbb{E}\left(\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial\theta_{efg}\partial\theta_{tuv}}\right) = 0$$

We can similarly show that

$$\mathbb{E}\left(\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial\theta_{efg}\partial\theta_{tfv}}\right) = 0$$

and the final expectation we need is

$$\mathbb{E}\left(\frac{\partial^2 \mathcal{L}_\beta(\theta_m)}{\partial\theta^2_{efg}}\right) = \frac{N^2}{r_z^2}\left(\beta + r_z(1-\beta)\right)\left(\frac{1}{\sum_m I_g(y_{me})} + \frac{1}{\sum_m I_{r_e}(y_{me})}\right) > 0$$

Hence $\mathbb{E}\left(H_\beta^2\right)$ is a diagonal matrix with positive diagonal elements and must be positive definite.

□

Theorem B.1 *For naive Bayesian networks with binary observable nodes, the Hessian at the point given in (2.26) will be positive definite for*

$$\beta \leq \frac{r_z}{2n(r_z - 1)}$$

Proof We make use of Gerschgorin's 'circle theorem' which states that

Sub-Theorem B.2 (Gerschgorin's 'Circle Theorem') *Every eigenvalue of an $n \times n$ matrix A lies in at least one of the circles $C_1, \ldots, C_n$, where C_i has its centre at the diagonal entry a_{ii} and its radius $\rho_i = \sum_{j \neq i} \mid a_{ij} \mid$ equal to the absolute sum along the rest of the row.*

As we are dealing with a symmetric real matrix this leads us to the corollary

Corollary B.3 *The real symmetric matrix A is positive definite if $a_{ii} > 0$ and $a_{ii} > \rho_i$ where*

$$\rho_i = \sum_{j \neq i} | a_{ij} |$$

For our matrix H_β^2 it is clear that the terms on the diagonal are all positive, so our corollary of Gerschgorin's theorem means that H_β^2 must be positive definite if

$$\frac{N^2}{r_z^2}(\beta + r_z(1-\beta))\left(\frac{1}{\sum_m y_{mi}} + \frac{1}{N - \sum_m y_{mi}}\right)$$
$$\geq \frac{N^2}{r_z^2}\beta(r_z - 1)\sum_{j \neq i}\left|\sum_{m=1}^{N}\left(\frac{y_{mi}}{\sum_m y_{mi}} - \frac{1-y_{mi}}{N - \sum_m y_{mi}}\right)\left(\frac{y_{mj}}{\sum_m y_{mj}} - \frac{1-y_{mj}}{N - \sum_m y_{mj}}\right)\right|$$
$$+ (r_z - 1)\frac{N^2}{r_z^2}\beta\sum_{j=1}^{n}\left|\sum_{m=1}^{N}\left(\frac{y_{mi}}{\sum_m y_{mi}} - \frac{1-y_{mi}}{N - \sum_m y_{mi}}\right)\left(\frac{y_{mj}}{\sum_m y_{mj}} - \frac{1-y_{mj}}{N - \sum_m y_{mj}}\right)\right|$$

Sub-Theorem B.4 *For any $1 \leq j \leq n$*

$$\left(\frac{1}{\sum_m y_{mi}} + \frac{1}{N - \sum_m y_{mi}}\right)$$
$$\geq \left|\sum_{m=1}^{N}\left(\frac{y_{mi}}{\sum_m y_{mi}} - \frac{1-y_{mi}}{N - \sum_m y_{mi}}\right)\left(\frac{y_{mj}}{\sum_m y_{mj}} - \frac{1-y_{mj}}{N - \sum_m y_{mj}}\right)\right|$$

Proof The summation between the modulus signs will take its maximum value when the condition $y_{mi} = 1 \Rightarrow y_{mj} = 1$ and will take its minimum value when $y_{mi} = 1 \Rightarrow y_{mj} = 0$. So the summation's maximum value will be

$$\sum_{m=1}^{N}\left(\frac{y_{mi}}{(\sum_m y_{mi})^2} + \frac{1-y_{mi}}{(N - \sum_m y_{mi})^2}\right) = \left(\frac{1}{\sum_m y_{mi}} + \frac{1}{N - \sum_m y_{mi}}\right)$$

and its minimum value will be

$$-\sum_{m=1}^{N}\frac{y_{mi}\left(1-y_{mj}\right)}{\sum_m y_{mi}\left(N-\sum_m y_{mj}\right)}+\frac{\left(N-y_{mi}\right)y_{mj}}{\left(N-\sum_m y_{mi}\right)\sum_m y_{mj}}$$
$$=-\left(\frac{1}{\sum_m y_{mi}}+\frac{1}{N-\sum_m y_{mi}}\right).$$

□

This leads to the condition

$$\frac{N^2}{r_z^2}\left(\beta+r_z\left(1-\beta\right)\right)\left(\frac{1}{\sum_m y_{mi}}+\frac{1}{N-\sum_m y_{mi}}\right)$$
$$\geq\left(n-1\right)\frac{N^2}{r_z^2}\beta\left(1-r_z\right)\left(\frac{1}{\sum_m y_{mi}}+\frac{1}{N-\sum_m y_{mi}}\right)$$
$$+n\left(r_z-1\right)\frac{N^2}{r_z^2}\beta\left(\frac{1}{\sum_m y_{mi}}+\frac{1}{N-\sum_m y_{mi}}\right)$$

and hence a sufficient, though not necessary condition for H_β^2 to be positive definite is

$$\beta\leq\frac{r_z}{2n\left(r_z-1\right)},$$

which proves Theorem 2.7.

□

Appendix C

Derivation of Hessian Matrices Used in the Laplace Approximation

C.1 Traditional Basis

We need to calculate

$$\begin{aligned} -H_{\theta_1\theta_2} &= -\frac{\partial^2}{\partial\theta_1\partial\theta_2}\log\left(P\left(D \mid \boldsymbol{\theta}\right)P\left(\boldsymbol{\theta}\right)\right) \\ &= -\frac{\partial^2}{\partial\theta_1\partial\theta_2}\log P\left(D \mid \boldsymbol{\theta}\right) - \frac{\partial^2}{\partial\theta_1\partial\theta_2}\log P\left(\boldsymbol{\theta}\right) \end{aligned} \tag{C.1}$$

where

$$P\left(D \mid \boldsymbol{\theta}\right) = \prod_{l=1}^{N} P\left(y_l \mid \boldsymbol{\theta}\right)$$

$$= \prod_{l=1}^{N} \sum_{j=1}^{c} p_j \prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1 - \theta_{i|j}\right)^{1-y_{li}} \tag{C.2}$$

and

$$P(\theta) = \frac{\Gamma(c\alpha)}{\Gamma(\alpha)^c} \prod_{j=1}^{c} p_j^{\alpha-1} . \prod_{j=1}^{c} \prod_{i=1}^{n} \frac{\Gamma(2\alpha)}{\Gamma(\alpha)^2} \theta_{i|j}^{\alpha-1} \left(1 - \theta_{i|j}\right)^{\alpha-1} \tag{C.3}$$

Starting with the first term on the right hand side of Equation C.1 we make use of the identity

$$-\frac{\partial^2}{\partial\theta_1 \partial\theta_2} \log\left(P(y_l \mid \theta)\right) = S_{\theta_1}(y_l) S_{\theta_2}(y_l) - D_{\theta_1\theta_2}(y_l) \tag{C.4}$$

where $S_{\theta_i}(y_l)$ is the score and is given by

$$S_{\theta_i}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial\theta_i} P(y_l \mid \theta)$$

and $D_{\theta_1\theta_2}(y_l)$ is is given by

$$D_{\theta_1\theta_2}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial^2}{\partial\theta_1 \partial\theta_2} P(y_l \mid \theta)$$

so we have

$$-H_{\theta_1\theta_2} = \sum_{l=1}^{N} \left(S_{\theta_1}(y_l) S_{\theta_2}(y_l) - D_{\theta_1\theta_2}(y_l)\right)$$

Beginning with the scores, and using the fact that $p_1 = 1 - \sum_{j=2}^{c} p_j$ to eliminate p_1, we have

$$S_{p_j}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial p_j} \left(\sum_{k=1}^{c} p_k \prod_{i=1}^{n} \theta_{i|k}^{y_{li}} \left(1 - \theta_{i|k}\right)^{1-y_{li}} \right)$$

$$= \frac{1}{P(y_l \mid \theta)} \left(\prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1-\theta_{i|j}\right)^{1-y_{li}} - \prod_{i=1}^{n} \theta_{i|1}^{y_{li}} \left(1-\theta_{i|1}\right)^{1-y_{li}} \right)$$

$$S_{\theta_{i|j}}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial \theta_{i|j}} \left(\sum_{k=1}^{c} p_k \prod_{h=1}^{n} \theta_{h|k}^{y_{li}} \left(1-\theta_{h|k}\right)^{1-y_{li}} \right)$$

$$= \frac{1}{P(y_l \mid \theta)} \left(p_j \left(\frac{y_{li}}{\theta_{i|j}} - \frac{1-y_{li}}{1-\theta_{i|j}} \right) \prod_{h=1}^{n} \theta_{h|j}^{y_{lh}} \left(1-\theta_{h|j}\right)^{1-y_{lh}} \right)$$

(C.5)

Now the values of $D_{\theta_1\theta_2}(y)$:

$$D_{p_j p_k}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial p_k} \left(\prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1-\theta_{i|j}\right)^{1-y_{li}} - \prod_{i=1}^{n} \theta_{i|1}^{y_{li}} \left(1-\theta_{i|1}\right)^{1-y_{li}} \right)$$

$$= 0$$

$$D_{p_j p_j}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial p_j} \left(\prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1-\theta_{i|j}\right)^{1-y_{li}} - \prod_{i=1}^{n} \theta_{i|1}^{y_{li}} \left(1-\theta_{i|1}\right)^{1-y_{li}} \right)$$

$$= 0$$

$$D_{p_j \theta_{i|j}}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial \theta_{i|j}} \left(\prod_{h=1}^{n} \theta_{h|j}^{y_{lh}} \left(1-\theta_{h|j}\right)^{1-y_{lh}} - \prod_{h=1}^{n} \theta_{h|1}^{y_{lh}} \left(1-\theta_{h|1}\right)^{1-y_{lh}} \right)$$

$$= \frac{1}{P(y_l \mid \theta)} \left(\frac{y}{\theta_{i|j}} - \frac{1-y}{1-\theta_{i|j}} \right) \prod_{h=1}^{n} \theta_{h|j}^{y_{lh}} \left(1-\theta_{h|j}\right)^{1-y_{lh}}$$

$$D_{p_j \theta_{i|1}}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial \theta_{i|1}} \left(\prod_{h=1}^{n} \theta_{h|j}^{y_{lh}} \left(1-\theta_{h|j}\right)^{1-y_{lh}} - \prod_{h=1}^{n} \theta_{h|1}^{y_{lh}} \left(1-\theta_{h|1}\right)^{1-y_{lh}} \right)$$

$$= -\frac{1}{P(y_l \mid \theta)} \left(\frac{y}{\theta_{i|1}} - \frac{1-y}{1-\theta_{i|1}} \right) \prod_{h=1}^{n} \theta_{h|1}^{y_{lh}} \left(1-\theta_{h|1}\right)^{1-y_{lh}}$$

$$D_{\theta_{i|j}\theta_{i|j}}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial \theta_{i|j}} \left(p_j \left(\frac{y}{\theta_{i|j}} - \frac{1-y}{1-\theta_{i|j}} \right) \prod_{h=1}^{n} \theta_{h|j}^{y_{lh}} \left(1-\theta_{h|j}\right)^{1-y_{lh}} \right)$$

$$= 0$$

$$D_{\theta_{i|j}\theta_{k|j}}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial \theta_{k|j}} \left(p_j \left(\frac{y}{\theta_{i|j}} - \frac{1-y}{1-\theta_{i|j}} \right) \prod_{h=1}^{n} \theta_{h|j}^{y_{lh}} \left(1-\theta_{h|j}\right)^{1-y_{lh}} \right)$$

$$= \frac{1}{P(y_l \mid \theta)} \left(p_j \left(\frac{y_{li}}{\theta_{i|j}} - \frac{1-y_{li}}{1-\theta_{i|j}} \right) \left(\frac{y_{lk}}{\theta_{k|j}} - \frac{1-y_{lk}}{1-\theta_{k|j}} \right) \prod_{h=1}^{n} \theta_{h|j}^{y_{lh}} \left(1-\theta_{h|j}\right)^{1-y_{lh}} \right)$$

$$D_{\theta_{i|j}\theta_{k|h}}(y_l) = \frac{1}{P(y_l \mid \theta)} \frac{\partial}{\partial \theta_{k|h}} \left(p_j \left(\frac{y}{\theta_{i|j}} - \frac{1-y}{1-\theta_{i|j}} \prod_{i=1}^{n} \theta_{i|j}^{y_{li}} \left(1-\theta_{i|j}\right)^{1-y_{li}} \right) \right)$$

$$= 0 \tag{C.6}$$

The second term is somewhat simpler. If we note that

$$\log P(\theta) = \text{constant} + (\alpha - 1) \left[\sum_{j=1}^{c} \log p_j + \sum_{j=1}^{c} \sum_{i=1}^{n} \left(\log \theta_{i|j} + \log \left(1-\theta_{i|j}\right) \right) \right]$$

we can evaluate the second order derivatives directly

$$\frac{\partial}{\partial p_j} \log P(\theta) = (\alpha - 1) \frac{\partial}{\partial p_j} \sum_{k=1}^{c} \log p_k$$

$$= (\alpha - 1) \left(\frac{1}{p_j} - \frac{1}{p_1} \right)$$

$$\frac{\partial}{\partial \theta_{i|j}} \log P(\theta) = (\alpha - 1) \frac{\partial}{\partial p_j} \left(\log \theta_{i|j} + \log \left(1-\theta_{i|j}\right) \right)$$

$$= (\alpha - 1) \left(\frac{1}{\theta_{i|j}} - \frac{1}{1-\theta_{i|j}} \right)$$

$$\frac{\partial^2}{\partial p_j^2} \log P(\theta) = (\alpha - 1) \frac{\partial}{\partial p_j} \left(\frac{1}{p_j} - \frac{1}{p_1} \right)$$

$$= -(\alpha - 1) \left(\frac{1}{p_j^2} + \frac{1}{p_1^2} \right)$$

$$\begin{aligned}
\frac{\partial^2}{\partial p_j \partial p_k} \log P(\boldsymbol{\theta}) &= (\alpha - 1) \frac{\partial}{\partial p_k} \left(\frac{1}{p_j} - \frac{1}{p_1} \right) \\
&= -(\alpha - 1) \frac{1}{p_1^2}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2}{\partial p_k \theta_{i|j}} \log P(\boldsymbol{\theta}) &= (\alpha - 1) \frac{\partial}{\partial \theta_{i|j}} \left(\left(\frac{1}{p_j} - \frac{1}{p_1} \right) \right) \\
&= 0
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2}{\partial \theta_{i|j}^2} \log P(\boldsymbol{\theta}) &= (\alpha - 1) \frac{\partial}{\partial \theta_{i|j}} \left(\frac{1}{\theta_{i|j}} - \frac{1}{1 - \theta_{i|j}} \right) \\
&= -(\alpha - 1) \left(\frac{1}{\theta_{i|j}^2} - \frac{1}{(1 - \theta_{i|j})^2} \right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2}{\partial \theta_{i|j} \theta_{h=|k}} \log P(\boldsymbol{\theta}) &= (\alpha - 1) \frac{\partial}{\partial \theta_{h|k}} \left(\frac{1}{\theta_{i|j}} - \frac{1}{1 - \theta_{i|j}} \right) \\
&= 0
\end{aligned} \tag{C.7}$$

C.2 Softmax Basis

In the softmax basis we reparameterise our model using

$$\begin{aligned}
\zeta_j &= \log \frac{p_j}{p_1} \qquad j = 2, \ldots, c \\
\phi_{i|j} &= \log \frac{\theta_{i|j}}{1 - \theta_{i|j}} \qquad \forall i, j
\end{aligned} \tag{C.8}$$

and the inverse transformation is given by

$$p_1(\boldsymbol{\phi}) = \frac{1}{1 + \sum_{k=2}^{c} e^{\zeta_k}}$$

$$
\begin{aligned}
p_j(\phi) &= \frac{e^{\zeta_j}}{1+\sum_{k=2}^{c} e^{\zeta_k}} \qquad j = 2, \ldots, c \\
\theta_{i|j}(\phi) &= \frac{e^{\phi_{i|j}}}{1+e^{\phi_{i|j}}}
\end{aligned}
\tag{C.9}
$$

We now need to calculate

$$
-H_{\phi_1,\phi_2} = -\frac{\partial^2}{\partial\phi_1\partial\phi_2}\log P(d \mid \phi) - \frac{\partial^2}{\partial\phi_1\partial\phi_2}\log P(\phi) \tag{C.10}
$$

where

$$
\begin{aligned}
P(D \mid \phi) &= \prod_{l=1}^{N} P(y_l \mid \phi) \\
&= \prod_{l=1}^{N}\sum_{j=1}^{c} p_j(\phi)\prod_{i=1}^{n}\theta_{i|j}(\phi)^{y_{li}}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}} \\
&= \prod_{l=1}^{N}\left(\frac{1}{1+\sum_{k=2}^{c} e^{\zeta_k}}\prod_{i=1}^{n}\left(\frac{e^{\phi_{i|1}}}{1+e^{\phi_{i|1}}}\right)^{y_{li}}\left(\frac{1}{1+e^{\phi_{i|1}}}\right)^{1-y_{li}}\right. \\
&\quad \left. + \sum_{j=2}^{c}\frac{e^{\zeta_j}}{1+\sum_{k=2}^{c} e^{\zeta_k}}\prod_{i=1}^{n}\left(\frac{e^{\phi_{i|j}}}{1+e^{\phi_{i|j}}}\right)^{y_{li}}\left(\frac{1}{1+e^{\phi_{i|j}}}\right)^{1-y_{li}}\right)
\end{aligned}
\tag{C.11}
$$

and

$$
\begin{aligned}
P(\phi) &= \frac{\Gamma(c(\alpha+1))}{\Gamma(\alpha+1)^c}\prod_{j=1}^{c} p_j(\phi)^{\alpha}\prod_{i=1}^{n}\prod_{j=1}^{c}\frac{\Gamma(2(\alpha+1))}{\Gamma(\alpha+1)^2}\theta_{i|j}(\phi)^{\alpha}\left(1-\theta_{i|j}(\phi)\right)^{\alpha} \\
&= \frac{\Gamma(c(\alpha+1))}{\Gamma(\alpha+1)^c}\left(\frac{1}{1+\sum_{k=2}^{c} e^{\zeta_k}}\right)^{\alpha}\prod_{j=2}^{c}\left(\frac{e^{\zeta_j}}{1+\sum_{k=2}^{c} e^{\zeta_k}}\right)^{\alpha} \\
&\quad \times\prod_{i=1}^{n}\prod_{j=1}^{c}\frac{\Gamma(2(\alpha+1))}{\Gamma(\alpha+1)^2}\left(\frac{e^{\phi_{i|j}}}{1+e^{\phi_{i|j}}}\right)^{\alpha}\left(\frac{1}{1+e^{\phi_{i|j}}}\right)^{\alpha}
\end{aligned}
\tag{C.12}
$$

Use the same identity as before to simplify the necessary calculations and note the following first and second order partial derivatives:

$$\begin{aligned}\frac{\partial p_1(\phi)}{\partial \zeta_j} &= \frac{-e^{\zeta_j}}{\left(1+\sum_{k=2}^{c} e^{\zeta_k}\right)^2} \\ &= -p_1(\phi)\, p_j(\phi)\end{aligned}$$

$$\begin{aligned}\frac{\partial p_l(\phi)}{\partial \zeta_j} &= \frac{-e^{\zeta_l} e^{\zeta_j}}{\left(1+\sum_{k=2}^{c} e^{\zeta_k}\right)^2} \\ &= -p_l(\phi)\, p_j(\phi)\end{aligned}$$

$$\begin{aligned}\frac{\partial p_j(\phi)}{\partial \zeta_j} &= \frac{e^{\zeta_j}}{1+\sum_{k=2}^{c} e^{\zeta_k}} - \frac{e^{2\zeta_j}}{\left(1+\sum_{k=2}^{c} e^{\zeta_k}\right)^2} \\ &= p_j(\phi)\left(1-p_j(\phi)\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 p_1(\phi)}{\partial \zeta_j^2} &= \frac{\partial}{\partial \zeta_j}\left(-p_1(\phi)\, p_j(\phi)\right) \\ &= p_1(\phi)\, p_j(\phi)^2 - p_1(\phi)\, p_j(\phi)\left(1-p_j(\phi)\right) \\ &= p_1(\phi)\, p_j(\phi)\left(2p_j(\phi)-1\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 p_l(\phi)}{\partial \zeta_j^2} &= \frac{\partial}{\partial \zeta_j}\left(-p_l(\phi)\, p_j(\phi)\right) \\ &= p_l(\phi)\, p_j(\phi)^2 - p_l(\phi)\, p_j(\phi)\left(1-p_j(\phi)\right) \\ &= p_l(\phi)\, p_j(\phi)\left(2p_j(\phi)-1\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2 p_j(\phi)}{\partial \zeta_j^2} &= \frac{\partial}{\partial \zeta_j}\left(p_j(\phi)\left(1-p_j(\phi)\right)\right) \\ &= p_j(\phi)\left(1-p_j(\phi)\right) - 2p_j(\phi)^2\left(1-p_j(\phi)\right)\end{aligned}$$

$$= p_j(\phi)(1-p_j(\phi))(1-2p_j(\phi))$$

$$\begin{aligned}
\frac{\partial^2 p_1(\phi)}{\partial\zeta_j\partial\zeta_k} &= \frac{\partial}{\partial\zeta_k} - p_1(\phi)p_j(\phi) \\
&= 2p_1(\phi)p_j(\phi)p_k(\phi)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 p_l(\phi)}{\partial\zeta_j\partial\zeta_k} &= \frac{\partial}{\partial\zeta_k} - p_l(\phi)p_j(\phi) \\
&= 2p_l(\phi)p_j(\phi)p_k(\phi)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 p_j(\phi)}{\partial\zeta_j\partial\zeta_k} &= \frac{\partial}{\partial\zeta_k} p_j(\phi)(1-p_j(\phi)) \\
&= p_j(\phi)p_k(\phi)(2p_j(\phi)-1) \\
\frac{\partial\theta_{i|j}(\phi)}{\partial\phi_{i|j}} &= \frac{\partial}{\partial\phi_{i|j}} \frac{e^{\phi_{i|j}}}{1+e^{\phi_{i|j}}} \\
&= \frac{e^{\phi_{i|j}}}{1+e^{\phi_{i|j}}} - \frac{e^{2\phi_{i|j}}}{\left(1+e^{\phi_{i|j}}\right)^2} \\
&= \theta_{i|j}(\phi)\left(1-\theta_{i|j}(\phi)\right)
\end{aligned} \tag{C.13}$$

$$\begin{aligned}
\frac{\partial}{\partial\phi_{i|j}}\theta_{i|j}(\phi)^{y_{li}}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}} &= y_{li}\theta_{i|j}(\phi)^{y_{li}-1}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}} \frac{\partial\theta_{i|j}(\phi)}{\partial\phi_{i|j}} \\
&\quad -(1-y_{li})\theta_{i|j}(\phi)^{y_{li}}\left(1-\theta_{i|j}(\phi)\right)^{-y_{li}} \frac{\partial\theta_{i|j}(\phi)}{\partial\phi_{i|j}} \\
&= \theta_{i|j}(\phi)^{y_{li}}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}} \\
&\quad \times \left(y_{li}\theta_{i|j}(\phi)^{-1}\theta_{i|j}(\phi)\left(1-\theta_{i|j}(\phi)\right)\right. \\
&\quad \left. -(1-y_{li})\theta_{i|j}(\phi)\left(1-\theta_{i|j}(\phi)\right)^{-1}\left(1-\theta_{i|j}(\phi)\right)\right) \\
&= \theta_{i|j}(\phi)^{y_{li}}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}}\left(y_{li}-\theta_{i|j}(\phi)\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2}{\partial\phi_{i|j}^2}\theta_{i|j}(\phi)^{y_{li}}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}} &= \theta_{i|j}(\phi)^{y_{li}}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}} \\
&\quad\times\left(\left(y-\theta_{i|j}(\phi)\right)^2+\frac{\partial}{\partial\phi_{i|j}}\left(y_{li}-\theta_{i|j}(\phi)\right)\right) \\
&= \theta_{i|j}(\phi)^{y_{li}}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}} \\
&\quad\times\left(\left(y_{li}-\theta_{i|j}(\phi)\right)^2-\theta_{i|j}(\phi)\left(1-\theta_{i|j}(\phi)\right)\right) \qquad \text{(C.14)}
\end{aligned}$$

Then

$$\begin{aligned}
S_{\zeta_j}(y) &= \frac{1}{P(y\mid\phi)}\frac{\partial}{\partial\zeta_j}\left(\sum_{j=1}^{c}p_j(\phi)\prod_{i=1}^{n}\theta_{i|j}(\phi)^{yli}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}}\right) \\
&= \frac{1}{P(y\mid\phi)}\sum_{j=1}^{c}\frac{\partial}{\partial\zeta_j}p_j(\phi)\prod_{i=1}^{n}\theta_{i|j}(\phi)^{yli}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}} \\
&= \frac{1}{P(y\mid\phi)}\left(p_j(\phi)\left(1-p_j(\phi)\right)\prod_{i=1}^{n}\theta_{i|j}(\phi)^{y_{li}}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}}\right. \\
&\quad\left.-\sum_{k\neq j}p_j(\phi)\,p_k(\phi)\prod_{i=1}^{n}\theta_{i|k}(\phi)^{y_{li}}\left(1-\theta_{i|k}(\phi)\right)^{1-y_{li}}\right)
\end{aligned}$$

$$\begin{aligned}
S_{\phi_{i|j}}(y) &= \frac{1}{P(y\mid\phi)}\frac{\partial}{\partial\phi_{i|j}}\left(\sum_{j=1}^{c}p_j(\phi)\prod_{i=1}^{n}\theta_{i|j}(\phi)^{yli}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}}\right) \\
&= \frac{1}{P(y\mid\phi)}p_j(\phi)\prod_{k=1}^{n}\theta_{k|j}(\phi)^{y_{lk}}\left(1-\theta_{k|j}(\phi)\right)^{1-y_{lk}}\left(y_{li}-\theta_{i|j}(\phi)\right)
\end{aligned} \qquad \text{(C.15)}$$

$$D_{\zeta_j\zeta_j}(y_l) = \frac{1}{P(y\mid\phi)}\frac{\partial^2}{\partial\zeta_j^2}\left(\sum_{j=1}^{c}p_j(\phi)\prod_{i=1}^{n}\theta_{i|j}(\phi)^{yli}\left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}}\right)$$

$$= \frac{1}{P(y \mid \phi)} \left(p_j(\phi)(1-p_j(\phi))(1-2p_j(\phi)) \prod_{i=1}^{n} \theta_{i|j}(\phi)^{y_{li}} \left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}} \right.$$
$$\left. + \sum_{k \neq j} p_j(\phi) p_k(\phi)(2p_j(\phi)-1) \prod_{i=1}^{n} \theta_{i|k}(\phi)^{y_{li}} \left(1-\theta_{i|k}(\phi)\right)^{1-y_{li}} \right)$$

$$D_{\zeta_j \zeta_m}(y_l) = \frac{1}{P(y \mid \phi)} \left(p_j(\phi) p_m(\phi)(2p_j(\phi)-1) \prod_{i=1}^{n} \theta_{i|j}(\phi)^{y_{li}} \left(1-\theta_{i|j}(\phi)\right)^{1-y_{li}} \right.$$
$$+ p_j(\phi) p_m(\phi)(2p_m(\phi)-1) \prod_{i=1}^{n} \theta_{i|m}(\phi)^{y_{li}} \left(1-\theta_{i|m}(\phi)\right)^{1-y_{li}}$$
$$\left. + \sum_{k \neq j,m} 2p_j(\phi) p_l(\phi) p_k(\phi) \prod_{i=1}^{n} \theta_{i|k}(\phi)^{y_{li}} \left(1-\theta_{i|k}(\phi)\right)^{1-y_{li}} \right)$$

$$D_{\zeta_j \phi_{i|j}}(y_l) = \frac{1}{P(y \mid \phi)} p_j(\phi)(1-p_j(\phi)) \left(y_{li}-\theta_{i|j}(\phi)\right) \prod_{k=1}^{n} \theta_{k|j}^{y_{lk}} \left(1-\theta_{k|j}\right)^{1-y_{lk}}$$

$$D_{\zeta_j \phi_{i|m}}(y_l) = -\frac{1}{P(y \mid \phi)} p_j(\phi) p_l(\phi) \left(y_{li}-\theta_{i|m}(\phi)\right) \prod_{k=1}^{n} \theta_{k|m}^{y_{lk}} \left(1-\theta_{k|m}\right)^{1-y_{lk}}$$

$$D_{\phi_{i|j} \phi_{i|j}}(y_l) = \frac{1}{P(y \mid \phi)} p_j(\phi) \prod_{k=1}^{n} \theta_{k|j}^{y_{lk}} \left(1-\theta_{k|j}\right)^{1-y_{lk}}$$
$$\times \left(\left(y-\theta_{i|j}(\phi)\right)^2 - \theta_{i|j}(\phi)\left(1-\theta_{i|j}(\phi)\right) \right)$$

$$D_{\phi_{i|j} \phi_{m|j}}(y_l) = \frac{1}{P(y \mid \phi)} p_j(\phi) \prod_{k=1}^{n} \theta_{k|j}^{y_{lk}} \left(1-\theta_{k|j}\right)^{1-y_{lk}} \left(y_{li}-\theta_{i|j}\right)\left(y_{lm}-\theta_{m|j}\right) \qquad \text{(C.16)}$$

Finally we need to deal with the second term on the right hand side of C.10

$$\log P(\phi) = \text{const} + \alpha \left[\sum_{j=1}^{c} \log\left(p_j(\phi)\right) + \sum_{j=1}^{c} \sum_{i=1}^{n} \left(\log \theta_{i|j} + \log\left(1-\theta_{i|j}\right) \right) \right]$$

$$
\begin{aligned}
&= \text{const} + \alpha \left[\log \frac{1}{1 + \sum_{k=2}^{c} e^{\zeta_k}} + \sum_{j=2}^{c} \log \frac{e^{\zeta_j}}{1 + \sum_{k=2}^{c} e^{\zeta_k}} \right. \\
&\qquad \left. + \sum_{j=1}^{c} \sum_{i=1}^{n} \left(\log \left(\frac{e^{\phi_{i|j}}}{1 + e^{\phi_{i|j}}} \right) + \log \left(\frac{1}{1 + e^{\phi_{i|j}}} \right) \right) \right] \\
&= \text{const} + \alpha \sum_{j=2}^{c} \zeta_j - c\alpha \log \left((1 + \sum_{k=2}^{c} e^{\zeta_k} \right) \\
&\qquad + \alpha \sum_{j=1}^{c} \sum_{i=1}^{n} \left(\phi_{i|j} - 2 \log \left(1 + e^{\phi_{i|j}} \right) \right) \qquad \text{(C.17)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial \zeta_k} \log \left(p \left(\phi \right) \right) &= \frac{\partial}{\partial \zeta_k} \left(\text{const} + \alpha \sum_{j=2}^{c} \zeta_j - c\alpha \log \left((1 + \sum_{k=2}^{c} e^{\zeta_k} \right) \right. \\
&\qquad \left. + \alpha \sum_{j=1}^{c} \sum_{i=1}^{n} \left(\phi_{i|j} - 2 \log \left(1 + e^{\phi_{i|j}} \right) \right) \right) \\
&= \alpha - c\alpha \frac{e^{\zeta_j}}{1 + \sum_{h=2}^{c} e^{\zeta_j}} \\
&= \alpha - c\alpha p_k \left(\phi \right)
\end{aligned}
$$

$$
\frac{\partial}{\partial \phi_{i|j}} \log \left(p \left(\phi \right) \right) = \alpha - 2\alpha \theta_{i|j} \left(\phi \right)
$$

$$
\begin{aligned}
\frac{\partial^2}{\partial \zeta_k^2} \log \left(p \left(\phi \right) \right) &= \frac{\partial}{\partial \zeta_k} \left(\alpha - c\alpha p_k \left(\phi \right) \right) \\
&= -c\alpha \frac{\partial}{\partial \zeta_k} p_k \left(\phi \right) \\
&= -c\alpha p_k \left(1 - p_k \right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^2}{\partial \zeta_k \partial \phi_{i|j}} \log \left(p \left(\phi \right) \right) &= \frac{\partial}{\partial \theta_{i|j}} \left(\alpha - c\alpha p_k \left(\phi \right) \right) \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^2}{\partial \phi_{i|j}^2} \log\left(p\left(\boldsymbol{\phi}\right)\right) &= \frac{\partial}{\partial \theta_{i|j}} \left(\alpha - 2\alpha\theta_{i|j}\left(\boldsymbol{\phi}\right)\right) \\
&= -2\alpha\theta_{i|j}\left(\boldsymbol{\phi}\right)\left(1 - \theta_{i|j}\left(\boldsymbol{\phi}\right)\right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^2}{\partial \phi_{i|j} \partial \phi_{h|k}} \log\left(p\left(\boldsymbol{\phi}\right)\right) &= \frac{\partial}{\partial \theta_{h|k}} \left(\alpha - 2\alpha\theta_{i|j}\left(\boldsymbol{\phi}\right)\right) \\
&= 0
\end{aligned}
\tag{C.18}
$$

Appendix D

Derivation of Acceptance Probabilities for Reversible Jump MCMC

The acceptance probabilities for split, combine, birth and death moves are given by $\min(1, A_S)$, $\min(1, A_C)$, $\min(1, A_B)$ and $\min(1, A_D)$ respectively where A_S, A_C, A_B and A_D are derived as follows.

D.1 Split and Combine Moves

Recall that we defined our split move as follows:

1. Select a population j_* to be split.

2. Generate random numbers

$$u_i \sim B(1,1) \quad \text{and} \quad u_{2,i} \sim B(1,1) \quad i = 1, \ldots, n$$

3. Our split moves are then defined by

$$\begin{aligned} p_{j_1} &= u_1 p_{j_*} \quad , \quad p_{j_2} = (1-u_1)\, p_{j_*} \\ \theta_{i|j_1} &= \tfrac{u_{2,i}}{u_1}\theta_{i|j_*} \quad , \quad \theta_{i|j_2} = \tfrac{1-u_{2,i}}{1-u_1}\theta_{i|j_*}. \end{aligned} \tag{D.1}$$

4. Finally, we update the allocation variable. For each $z_i = j_*$ we reassign z_i according to

$$\begin{aligned} P(z_i = j_1) &\propto p_{j_1} f(y_i \mid \boldsymbol{\theta}_j) \\ P(z_i = j_2) &\propto p_{j_2} f(y_i \mid \boldsymbol{\theta}_j). \end{aligned}$$

From Green [31] A_S in (5.8) is given by

$$A_S = \frac{P(\psi' \mid y)\, r_m(\psi')}{P(\psi \mid y)\, r_m(\psi)\, q(u)} \left| \frac{\partial \psi'}{\partial(\psi, u)} \right|$$

where ψ is the current state, ψ' is the proposed higher dimensional state, $r_m(\psi)$ is the probability of attempting move type m when in state ψ, and $q(u)$ is the density function of u. It is easier to evaluate this expression if we use Bayes theorem to rewrite it as

$$A_S = \frac{P(y \mid \psi')\, P(\psi')\, r_m(\psi')}{P(y \mid \psi)\, P(\psi)\, r_m(\psi)\, q(u)} \left| \frac{\partial \psi'}{\partial(\psi, u)} \right| \tag{D.2}$$

Utilising the conditional independencies encoded in the DAG given in Figure 5.1, $P(y \mid \psi)\, P(\psi)$ is given by

$$P(y \mid \psi)\, P(\psi) = P(k \mid k_{max})\, P(\boldsymbol{\theta} \mid k, \delta_2)\, P(\boldsymbol{p} \mid k, \delta_1)\, P(\boldsymbol{z} \mid k, \boldsymbol{p})\, P(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{z})$$

For a naive Bayesian network these probabilities are given by

$$\begin{aligned}
P(\boldsymbol{\theta} \mid k, \delta_2) &= \prod_{i=1}^{n}\prod_{j=1}^{k} \frac{\Gamma(\delta_2)^2}{\Gamma(2\delta_2)} \theta_{i|j}^{\delta_2-1} \left(1-\theta_{i|j}\right)^{\delta_2-1} \\
P(\boldsymbol{p} \mid k, \delta_1) &= \frac{\Gamma(\delta_1)^k}{\Gamma(k\delta_1)} \prod_{j=1}^{k} p_j^{\delta_1-1} \\
P(z \mid k, \boldsymbol{p}) &= \prod_{j=1}^{k} p_j^{l_j} \\
P(y \mid \boldsymbol{\theta}, z) &= \prod_{h=1}^{N}\prod_{i=1}^{n} \theta_{i|z_h}^{y_{hi}} \left(1-\theta_{i|z_h}\right)^{(1-y_{hi})}.
\end{aligned} \tag{D.3}$$

Hence for a naive Bayesian network

$$\begin{aligned}
P(\psi) &= P(k \mid k_{max}) \times \prod_{i=1}^{n}\prod_{j=1}^{k} \frac{\Gamma(\delta_2)^2}{\Gamma(2\delta_2)} \theta_{i|j}^{\delta_2-1} \left(1-\theta_{i|j}\right)^{\delta_2-1} \\
&\quad \times \frac{\Gamma(\delta_1)^k}{\Gamma(k\delta_1)} \prod_{j=1}^{k} p_j^{\delta_1-1} \times \prod_{j=1}^{k} p_j^{l_j},
\end{aligned} \tag{D.4}$$

and it follows that

$$\begin{aligned}
\frac{P(\psi' \mid y)}{P(\psi \mid y)} &= \frac{P(y \mid \psi')\, P(\psi')}{P(y \mid \psi)\, P(\psi)} \\
&= \frac{P(y \mid z', \theta')}{P(y \mid z, \theta)} \times \frac{P(k+1)}{P(k)} \times \frac{\Gamma((k+1)\,\delta_1)}{\Gamma(\delta_1)\,\Gamma(k\delta_1)} \frac{p_{j_1}^{\delta_1-1+l_1} p_{j_2}^{\delta_1-1+l_2}}{p_{j_*}^{\delta_1-1+l_1+l_2}} \\
&\quad \times \prod_{i=1}^{n} \frac{\Gamma(2\delta_2)}{\Gamma(\delta_2)^2} \frac{\theta_{i|j_1}^{\delta_2-1} \left(1-\theta_{i|j_1}\right)^{\delta_2-1} \theta_{i|j_2}^{\delta_2-1} \left(1-\theta_{i|j_2}\right)^{\delta_2-1}}{\theta_{i|j_*}^{\delta_2-1} \left(1-\theta_{i|j_*}\right)^{\delta_2-1}}.
\end{aligned} \tag{D.5}$$

The ratio $r_m(\psi') \Big/ r_m(\psi)$ is simply given by

$$\begin{aligned} \frac{r_m(\psi')}{r_m(\psi)} &= \frac{P(\text{attempt a combine move when in state } \psi')}{P(\text{attempt a split move when in state } \psi)} \\ &= \frac{b_{k-1} P_{\text{alloc}}}{d_k}, \end{aligned} \tag{D.6}$$

where P_{alloc} is the probability of the allocation being proposed and is given by

$$\begin{aligned} P_{\text{alloc}} = \prod_{h:z'_h \in \{j_1, j_2\}} & \left(p_{z_h} \prod_{i=1}^{n} \theta_{i|z'_h}^{y_{hi}} \left(1 - \theta_{i|z'_h}\right)^{(1-y_{hi})} \right. \\ & \left. \Big/ \left(p_{j_1} \prod_{i=1}^{n} \theta_{i|j_1}^{y_{hi}} \left(1 - \theta_{i|j_1}\right)^{(1-y_{hi})} + p_{j_2} \prod_{i=1}^{n} \theta_{i|j_2}^{y_{hi}} \left(1 - \theta_{i|j_2}\right)^{(1-y_{hi})} \right) \right). \end{aligned} \tag{D.7}$$

The term $q(u)^{-1}$ is simply given by

$$q(u)^{-1} = \left(g_{1,1}(u_1) \prod_{i=1}^{n} g_{1,1}(u_{2,i}) \right)^{-1}. \tag{D.8}$$

Finally we have to calculate the determinant of the Jacobian of the transformation between the two parameter spaces. From equations (D.1) the required differentials are

$$
\begin{aligned}
\frac{\partial p_{j_1}}{\partial p_{j_*}} &= u_1 & \frac{\partial p_{j_2}}{\partial p_{j_*}} &= 1-u_1 \\
\frac{\partial p_{j_1}}{\partial u_1} &= p_{j_*} & \frac{\partial p_{j_2}}{\partial u_1} &= -p_{j_*} \\
\frac{\partial \theta_{i|j_1}}{\partial \theta_{i|j_*}} &= \frac{u_{2,i}}{u_1} & \frac{\partial \theta_{i|j_2}}{\partial \theta_{i|j_*}} &= \frac{1-u_{2,i}}{1-u_1} \\
\frac{\partial \theta_{i|j_1}}{\partial u_{2,i}} &= \frac{\theta_{i|j_*}}{u_1} & \frac{\partial \theta_{i|j_2}}{\partial u_{2,i}} &= -\frac{\theta_{i|j_*}}{1-u_1} \\
\frac{\partial \theta_{i|j_1}}{\partial u_1} &= -\frac{u_{2,i}\theta_{i|j_*}}{u_1^2} & \frac{\partial \theta_{i|j_2}}{\partial u_1} &= \frac{(1-u_{2,i})\,\theta_{i|j_*}}{(1-u_1)^2}
\end{aligned}
$$

with all other partial derivatives equal to zero. Hence we can see that

$$
\begin{aligned}
\left|\frac{\partial \psi'}{\partial (\psi, u)}\right| &= \begin{vmatrix}
u_1 & 1-u_1 & 0 & 0 & \cdots & 0 & 0 \\
p_* & -p_* & -\frac{u_{2,1}\theta_{1|j_*}}{u_1^2} & \frac{(1-u_{2,1})\theta_{1|j_*}}{(1-u_1)^2} & \cdots & -\frac{u_{2,n}\theta_{n|j_*}}{u_1^2} & \frac{(1-u_{2,n})\theta_{n|j_*}}{(1-u_1)^2} \\
0 & 0 & \frac{u_{2,1}}{u_1} & \frac{(1-u_{2,1})}{(1-u_1)} & \cdots & 0 & 0 \\
0 & 0 & \frac{\theta_{1|j_*}}{u_1} & \frac{-\theta_{1|j_*}}{(1-u_1)} & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & \frac{u_{2,n}}{u_1} & \frac{(1-u_{2,n})}{(1-u_1)} \\
0 & 0 & 0 & 0 & \cdots & \frac{\theta_{n|j_*}}{u_1} & \frac{-\theta_{n|j_*}}{(1-u_1)}
\end{vmatrix} \\
&= \begin{vmatrix} u_1 & 1-u_1 \\ p_* & -p_* \end{vmatrix} \times \begin{vmatrix} \frac{u_{2,1}}{u_1} & \frac{(1-u_{2,1})}{(1-u_1)} \\ \frac{\theta_{1|j_*}}{u_1} & \frac{-\theta_{1|j_*}}{(1-u_1)} \end{vmatrix} \times \ldots \times \begin{vmatrix} \frac{u_{2,n}}{u_1} & \frac{(1-u_{2,n})}{(1-u_1)} \\ \frac{\theta_{n|j_*}}{u_1} & \frac{-\theta_{n|j_*}}{(1-u_1)} \end{vmatrix} \\
&= p_{j_*} \prod_{i=1}^{n} \frac{\theta_{i|j_*}}{u_1 (1-u_1)} \qquad \text{(D.9)}
\end{aligned}
$$

Substituting D.5 - D.9 into D.2 gives

$$\begin{aligned} A_S &= \frac{P(y \mid z', \theta')}{P(y \mid z, \theta)} \times \frac{P(k+1)}{P(k)} \times \frac{\Gamma((k+1)\delta_1)}{\Gamma(\delta_1)\Gamma(k\delta_1)} \frac{p_{j_1}^{\delta_1-1+l_1} p_{j_2}^{\delta_1-1+l_2}}{p_{j_*}^{\delta_1-1+l_1+l_2}} \\ &\times \prod_{i=1}^{n} \frac{\Gamma(2\delta_2)}{\Gamma(\delta_2)^2} \frac{\theta_{i|j_1}^{\delta_2-1}\left(1-\theta_{i|j_1}\right)^{\delta_2-1} \theta_{i|j_2}^{\delta_2-1}\left(1-\theta_{i|j_2}\right)^{\delta_2-1}}{\theta_{i|j_*}^{\delta_2-1}\left(1-\theta_{i|j_*}\right)^{\delta_2-1}} \\ &\times \frac{d_{k+1}}{b_k P_{\text{alloc}}} \left(g_{1,1}(u_1) \prod_{i=1}^{n} g_{1,1}(u_{2,i}) \right)^{-1} \\ &\times p_{j_*} \prod_{i=1}^{n} \frac{\theta_{i|j_*}}{u_1(1-u_1)}. \end{aligned} \tag{D.10}$$

A_C is derived similarly and is simply the inverse of A_S with a couple of obvious substitutions.

D.2 Birth and Death Moves

Recall that we define a birth move by

$$\begin{aligned} p_{j_*} &\sim B(1,k) \\ \theta_{i|j_*} &\sim B(\delta_2,\delta_2) \qquad i=1,\ldots,n. \end{aligned}$$

We then have to re-scale the existing weights $p_1,\ldots,p_k$ by multiplying them by $(1-p_{j_*})$ in order that our new weights will sum to 1. No other change is made; in particular the allocation variables z_i are left unaltered. As before we need to evaluate

$$\begin{aligned} A_B &= \frac{P(\psi' \mid y)\, r_m(\psi')}{P(\psi \mid y)\, r_m(\psi)\, q(u)} \left| \frac{\partial \psi'}{\partial(\psi, u)} \right| \\ &= \frac{P(y \mid \psi')\, P(\psi')\, r_m(\psi')}{P(y \mid \psi)\, P(\psi)\, r_m(\psi)\, q(u)} \left| \frac{\partial \psi'}{\partial(\psi, u)} \right| \end{aligned} \tag{D.11}$$

First we consider

$$\frac{P(y \mid \psi')\,P(\psi')}{P(y \mid \psi)\,P(\psi)} = \frac{P(k+1)}{P(k)}\frac{P(\theta' \mid k+1, \delta_2)}{P(\theta \mid k, \delta_2)}\frac{P(p' \mid k+1, \delta_1)}{P(p \mid k, \delta_1)} \times \frac{P(z' \mid k+1, p')}{P(z \mid k, p)}\frac{P(y \mid \theta', z')}{P(y \mid \theta, z)} \tag{D.12}$$

Since we haven't updated the allocation variables, $z = z'$ and hence

$$\frac{P(y \mid \theta', z')}{P(y \mid \theta, z)} = 1.$$

We can also able to simplify other expressions on the right hand side of Equation (D.12)

$$\begin{aligned}
\frac{P(\theta' \mid k+1, \delta_2)}{P(\theta \mid k, \delta_2)} &= \left(\prod_{j=1}^{k}\prod_{i=1}^{n}\frac{\Gamma(2\delta_2)}{\Gamma(\delta_2)^2}\theta_{i|j_*}^{\delta_2-1}\left(1-\theta_{i|j_*}\right)^{\delta_2-1}\prod_{i=1}^{n}\frac{\Gamma(2\delta_2)}{\Gamma(\delta_2)^2}\theta_{i|j_*}^{\delta_2-1}\left(1-\theta_{i|j_*}\right)^{\delta_2-1}\right) \\
&\quad \Big/ \left(\prod_{j=1}^{k}\prod_{i=1}^{n}\frac{\Gamma(2\delta_2)}{\Gamma(\delta_2)^2}\theta_{i|j_*}^{\delta_2-1}\left(1-\theta_{i|j_*}\right)^{\delta_2-1}\right) \\
&= \prod_{i=1}^{n}\frac{\Gamma(2\delta_2)}{\Gamma(\delta_2)^2}\theta_{i|j_*}^{\delta_2-1}\left(1-\theta_{i|j_*}\right)^{\delta_2-1} \\
&= \prod_{i=1}^{n} g_{\delta_1,\delta_1}\left(\theta_{i|j_*}\right) \qquad \text{(D.13)}
\end{aligned}$$

$$\begin{aligned}
\frac{P(p' \mid k+1, \delta_1)}{P(p \mid k, \delta_1)} &= \left(\frac{\Gamma((k+1)\,\delta_1)}{\Gamma(\delta_1)^{k+1}}\prod_{j=1}^{k}\left(p_j\left(1-p_{j_*}\right)\right)^{d_1-1}p_{j_*}^{d_1-1}\right) \Big/ \left(\frac{\Gamma((k)\,\delta_1)}{\Gamma(\delta_1)^{k}}\prod_{j=1}^{k}p_j^{d_1-1}\right) \\
&= \frac{\Gamma((k+1)\,\delta_1)}{\Gamma(\delta_1)\,\Gamma(k\delta_1)}p_{j_*}^{\delta_1-1}\left(1-p_{j_*}\right)^{k(\delta_1-1)} \qquad \text{(D.14)}
\end{aligned}$$

$$\begin{aligned}
\frac{P(z' \mid k+1, p')}{P(z \mid k, p)} &= \left(\prod_{j=1}^{k}\left(p_j\left(1-p_{j_*}\right)\right)^{l_j}\right) \Big/ \left(\prod_{j=1}^{k}p_j^{l_j}\right) \\
&= \left(1-p_{j_*}\right)^{\sum_{j=1}^{k} l_j} \\
&= \left(1-p_{j_*}\right)^{N} \qquad \text{(D.15)}
\end{aligned}$$

The ratio $\frac{r_m(\psi')}{r_m(\psi)}$ is simplified by the fact that no update of the allocation variables has taken place and is given by

$$\frac{r_m(\psi')}{r_m(\psi)} = \frac{b_{k-1}}{d_k}$$

$q(u)^{-1}$ is given by

$$\left(g_{1,k}(p_{j\bullet}) \prod_{i=1}^{n} g_{\delta_2,\delta_2}(\theta_{i|j\bullet}) \right)^{-1}$$

and finally, the Jacobian is given by

$$\left| \frac{\partial \psi'}{\partial(\psi, u)} \right| = \begin{vmatrix} 1-p_{j\bullet} & 0 & \cdots & 0 & p_1 \\ 0 & 1-p_{j\bullet} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & 1-p_{j\bullet} & p_{k-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{vmatrix}$$

$$= (1-p_{j\bullet})^{k-1} \qquad \text{(D.16)}$$

Hence substituting D.13, - D.2 into D.11 gives the following expression for A_B:

$$\begin{aligned} \frac{P(\psi' \mid y)}{P(\psi \mid y)} &= \frac{P(k+1)}{P(k)} \prod_{i=1}^{n} g_{\delta_2,\delta_2}(\theta_{i|j\bullet}) \frac{\Gamma((k+1)\delta_1)}{\Gamma(\delta_1)\Gamma(k\delta_1)} p_{j\bullet}^{\delta_1-1} (1-p_{j\bullet})^{k(\delta_1-1)} \\ &\quad \times (1-p_{j\bullet})^{N} \frac{b_{k-1}}{d_k} \left(g_{1,k}(p_{j\bullet}) \prod_{i=1}^{n} g_{\delta_2,\delta_2}(\theta_{i|j\bullet}) \right)^{-1} (1-p_{j\bullet})^{k-1} \\ &= \frac{P(k+1)}{P(k)} \frac{\Gamma((k+1)\delta_1)}{\Gamma(\delta_1)\Gamma(k\delta_1)} p_{j\bullet}^{\delta_1-1} (1-p_{j\bullet})^{N+k(\delta_1-1)} \\ &\quad \times \frac{b_{k-1}}{d_k} \frac{\Gamma(1)\Gamma(K)}{\Gamma(k+1)} \frac{(1-p_{j\bullet})^{k-1}}{(1-p_{j\bullet})^{k-1}} \\ &= \frac{P(k+1)}{P(k)} \frac{\Gamma((k+1)\delta_1)}{\Gamma(\delta_1)\Gamma(k\delta_1)} p_{j\bullet}^{\delta_1-1} (1-p_{j\bullet})^{N+k(\delta_1-1)} \frac{b_{k-1}}{d_k} \frac{1}{k} \end{aligned}$$

(D.17)

A_D is the inverse of A_B but with a few obvious substitutions.

Bibliography

[1] H. Akaike. A new look at statistical model selection. *IEEE Transactions on Automatic Control*, 19 6:716–723, 1974.

[2] A. Baddeley. Errors in binary images and a L^p version of the Hausdorff metric. *Nieuw Archief voor Wiskunde*, 10:157–183, 1992.

[3] C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated classification likelihood. Technical Report 3521, INRIA, 655 Avenue de l'Europe, 38330 Montbonnot St Martin, France, 1998.

[4] J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29 2-3:213–244, 1997.

[5] C.M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, UK, 1995.

[6] C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1998.

[7] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8 2:195–210, 1996.

[8] E.A. Catchpole. Solving problems in parameter redundancy using computer algebra. Technical report, School of Mathematics and Statistics, University College UNSW, Australian Defence Force Academy, Canberra, Australia, 1999.

[9] E.A. Catchpole and B.J.T. Morgan. Detecting parameter redundancy. *Biometrika*, 84:187–196, 1997.

[10] E.A. Catchpole, B.J.T. Morgan, and S.N. Freeman. Estimation in parameter-redundant models. *Biometrika*, 85 2:462–468, 1998.

[11] G. Celeux, M. Hurn, and C.P. Robert. Computational and inferential difficulties with mixture posterior distributions. Technical Report 9914, INSEE, France, 1999.

[12] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In U.M. Fayyad, G. Plateetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in knowledge discovery & data mining*, pages 153–180, Menlo Park, CA., 1995. AAAI Press.

[13] B. Cheng and D.M. Titterington. Neural networks: a review from a statistical perspective (with discussion). *Statistical Science*, 9 1:2–54, 1994.

[14] S. Chib. Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90:1313–1321, 1995.

[15] D.M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of incomplete data given a Bayesian network. Technical Report MSR-TR-96-08, Microsoft, Redmond, WA, 1996.

[16] D.M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29:181–212, 1997.

[17] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[18] S. Crawford. An application of the Laplace method to finite mixture distributions. *Journal of the American Statistical Association*, 89:259–267, 1994.

[19] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society*, B 39:1–38, 1977.

[20] J. Diebolt and C. Robert. Estimation of finite mixture distributions through Bayesian sampling. *Journal of the Royal Statistical Society*, B 56:163–75, 1994.

[21] D. Draper. Assesment and propagation of model uncertainty (with discussion). *Journal of the Royal Statistical Society B*, 57:45–97, 1995.

[22] A. Erkanli. Laplace approximations for posterior expectations when the mode occurs at the boundary of the parameter space. *Journal of the American Statistical Association*, 89:250–258, 1994.

[23] R. Fletcher and C.M. Reeves. Function minimisation by conjugate gradients. *Computer Journal*, 6:149–154, 1964.

[24] D. Geiger, D. Heckerman, and C. Meek. Asymptotic model selection for directed networks with hidden variables. Technical Report MSR-TR-96-07, Microsoft, Redmond, WA, 1996.

[25] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–742, 1984.

[26] C.J. Geyer. Estimating normalizing constants and reweighting mixtures in Markov chain Monte Carlo. Technical Report 568, School of Statistics, University of Minnesota, 1994.

[27] C.J. Geyer and E.A. Thompson. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90, 431:909–920, 1995.

[28] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov Chain Monte Carlo in Practice.* Chapman & Hall, London, UK, 1996.

[29] L.A. Goodman. Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61 2:215–231, 1974.

[30] P.J. Green. On use of the EM algorithm for penalized likelihood estimation. *Journal of the Royal Statistical Society B*, 52:443–452, 1990.

[31] P.J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–32, 1995.

[32] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft, Redmond, WA, 1995.

[33] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[34] D.P. Helmbold, R.E. Shapire, and Y.Singer M.K. Warmuth. A comparison of new and old algorithms for a mixture estimation problem. *Machine Learning*, 27 1:97–119, 1997.

[35] D.W. Hosmar and S. Lemeshowe. *Applied Logistic Regression.* Wiley, New York, USA, 1989.

[36] H.Teicher. Identifiability of mixtures. *Annals of Mathematical Statistics*, 32:244–248, 1961.

[37] M. Jamshidian and R.I. Jennrich. Conjugate gradient acceleration of the EM algorithm. *Journal of the American Statistical Association*, 88:221–228, 1993.

[38] M. Jamshidian and R.I. Jennrich. Acceleration of the EM algorithm by using quasi-Newton methods. *Journal of the Royal Statistical Society B*, 59 3:569–587, 1997.

[39] F.V. Jensen. *An Introduction to Bayesian Networks.* UCL Press, University College London, Gower street, London, UK, 1996.

[40] P. Kontkanen, P. Myllymaki, T. Silander, and H. Tirri. A Bayesian approach to discretization. In *Proceedings of the European Symposium on Intelligent Techniques (Bari, Italy, March 1997)*, pages 265–268, 1997.

[41] P. Kontkanen, P. Myllymaki, and H. Tirri. Comparing Bayesian model class selection criteria by discrete finite mixtures. In D.L. Dowe, K.B. Korb, and J.J. Oliver, editors, *Information, Statistics and Induction in Science (Proceedings of the ISIS'96 Conference, Melbourne, Australia, August 1996)*, 1996.

[42] P. Kontkanen, P. Myllymaki, and H. Tirri. Constructing Bayesian finite mixture models by the EM algorithm. Technical Report NC-TR-97-003, NeuroCOLT, Royal Holloway University of London, Egham, Surrey TW20 0EX, UK, 1997.

[43] D. Mackay. Choice of basis for the Laplace approximation. *Machine Learning*, 33:77–86, 1998.

[44] D. Madigan and A.E. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. Technical report, Department of Statistics, University of Washington, 1993.

[45] D. Madigan and A.E. Raftery. Strategies for graphical model selection. Technical report, Department of Statistics, University of Washington, 1994.

[46] E. Marinari and G. Parisi. Simulated tempering : a new Monte Carlo scheme. *Europhysics Letters*, 19:451–458, 1983.

[47] J. Martin and K. Vanlehn. Student assessment using Bayesian nets. *International Journal of Human-Computer Studies*, 42 6:575–591, 1995.

[48] X.L. Meng and D. van Dyk. The EM algorithm - an old folk-song to a fast new tune (with discussion). *Journal of the Royal Statistical Society B*, 59 3:511–567, 1997.

[49] R.M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

[50] R.M. Neal. Bayesian mixture modeling by Monte Carlo simulation. Technical Report CRG-TR-91-2, Department of Computer Science, University of Toronto, 1996.

[51] R.M. Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6 4:353–366, 1996.

[52] R.M. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in Graphical Models (proceedings of the NATO Advanced Study Institute on Learning in Graphical Models, Erice, Italy, 1996)*, pages 355–368. Kluwer Academic Press, 1996.

[53] S.J. Nowlan. *Soft Competitive Adaption: Neural Network Learning Algorithms based on Fitting Statistical Mixtures.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh., 1991.

[54] B.C. Peters and H.F. Walker. An iterative procedure for obtaining maximum-likelihood estimates of the parameters for a mixture of normal distributions. *Siam Journal of Applied Mathematics*, 35:362–378, 1978.

[55] J. Phillips. *The NAG Library: A Beginners Guide.* Clarendon Press, Oxford, UK, 1986.

[56] P.J. Plauger. *The Standard C Library.* Prentice-Hall, Eaglewood Cliffs, New Jersey, USA, 1992.

[57] A.E. Raftery. *Hypothesis Testing and Model Selection*, chapter 10. Chapman and Hall, 1996.

[58] M. Ramoni and P. Sebastiani. Efficient parameter learning in bayesian networks from incomplete data. Technical Report KMi-TR-41, Knowledge Media Institute, The Open University, Milton Keynes MK7 6AA, UK, 1997.

[59] S. Richardson and P.J. Green. On Bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society B*, 59 4:731–758, 1997.

[60] S. Richardson and P.J. Green. Corrigendum: On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society B*, 60 3:U3, 1998.

[61] B.D. Ripley. *Stochastic Simulation.* John Wiley & Sons, New York, USA, 1987.

[62] J. Rissanen. Stochastic complexity (with discussion). *Journal of the Royal Statistical Society B*, 49:223–239 and 253–265, 1987.

[63] C.P. Robert and K.L. Mengersen. Reparameterisation issues in mixture modelling and their bearing on the Gibbs sampler. *Computational Statistics and Data Analysis*, 29:325–343, 1999.

[64] H. Rue. New loss functions in Bayesian imaging. *Journal of the American Statistical Association*, 90:900–908, 1995.

[65] S. Russell, J. Binder, and D. Koller. Adaptive probabilistic networks. Technical Report UCB/CSD-94-824, Computer Science Division, University of California, Berkeley, CA 94720, 1994.

[66] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6 2:461–464, 1978.

[67] Y. Singer and M.K. Warmuth. Training algorithms for hidden markov models using entropy based distance functions. In J.D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 9 (NIPS '96)*, pages 641–647. Morgan Kaufmann, 1996.

[68] B. Theisson. Accelerated quantification of Bayesian networks with incomplete data. In U.M. Fayyad and R. Uthurusamy, editors, *First International Conference on Knowledge Discovery and Data Mining*, pages 306–311. AIII Press, 1996.

[69] B. Thiesson. Score and information for recursive exponential models with incomplete data. Technical report, Institute of Electronic Systems, Aalborg University, Aalborg, Denmark, 1995.

[70] R.A. Thisted. *Elements of Statistical Computing.* Chapman & Hall, New York, USA, 1988.

[71] D.M. Titterington, G.D. Murray, L.S. Murray, D.J. Spiegelhalter, A.M. Skene, J.D.F. Habbema, and G.J. Gelpka. Comparison of discrimination techniques applied to a complex dataset of head injured patients. *Journal of the Royal Statistical Society, A*, 144:145–175, 1981.

[72] N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11:271–282, 1998.

[73] J.H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation*, volume II. Springer-Verlag, Heidelberg, Germany, 1971.

[74] A. Zellner and M. Chung-ki. Gibbs sampler convergence criteria. *Journal of the American Statistical Association*, 90:921–927, 1995.