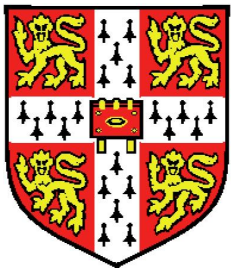
Bayesian non-parametric models and inference for sparse and hierarchical latent structure



David Arthur Knowles

Wolfson College

University of Cambridge

A thesis submitted for the degree of

Doctor of Philosophy

April 2012

I, DAVID ARTHUR KNOWLES, confirm that *this dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.* Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

I also confirm that this thesis is below 60,000 words and contains less than 150 figures, in fulfilment of the requirements set by the degree committee for the Department of Engineering at the University of Cambridge.

Acknowledgements

I would firstly like to thank my supervisor, Prof. Zoubin Ghahramani. Zoubin is a great supervisor and mentor because he is the antithesis of the XKCD stereotype of the academic professor. Where the stereotype is uncaring, uninterested and absent, Zoubin genuinely cares about the well-being, development and success of his students.

The list of amazing lab members I have had the privilege of interacting and collaborating with is really too long to list by name, a testament in itself to how friendly our lab is, but I would like to specifically mention Jurgen Van Gael, Finale Doshi-Velez, Shakir Mohamed, Ryan Turner, Sinead Williamson, Peter Orbanz, Simon Lacoste-Julien, Andrew Wilson, Konstantina Palla, and Alex Davies.

During my time in Cambridge I have also had the opportunity to collaborate extensively with John Winn, Tom Minka and John Guiver at Microsoft Research Cambridge. Their guidance and discussion has been been invaluable.

Despite being a proud member of Wolfson College Boat Club, my closest friendships during my PhD have been with members of Corpus Christi college, particularly Richard Pymar, Chris Bowman, Alison Knight, Susannah Gibson, Luke Gardiner, Rob Payne, Megan Cavell, James Brown and Elena Kazamia.

Abstract

Increasingly complex and plentiful data is a modern trend. In biology for example high throughput experimental methods such as gene expression microarray, SNP arrays and DNA sequencing are becoming the norm. However, extracting meaningful, interpretable knowledge and understanding from these data sources remains a challenge. In this thesis we contribute two Bayesian nonparametric models for discovering latent structure in data. The first, a non-parametric, sparse extension of factor analysis, explains observed data as a sparse superposition of an unbounded number of latent processes. The structure discovered is a bipartite graph connecting observed and latent variables. The second model, the Pitman Yor diffusion tree, is able to learn a latent hierarchy where the leaves of the tree correspond to observed entities. Unlike its predecessor, the Dirichlet diffusion tree, the model has support for arbitrary branching structure rather than just binary branching, allowing more interpretable solutions to be found.

Rich, expressive models are clearly desirable, but to be useful we must be able to handle them computationally and learn parameters in a reasonable amount of time. Efficient approximate message passing and variational methods are reviewed, and a new algorithm is proposed which extends a particular method, variational message passing (VMP), to a much greater range of probabilistic models. We apply this method, which we call non-conjugate VMP, to multinomial logistic regression and heteroskedastic regression. Finally we show how message passing methods can be used as part of an efficient greedy algorithm to find optimal tree structures under the Pitman Yor diffusion tree introduced previously.

Contents

List of Figures

Chapter 1

Introduction

Statistical inference in principle is simple. Write down your probabilistic model of the world, condition on all available observations and Bayes rule tells you what your belief about the state of all unknowns should be. In reality of course there are many obstacles to this approach: how can we ever hope to formalise our intuition about the world as a well defined mathematical model? Even if we knew the model, could we handle it computationally? Thus we settle for compromises: choose a simple model which we can understand, and which is tractable using modern computer algorithms and hardware. But we should not forget that this compromise has been made, and that progress is made if we can move a little closer to modeling the complexity of the real world.

Much work in machine learning and statistics can be viewed as attempting to do this. More flexible, expressive models, with complex latent structure, can better approximate our complex world and better incorporate rich prior knowledge. One such class of models use Bayesian non-parametric priors [Orbanz & Teh, 2010; Teh & Jordan, 2009]. While flexible, adaptive models are clearly desirable in principle, they are of little practical use unless we are able to handle them computationally. The work in this thesis contributes to both these threads: introducing new non-parametric models and developing methodology to make such complex models tractable.

This thesis describes two novel Bayesian non-parametric models. The first, in Chapter 3, is a factor analysis model, where observed data are explained as sparse linear combinations of latent continuous-valued processes or "factors". In

contrast to previous approaches to this problem, the model is naturally able to adapt the number of latent factors used to best explain the observed data. The second, in Chapter 4, is a Bayesian non-parametric prior over hierarchies with arbitrary branching structure. We show this process has attractive properties for modelling, and can be used to learn interpretable hierarchies over objects.

Chapter 5 reviews message passing techniques. Contributing to the line of work expanding the range of models amenable to statistical inference, Chapter 6 introduce a novel message passing algorithm which significantly extends the class of models that can be learnt using a particular method, known as variational message passing.

Chapter 7 combines these lines of research, demonstrating a message passing inference algorithm for models based on the prior over trees introduced in Chapter 4.

To demonstrate the practical value of these methods we apply them to various real world datasets, particularly focusing on problems in computational biology. We show that our non-parametric sparse factor analysis provides an excellent statistical model of the variation typical in gene expression data, while also finding interpretable latent structure. The message passing techniques partially developed here and implemented in Infer.NET [Minka *et al.*, 2010] allow us to easily model rich phenotypic data [Knowles *et al.*, 2010, 2011b].

1.1 Bayesian non-parametrics

Statistical models typically have "parameters": variables which can be thought as indexing the available models within a particular class. In the Bayesian approach, parameters are themselves viewed as random variables which should therefore have some prior distribution. The model specifies a distribution over observed data which is conditional on these parameters. Traditional models have a fixed, finite number of parameters, and are known as "parametric models". A Gaussian distribution with unknown mean and variance is perhaps the most canonical example. However, this is quite a strong restriction: if the data does not really come from a Gaussian, even in the limit of infinite data we cannot recover the true density. Instead of a single Gaussian we could use a mixture of Gaussians

for example. The more mixture components the more flexible our model will be, which might suggest choosing a large number of mixture components. Traditional statistics might discourage this idea because of the worry of overfitting: there might be "insufficient" data to get good estimates of the many parameters in the model. However, if we are able to perform full Bayesian inference, then we are protected from overfitting because any parameters that are not needed to explain the observed data are effectively marginalised out. The logical limit of this idea is to use a model with (countably or uncountably) infinitely many parameters, for example using a mixture model with a countably infinite number of components. With suitably structured models we find that when performing inference we only need represent a finite number of parameters, since all but finitely many parameters do not effect the data. One such model with this property in the class of infinite mixture models is the Dirichlet process mixture model [Antoniak, 1974; Escobar & West, 1995; Rasmussen, 2000]. The model we introduce in Chapter 4 can be seen as a generalisation of a Dirichlet process mixture model which incorporates additional latent structure while maintaining desirable statistical properties, in particular exchangeability and projectivity.

Exchangeable sequences. A sequence of random variables $X_1, \cdots, X_n$ is said to be exchangeable if the joint distribution is invariant to permutations, i.e. $\mathbb{P}(X_1, \cdots, X_n) = \mathbb{P}(X_{\sigma(1)}, \cdots, X_{\sigma(n)})$ for all finite permutations σ. When we have a infinite sequence of random variables $X_1, X_2, \cdots$ we are usually interested in whether the sequence is *infinitely* exchangeable, i.e. is invariant to permutations of any finite subsequence. Intuitively exchangeability corresponds to the assumption that the order of the data points does not matter. Exchangeability is not only a key modelling assumption, but also a valuable property allowing considerably simpler analysis and inference. For example, a common way to perform Markov chain Monte Carlo sampling in an exchangeable model is to consider the current data point to be the final point sampled in the generative process, which is possible because of exchangeability. This simplifies the update because we do not need to consider the effect of the state of the current data point on the probability of subsequent dat points.

De Finetti's Theorem. Through de Finetti's theorem [De Finetti, 1931] the idea of exchangeability provides strong motivation for hierarchical probabilistic models. The theorem states the distribution of any infinitely exchangeable sequence $X_1, X_2, ...$ can be represented as

$$\mathbb{P}(X_1, X_2, X_3, ...) = \int \prod_i \theta(X_i) d\nu(\theta), \tag{1.1}$$

where θ is a probability measure and ν is known as the de Finetti measure, which we associate with a prior in Bayesian modelling.

Projectivity. Consider the family of models $\{\mathbb{P}_n : n = 1, 2, ...\}$, where $\mathbb{P}_n$ is the model (joint distribution) for n data points. If the joint distribution over $n-1$ data points is given by marginalising out the n-th data point from the joint distribution over n data points, i.e.

$$\mathbb{P}_{n-1}(X_1, \cdots, X_{n-1}) = \int \mathbb{P}_n(X_1, \cdots, X_{n-1}, X_n) dX_n, \tag{1.2}$$

then the family of models is said to be projective. This property satisfies two desiderata: firstly that knowing extra unobserved data points exist does not affect our prior over the observed data points, and secondly that the model can be consistently extended to new data, for example in a training/test split.

Most of the models used in the Bayesian non-parametrics literature demonstrate exchangeability and projectivity. When the assumption of exchangeability is undesirable it is often straightforward to extend such models, for example allowing Markov [Beal *et al.*, 2002] or tree structured [Miller *et al.*, 2008; Rai & Daumé III, 2008] dependencies.

1.2 Message passing algorithms

While Markov chain Monte Carlo [Neal, 1993] methods continue to be the standard tool for performing inference in probabilistic models, a different class of algorithms has been developing, perhaps described best as "deterministic methods" to contrast the stochastic nature of MCMC. These methods generally specify

a functional form of an approximate posterior distribution, and then attempt to make this approximation "close" to the true posterior in some way. We will focus in particular on a subset of deterministic methods which can be described using a message passing formalism.

The canonical message passing algorithm is belief propagation (BP), introduced in Pearl [1982], making exact inference tractable in large tree structured Bayesian networks. For cyclic graphs exact inference can be performed using the junction tree algorithm [Jensen *et al.*, 1994; Lauritzen & Spiegelhalter, 1988], which collapses cliques in the graph to give a tree structure. Unfortunately this method has computational cost exponential in the size of the largest clique. A solution originally proposed as a heuristic is to simply iterate standard BP on cyclic graphs, an algorithm known as "loopy BP" [Frey & MacKay, 1998; Gallager, 1963]. Loopy BP was found to have excellent empirical performance on certain decoding problems [Berrou *et al.*, 1993], and is now understood to have fixed points at stationary points of the Bethe free energy [Yedidia *et al.*, 2003], an approximation of the correct Gibbs free energy. Although BP is of great use for inference in discrete and Gaussian Bayesian networks (and many undirected models) it is not generally feasible for more complex models including non-linearity, non-conjugacy, and mixed discrete and continuous latent variables. Expectation propagation [Minka, 2001b] generalises BP to handle many of these situations by "projecting" the complex distributions BP would require sending onto simpler exponential family distributions which are "close" in terms of KL divergence.

Another group of deterministic methods are known as variational methods. Variational Bayes [Attias, 2000; Beal & Ghahramani, 2006] for example constructs and optimises a lower bound on the marginal likelihood of the model, which can be interpreted as minimising the KL divergence between the true and approximate posterior. The variational Bayes (VB) updates can be represented as local message passing on the factor graph, an implementation known as variational message passing [Winn & Bishop, 2006]. Variational message passing (VMP) and expectation propagation (EP) can be seen to be specific instances of a more general message passing algorithm based on α-divergences [Minka, 2005]. Chapter 5 provides a practical review of these methods.

There are many discussions of the relative merits of MCMC and determin-

istic methods for specific models [Blei & Jordan, 2006; Mohamed *et al.*, 2011; Nickisch & Rasmussen, 2008; Rattray *et al.*, 2009; Turner *et al.*, 2008] and some more general reviews [Ghahramani, 2006], but for completeness we summarise some of the key factors here. A primary motivation of MCMC is that under mild and easily verifiable conditions the sampler will eventually sample from the true equilibrium distribution. This is a valuable guarantee, and has practical implications as well: if I need more accuracy, I can simply try running my sampler for longer. Deterministic methods offer no such promise that even with infinite computational resource they will be able to get close to the true posterior. However in practice, MCMC's guarantees are often vacuous. It may take an exponentially long time to explore the posterior, and moving from one mode to another may be almost impossible. The mixing time of the chain may be poor even if a reasonable mode can be found. Worse still, even assessing whether an MCMC algorithm has converged is an art all to itself [Cowles & Carlin, 1996]. By contrast, assessing whether a deterministic method has converged is very straightforward. The different representations of the posterior are significant from a user's point of view: while deterministic methods give an easy to handle parametric form, MCMC gives a set of posterior samples which must be stored, manipulated and summarised somehow. Comparing these methods computationally is very difficult: what is the model? what is the scale of the dataset? are we memory or CPU bound? can we parallelise? what error tolerance is acceptable? The answers to all these questions will affect the choice of algorithm. For example, in mixture models when sampling a particular component parameter we need only look at the points currently assigned to that component. In contrast, since standard variational methods will never put exactly zero probability on assigning a point to a cluster, we must consider all points when updating a single component parameter. While deterministic methods have been found to be faster than MCMC methods for certain models, their memory overhead is typically higher because of having to store messages. Some considerations are specific to a particular algorithm. EP can fail to converge, either entering a limit cycle or having parameters diverge, behaviour that is guaranteed not to occur for MCMC or VB. An attraction of VB is ease of "debugging": since every update should increase (or at least not decrease) the lower bound any erroneous updates can easily be detected.

Our contribution to this field is Non-conjugate Variational Message Passing, an extension of VMP that allows factors of almost arbitrary form. NCVMP has been incorporated into the free probabilistic programming language Infer.NET [Minka *et al.*, 2010], allowing practitioners to use a broader range of models than before. If a user requests VMP inference in a model with factors that can be handled by NCVMP but not by VMP, then NCVMP will be invoked automatically. Infer.NET allows the specification of complex and structured models without the burden of hand derived inference routines. This is another way of moving towards representing the complex nature of real world data.

Chapter 2

Factor analysis models

Principal components analysis (PCA), factor analysis (FA) and independent components analysis (ICA) are bilinear models which explain observed data, $\mathbf{y}_n \in \mathbb{R}^D$, in terms of a linear superposition of independent hidden factors, $\mathbf{x}_n \in \mathbb{R}^K$. We will focus on the application of such methods to gene expression modelling, where we envisage the hidden factors will correspond to underlying biological processes or confounding experimental factors. A microarray typically might have 10,000 probes (the unique sequences of complementary RNA that bind to messenger RNA), making analysing this data a very high dimensional problem. Moreover, the number of available samples is typically relatively small, on the order of hundreds rather than thousands, which introduces further challenges statistically.

The number of probes is D, the number of samples is N and the number of underlying gene signatures (latent factors) is K. The bilinear model we will consider is:

$$\mathbf{y}_n = \mathbf{G}\mathbf{x}_n + \boldsymbol{\epsilon}_n, \tag{2.1}$$

where $\mathbf{G}$ is the factor loading matrix and $\boldsymbol{\epsilon}_n$ is a noise vector, usually assumed to be Gaussian. Factor analysis and principal components analysis have become fundamental data analysis tools, used in data compression, image processing, ecology, genetics, portfolio management, and even time series data.

2.1 Principal components analysis

PCA is commonly derived in two complimentary fashions. The first, described in Hotelling [1933], is as the normalised linear projection which maximises the the variance in the projected space. Consider N vectors $\mathbf{y}_n$ with dimension D and sample mean $\bar{\mathbf{y}}$. The projection into the principal components space is $\mathbf{x}_n = \mathbf{G}^T(\mathbf{y}_n - \bar{\mathbf{y}})$, where $\mathbf{x}_n$ is the K-dimensional vector of principal components, and $\mathbf{G}$ is $D \times K$. The rows $\mathbf{w}_j$ of $\mathbf{G}$ are constrained to have unit length so that the problem is well defined. It can be shown that maximising the variance $|\sum_n \mathbf{x}_n \mathbf{x}_n^T / N|$ (where $|\cdot|$ denotes the determinant) is equivalent to setting the rows $\mathbf{w}_j$ of $\mathbf{G}$ equal to the eigenvectors of the sample covariance matrix $\mathbf{S} = \sum_n (\mathbf{y}_n - \bar{\mathbf{y}})(\mathbf{y}_n - \bar{\mathbf{y}})^T / N$ with the largest eigenvalues. The second derivation, dating back to Pearson [1901], is as the orthogonal projection which minimises the squared reconstruction error $\sum ||\mathbf{y}_n - \hat{\mathbf{y}}_n||^2$ where $\hat{\mathbf{y}}_n = \mathbf{G}\mathbf{x}_n + \bar{\mathbf{y}}$.

Tipping & Bishop [1999] and Roweis [1998] simultaneously noted the interpretation of Principal Components Analysis as maximum likelihood estimation in an appropriate probabilistic model. In both PCA and FA the latent factors are given a standard (zero mean, unit variance) normal prior. The only difference is that in PCA the noise is isotropic, whereas in FA the noise covariance is only constrained to be diagonal.

2.2 Factor analysis

Factor analysis (FA) was originally developed by the psychology community attempting to understand intelligence in terms of a small number of underlying "factors" [Young, 1941]. In Young's formulation, the $\mathbf{x}_n$ are viewed as parameters to be estimated. More recently, the convention has been to consider $\mathbf{x}_n$ as latent variables which can be given a prior and marginalised out. The latent factors are usually considered as random variables, and the mixing matrix as a parameter to estimate.

2.3 Inferring the latent dimensionality

Bishop [1999] extends the probabilistic PCA formulation of Tipping & Bishop [1999] to allow implicit inference of the latent dimensionality. Rather than performing a discrete model search, which could be performed for example by Bayesian model selection, Bishop uses the Automatic Relevance Determination (ARD) framework introduced by Mackay and Neal for complexity control in neural networks [MacKay, 1994]. Each column $\mathbf{g}_k$ of $\mathbf{G}$ is given a prior distribution $N(0, \alpha_k^{-1}\mathbf{I})$. Thus α_k is the precision (inverse variance) of $\mathbf{g}_k$. If α_k is inferred to be large, then $\mathbf{g}_k$ is forced towards zero, effectively suppressing this dimension. In the original ARD framework proposed by MacKay, Type-II maximum likelihood estimation of the α's is performed based on a Laplace approximation to a mode of the posterior of $\mathbf{G}$. Bishop follows this framework but also suggests Gibbs sampling or variational Bayes as alternative strategies to approximately marginalise out $\mathbf{G}$.

Minka [2000] shows that for probabilistic PCA, Bayesian model selection can be performed efficiently using a Laplace approximation. Laplace approximation involves fitting a Gaussian distribution to the posterior mode by matching the Hessian matrix of the log likelihood. Laplace's method is more accurate if a parameterisation can be found where the Gaussian approximation to the posterior is more reasonable. The noise variance σ^2 is a positive scalar. However, its logarithm, $\log \sigma^2$ can take any real value and it is therefore more reasonable to approximate the posterior as Gaussian. Minka uses an improper uniform prior on $\mathbf{m}$, the latent mean. The mixing matrix $\mathbf{G}$ is decomposed as

$$\mathbf{U}(\mathbf{L} - \sigma^2\mathbf{I}_k)^{1/2}\mathbf{R}$$

where $\mathbf{U}$ is an orthogonal basis (i.e. $\mathbf{U}^T\mathbf{U} = \mathbf{I}$), $\mathbf{L}$ is a diagonal scaling matrix with positive diagonal elements l_i, and $\mathbf{R}$ is an arbitrary and irrelevant rotation matrix. The condition $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ restricts $\mathbf{U}$ to a subspace known as the Stiefel manifold, which has a finite area given by an analytic expression. The matrix $\mathbf{U}$ can therefore be given a uniform prior distribution on the Stiefel manifold, with normalised density of one over the area. Parameterising the manifold in Euler

vector co-ordinates also makes Laplace's method more accurate. The fact that any column of $\mathbf{U}$ can be negated without changing the model means that there are 2^k identical modes in the posterior. To account for these the result from Laplace's method is simply multiplied by 2^k.

2.4 Sparsity

Sparsity is the idea that only some small proportion of coefficients should be non-zero. There are three main motivations for "sparse" models:

1. *Interpretability.* Having fewer active links in a model makes it easier to interpret.

2. *Reflects reality.* Many real-world systems are sparse. In genetics, transcription factors only bind to specific motifs, and therefore only regulate a small set of genes. In social networks, individuals only interact with a small number of friends relative to the total population. In finance, a company's performance is driven by only a few key factors. Incorporating this prior expectation of sparsity into a model is therefore often very natural.

3. *Improved predictive performance.* Sparsity helps prevent overfitting because coefficients that would be non-zero only because of noise in the data are forced to zero.

There are two main types of sparse model, which we refer to as "hard" and "soft". Hard sparsity means that coefficients have finite mass on zero, the main example being the so-called "slab-and-spike" prior, a mixture between a continuous distribution and a delta-spike at 0 [Ishwaran & Rao, 2003, 2005]. Soft sparsity means that the coefficient priors have heavy tails, and so a priori are likely to have either very small (but non-zero) or large values. Examples include the Laplace distribution corresponding to LASSO regression [Tibshirani, 1994], the student-t distribution [Geweke, 1993], or the Horseshoe prior [Carvalho *et al.*, 2009]. The student-t distribution can be efficiently modeled as a scale mixture of Gaussians by giving the inverse variance (precision) parameter a Gamma prior. Figure 2.1 shows the density of two independent student-t variables with degrees of freedom

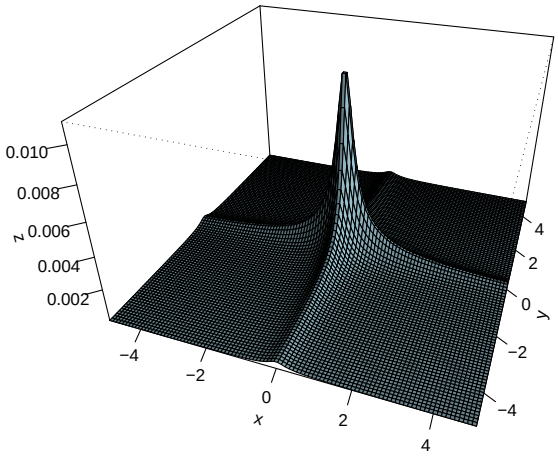


Figure 2.1: The density of two independent Student-t variables with 0.05 degrees of freedom.

$a = 0.05$. The mass is concentrated on the axes where one of the components is close to 0.

Another definition of hard vs. soft sparsity would be to consider whether the algorithm output can have coefficients that are exactly zero. Bayesian inference will generally not give this result, because the posterior given data will specify a probability of a coefficient being non-zero. Maximum likelihood or maximum a posterior (MAP) estimation however may give zero coefficients for certain types of prior (regularisation).

The student-t distribution is given by

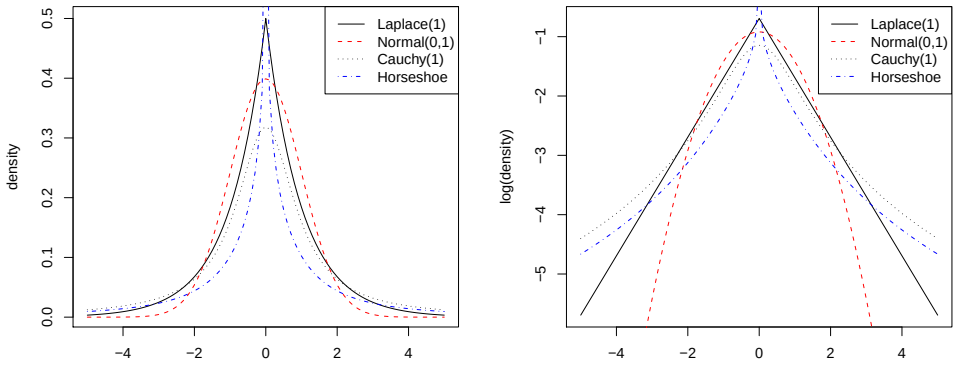
$$T(x; a, b) \propto \left[1 + \frac{1}{a}(x/b)^2\right]^{-\frac{a+1}{2}} . \tag{2.2}$$

This is achieved as a scale mixture by setting:

$$\lambda \sim G\left(\lambda; \frac{a}{2}, \frac{2}{ab^2}\right) \tag{2.3}$$

$$x|\lambda \sim N(x; 0, \lambda^{-1}). \tag{2.4}$$

The horseshoe prior is another scale mixture of normal prior which has recently been proposed [Carvalho *et al.*, 2009] which also admits a conjugate hierarchical representation [Armagan *et al.*, 2011]. The normal, Laplace, Cauchy, and horseshoe priors are shown in Figure 2.2. The tail of the normal prior falls off most sharply, as e^{-x^2}, which will result in the greatest bias for large coefficients. Next is the Laplace prior, where the tails decay as $e^{-|x|}$. The Cauchy and horseshoe priors both have tails which fall off as $1/x^2$. This slow decay results in reduced bias.



(a) Comparison of sparse priors (b) Log density shows tail behaviour

Figure 2.2: Comparing the normal, Laplace, Cauchy and horseshoe priors.

2.5 Sparse factor analysis

The idea of using a student-t prior, decomposed as a scale mixture of normal, in factor analysis, seems to have been simultaneously proposed in Fokoue [2004]

and Fevotte & Godsill [2006]. The elements of the mixing matrix $\mathbf{G}$ are given a student-t prior, and efficient inference is performed by introducing a per element precision as in Equation 2.3. Fokoue [2004] and Fevotte & Godsill [2006] perform Gibbs sampling to perform posterior inference in this model, whereas Cemgil *et al.* [2005] use variational EM [Ghahramani & Beal, 2001; Wainwright & Jordan, 2003].

The Bayesian Factor Regression Model (BFRM) of West *et al.* [2007] is closely related to the finite version of the model we will describe in the next chapter. The key difference is the use of a hierarchical sparsity prior. In both our model and BFRM each element of the mixing matrix $\mathbf{G}$ has a prior of the form

$$g_{dk} \sim (1 - \pi_{dk})\delta_0(g_{dk}) + \pi_{dk}\mathcal{N}\left(g_{dk}; 0, \lambda_k^{-1}\right). \tag{2.5}$$

In order to avoid requiring overly aggresive sparsifying priors, BFRM uses a hierarchical sparsity prior:

$$\pi_{dk} \sim (1 - \rho_k)\delta_0(\pi_{dk}) + \rho_k \text{Beta}(\pi_{dk}; am, a(1 - m)) \tag{2.6}$$

where $\rho_k \sim \text{Beta}(sr, s(1 - r))$. Non-zero elements of π_{dk} are given a diffuse prior favouring larger probabilities ($a = 10, m = 0.75$ are suggested in West *et al.* [2007]), and ρ_k is given a prior which strongly favours small values, corresponding to a sparse solution (e.g. $s = D, r = \frac{5}{D}$). Note that on integrating out π_{dk}, the prior on g_{dk} is

$$g_{dk} \sim (1 - m\rho_k)\delta_0(g_{dk}) + m\rho_k\mathcal{N}\left(g_{dk}; 0, \lambda_k^{-1}\right) \tag{2.7}$$

The LASSO-based Sparse PCA (SPCA) method of Zou *et al.* [2006] and Witten *et al.* [2009] has similar aims to our work in terms of providing a sparse variant of PCA to aid interpretation of the results. However, since SPCA is not formulated as a generative model it is not necessarily clear how to choose the regularization parameters or dimensionality without resorting to cross-validation. In our experimental comparison to SPCA we adjust the regularization constants such that each component explains roughly the same proportion of the total variance as the corresponding standard (non-sparse) principal component.

In Mohamed *et al.* [2011] sparse factor analysis is generalised to the general exponential family case, analogously to generalised linear models (GLMs) for regression. They show, on various real world datasets, that for Gaussian, binary and Poisson observations, spike-and-slab priors give greater predictive performance than using Laplace priors or L1 regularisation.

2.6 The Indian buffet process

The model presented in Chapter 3 will be based on the Indian buffet process (IBP). The IBP defines a distribution over infinite binary matrices, which can be used to construct latent feature models where the number of features is unbounded a priori. Models constructed using the IBP are sparse in that only a small number of features are typically active for each entity. The Indian Process Process (IBP) was originally introduced by Griffiths & Ghahramani [2005] and is reviewed in Griffiths & Ghahramani [2011]. Two and three parameter generalisations are developed in Ghahramani *et al.* [2007] and Teh & Görür [2009], a stick-breaking construction is presented in Teh *et al.* [2007], and the beta process [Hjort, 1990] is shown to be the de Finetti measure for the IBP in Thibaux & Jordan [2007].

2.6.1 Start with a finite model.

We derive the distribution on an infinite binary matrix $\mathbf{Z}$ by first defining a finite model with K features and taking the limit as $K \to \infty$. We then show how the infinite case corresponds to a simple stochastic process.

We have D dimensions and K hidden sources. Element Z_{dk} of matrix $\mathbf{Z}$ tells us whether the hidden factor k contributes to dimension d. We assume that the probability of factor k contributing to any dimension is π_k, and that the rows are generated independently. We find

$$P(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{k=1}^{K}\prod_{d=1}^{D} P(z_{dk}|\pi_k) = \prod_{k=1}^{K} \pi_k^{m_k}(1-\pi_k)^{D-m_k} \tag{2.8}$$

where $m_k = \sum_{d=1}^{D} z_{dk}$ is the number of dimensions to which source k contributes.

The inner term of the product is a binomial distribution, so we choose the conjugate Beta(r, s) distribution for π_k. For now we take $r = \frac{\alpha}{K}$ and $s = 1$, where α is the strength parameter of the IBP. The model is defined by

$$\pi_k|\alpha \sim \text{Beta}\left(\frac{\alpha}{K}, 1\right) \tag{2.9}$$

$$z_{dk}|\pi_k \sim \text{Bernoulli}(\pi_k) \tag{2.10}$$

Due to the conjugacy between the binomial and beta distributions we are able to integrate out π to find

$$P(\mathbf{Z}) = \prod_{k=1}^{K} \frac{\frac{\alpha}{K}\Gamma(m_k + \frac{\alpha}{K})\Gamma(D - m_k + 1)}{\Gamma(D + 1 + \frac{\alpha}{K})} \tag{2.11}$$

where $\Gamma(.)$ is the Gamma function.

2.6.2 Take the infinite limit.

Griffiths & Ghahramani [2005] define a scheme to order the non-zero rows of $\mathbf{Z}$ which allows us to take the limit $K \to \infty$ and find

$$P(\mathbf{Z}) = \frac{\alpha^{K_+}}{\prod_{h>0} K_h!} \exp\left(-\alpha H_D\right) \prod_{k=1}^{K_+} \frac{(D - m_k)!(m_k - 1)!}{D!}, \tag{2.12}$$

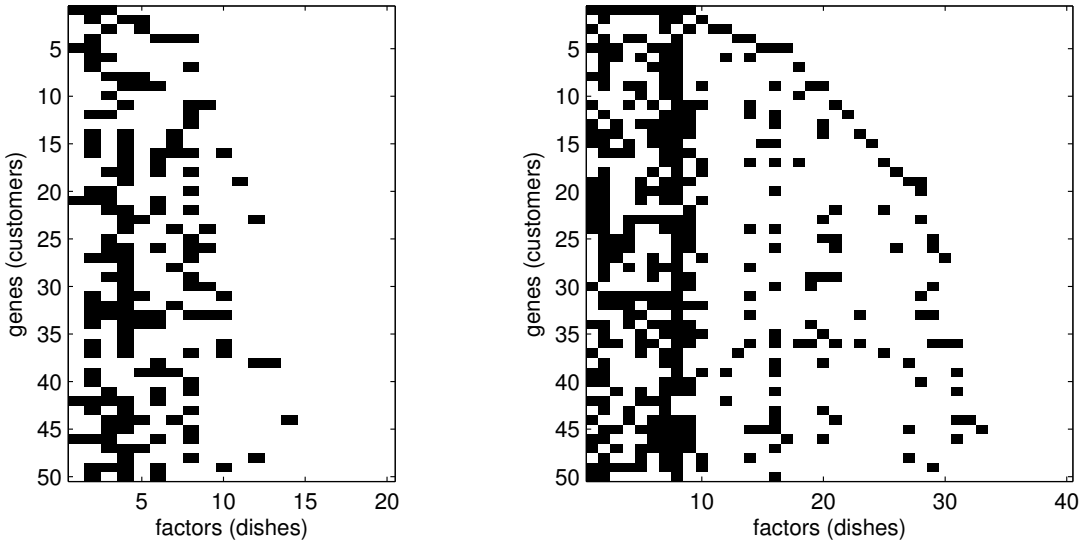
where K_+ is the number of active features (i.e. non-zero columns of $\mathbf{Z}$), $H_D := \sum_{j=1}^{D} \frac{1}{j}$ is the D-th harmonic number, and K_h is the number of rows whose entries correspond to the binary number h[1].

2.6.3 Go to an Indian buffet.

This distribution corresponds to a simple stochastic process, the Indian Buffet Process. Consider a buffet with a seemingly infinite number of dishes (hidden sources) arranged in a line. The first customer (observed dimension) starts at the left and samples Poisson(α) dishes. The ith customer moves from left to right sampling dishes with probability $\frac{m_k}{i}$ where m_k is the number of customers

[1] Equation 2.12 is actually the probability of an equivalence class of $\mathbf{Z}$'s

to have previously sampled dish k. Having reached the end of the previously sampled dishes, he tries Poisson($\frac{\alpha}{i}$) new dishes. Figure 2.3 shows two draws from the IBP for two different values of α.



(a) $\alpha = 4$ (b) $\alpha = 8$

Figure 2.3: Draws from the one parameter IBP for two different values of α.

If we apply the same ordering scheme to the matrix generated by this process as for the finite model, we recover the correct distribution over equivalance classes of matrices, which is exchangeable with respect to the customers. Since the distribution is exchangeable with respect to the customers we find by considering the last customer that

$$P(z_{kt} = 1|\mathbf{z}_{-kn}) = \frac{m_{k,-t}}{D} \tag{2.13}$$

where $m_{k,-t} = \sum_{s \neq t} z_{ks}$, which is used in sampling $\mathbf{Z}$. By exchangeability and considering the first customer, the number of active sources per dimension follows a Poisson(α) distribution, and the expected number of entries in $\mathbf{Z}$ is $D\alpha$. We also see that the number of active features, $K_+ = \sum_{d=1}^{D} \text{Poisson}(\frac{\alpha}{d}) = \text{Poisson}(\alpha H_D)$.

In the next chapter we will use the Indian Buffet Process to construct a factor analysis model that naturally incorporates sparsity and learns the appropriate number of latent factors to use for a specific dataset.

Chapter 3

Non-parametric sparse factor analysis

The work in this chapter was published in Knowles & Ghahramani [2011a]. In our earlier work [Knowles & Ghahramani, 2007] we investigated the use of sparsity on the latent factors $\mathbf{x}_n$, but this formulation is not appropriate in the case of modelling gene expression, where a transcription factor will regulate only a small set of genes, corresponding to sparsity in the factor loadings, $\mathbf{G}$. Here we propose a novel approach to sparse latent factor modelling where we place sparse priors on the factor loading matrix, $\mathbf{G}$. The Bayesian Factor Regression Model of West *et al.* [2007] is closely related to our work in this way, although the hierarchical sparsity prior they use is somewhat different, see Section 2.5. An alternative "soft" approach to incorporating sparsity is to put a Gamma(a, b) (usually exponential, i.e. $a = 1$) prior on the precision of each element of $\mathbf{G}$ independently, resulting in the elements of $\mathbf{G}$ being marginally student-t distributed a priori: see Section 2.4 for more details. We compare these three sparsity schemes empirically in the context of gene expression modelling.

We use the Indian Buffet Process [Griffiths & Ghahramani, 2005], reviewed in Section 2.6, which defines a distribution over infinite binary matrices, to provide sparsity and a framework for inferring the appropriate latent dimension for the dataset using a straightforward Gibbs sampling algorithm. The Indian Buffet Process (IBP) allows a potentially unbounded number of latent factors, so we

do not have to specify a maximum number of latent dimensions a priori. We denote our model "NSFA" for "non-parametric sparse Factor Analysis". Our model is closely related to that of Rai & Daumé III [2008], and is a simultaneous development.

3.1 Model

As in the previous chapter we will consider the bilinear model for the observed data $\mathbf{y}_n$:

$$\mathbf{y}_n = \mathbf{G}\mathbf{x}_n + \boldsymbol{\epsilon}_n, \tag{3.1}$$

where $\mathbf{G}$ is the factor loading matrix and $\boldsymbol{\epsilon}_n$ is a noise vector. Let $\mathbf{Z}$ be a binary matrix whose (d, k)-th element represents whether observed dimension d includes any contribution from factor k. We then model the factor loading matrix by

$$p(G_{dk}|Z_{dk}, \lambda_k) = Z_{dk}\mathcal{N}\left(G_{dk}; 0, \lambda_k^{-1}\right) + (1 - Z_{dk})\delta_0(G_{dk}) \tag{3.2}$$

where λ_k is the inverse variance (precision) of the kth factor and δ_0 is a delta function (point-mass) at 0. Distributions of this type are sometimes known as "spike and slab" distributions. We allow a potentially infinite number of hidden sources, so that $\mathbf{Z}$ has infinitely many columns, although only a finite number will have non-zero entries. This construction allows us to use the IBP to provide sparsity and define a generative process for the number of latent factors.

We will now describe the modelling choices available for the rest of the model. We assume independent Gaussian noise, $\boldsymbol{\epsilon}_n$, with diagonal covariance matrix, $\boldsymbol{\Psi}$. We find that for many applications assuming isotropic noise is too restrictive, but this option is available for situations where there is strong prior belief that all observed dimensions should have the same noise variance. The latent factors, $\mathbf{x}_n$, are given Gaussian priors. The graphical model is shown in Figure 3.1.

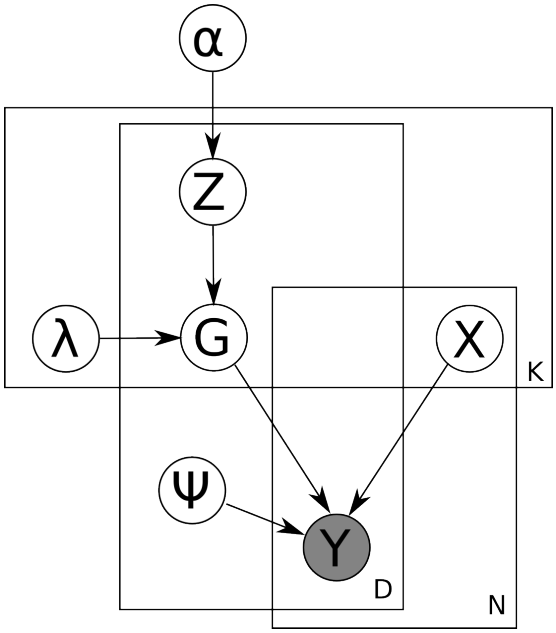


Figure 3.1: Graphical model for Non-parametric Sparse Factor Analysis

3.2 Inference

Given the observed data $\mathbf{Y}$, we wish to infer the hidden factors $\mathbf{X}$, which factors are active for each observed dimension $\mathbf{Z}$, the factor loading matrix $\mathbf{G}$, and all hyperparameters. We use Gibbs sampling, but with Metropolis-Hastings (MH) steps for sampling new features. We draw samples from the posterior distribution of the model parameters given the data by successively sampling the conditional distributions of each parameter in turn, given all other parameters.

Since we assume independent Gaussian noise, the likelihood function is

$$P(\mathbf{Y}|\mathbf{G},\mathbf{X},\boldsymbol{\psi}) = \prod_{n=1}^{N} \frac{1}{(2\pi)^{\frac{D}{2}}|\boldsymbol{\psi}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{y}_n - \mathbf{G}\mathbf{x}_n)^T\boldsymbol{\psi}^{-1}(\mathbf{y}_n - \mathbf{G}\mathbf{x}_n)\right) \quad (3.3)$$

where $\boldsymbol{\psi}$ is a diagonal noise precision matrix.

Mixture coefficients. $\mathbf{Z}$ is a matrix with infinitely many columns, but only the non-zero columns contribute to the likelihood and are held in memory. However, the zero columns still need to be taken into account since the number of active

factors can change. This can be done by Gibbs sampling the non-zero columns and performing a Metropolis-Hastings move on the number, κ_d, of "singleton" features for each row d in turn. Thus κ_d is the number of columns of $\mathbf{Z}$ which contain 1 only in row d, i.e. the number of features which are active only for dimension d. Figure 3.2 illustrates κ_d for a sample $\mathbf{Z}$ matrix. Let $m_{-d,k} = \sum_{c \neq d} Z_{dk}$ be the number of rows for which feature k is active, excluding the current row d.

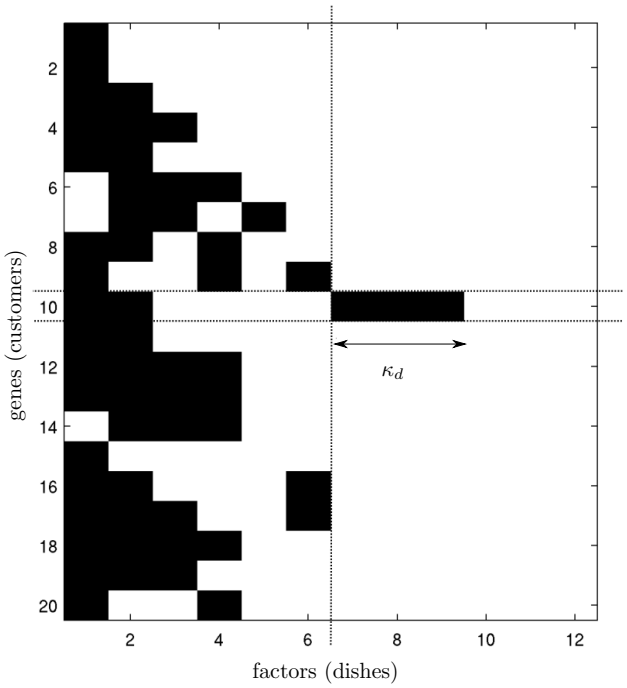


Figure 3.2: A diagram to illustrate the definition of κ_d, for $d = 10$.

It was pointed out in Griffiths & Ghahramani [2011] that it is invalid to simply Gibbs sample the singleton features so that $\kappa_d = 0$ as a result of Equation 3.6, as is done in various references [Doshi-Velez & Ghahramani, 2009a; Knowles & Ghahramani, 2007], because this corresponds to choosing the Markov chain kernel depending on the current state of the variable being sampled. Specifically for columns of $\mathbf{Z}$ where $m_{-d,k} = 0$, this method Gibbs samples Z_{dk} if $Z_{dk} = 1$ (thereby setting $Z_{dk} = 0$ deterministically) and samples it as part of a MH step if $Z_{dk} = 0$. However, there is a simple solution: we are allowed to choose the MC kernel depending on the state of variables that are currently fixed, since

the transition kernel for the variables currently being sampled will still maintain detailed balance. We will Gibbs sample $Z_{d:}$ for columns where $m_{-d,k} > 0$, and use an MH step to sample how many of the infinitely many remaining elements where $m_{-d,k} = 0$ should be on. Thus the choice of kernel depends only on the state of *other* rows of $\mathbf{Z}$ than the row currently being sampled. We will show empirically in Section 3.2.1 that while the method proposed here does sample from the correct equilibrium distribution, deleting all singleton features before adding new ones does not, and systematically underestimates the number of latent features. This correct method is in fact proposed in Meeds *et al.* [2006], although they do not point this out.

Thus for row d we Gibbs sample the elements Z_{dk} for which $m_{-d,k} > 0$. Integrating out the Gaussian "slab" for the (d,k)-th element of the factor loading matrix, g_{dk}, in Equation 3.2 we obtain

$$\frac{P(\mathbf{Y}|Z_{dk}=1,-)}{P(\mathbf{Y}|Z_{dk}=0,-)} = \frac{\int P(\mathbf{Y}|g_{dk},-)\mathcal{N}\left(g_{dk};0,\lambda_k^{-1}\right)dg_{dk}}{P(\mathbf{Y}|g_{dk}=0,-)} \tag{3.4}$$

$$= \sqrt{\frac{\lambda_k}{\lambda}}\exp\left(\frac{1}{2}\lambda\mu^2\right) \tag{3.5}$$

where $-$ denotes the current state of the chain excluding those variables explicitly mentionned, $\lambda = \psi_d^{-1}X_{k:}^T X_{k:} + \lambda_k$ and $\mu = \frac{\psi_d^{-1}}{\lambda}X_{k:}^T\hat{E}_{d:}$ with the matrix of residuals $\hat{\mathbf{E}} = \mathbf{Y} - \mathbf{GX}$ evaluated with $G_{dk} = 0$. The dominant calculation is that for μ since the calculation for λ can be cached. This operation is $O(N)$ and must be calculated $D \times K$ times, so sampling the IBP matrix, $\mathbf{Z}$ and factor loading matrix, $\mathbf{G}$ is order $O(NDK)$.

From the exchangeability of the IBP we see that the ratio of the priors is

$$\frac{P(Z_{dk}=1|-)}{P(Z_{dk}=0|-)} = \frac{m_{-d,k}}{D-1-m_{-d,k}} \tag{3.6}$$

Multiplying Equations 3.5 and 3.6 gives the expression for the ratio of posterior probabilities for Z_{dk} being 1 or 0, which is used for sampling. If Z_{dk} is set to 1, we sample $g_{dk}|- \sim \mathcal{N}(\mu,\lambda^{-1})$ with μ,λ defined as for Equation 3.5.

Adding new features. New features are proposed by sampling the number of singleton features for row d, denoted κ_d, with a MH step, where the proposed new value is κ_d^*. It is straightforward to integrate out either the new elements of the mixing matrix, $\mathbf{g}^*$ (a $1 \times \kappa_d^*$ vector), or the new rows of the latent feature matrix, $\mathbf{X}^*$ (a $\kappa_d^* \times N$ matrix), but not both. Since the latter is likely to have higher dimension, we choose to integrate out $\mathbf{X}^*$ and include $\mathbf{g}^*$ as part of the proposal. Thus the proposal is $\xi^* = \{\kappa_d^*, \mathbf{g}^*\}$, and we propose a move $\xi \to \xi^*$ with probability $J(\xi^*|\xi)$. Our proposal will be of the form

$$J(\xi^*|\xi) = J(\kappa_d^*)J(\mathbf{g}^*). \tag{3.7}$$

The simplest proposal, following Meeds *et al.* [2006], would be to use the prior on ξ^*, i.e.

$$J(\kappa_d^*) = P(\kappa_d^*|\alpha) = \text{Poisson}\left(\kappa_d^*; \gamma\right), \tag{3.8}$$
$$J(\mathbf{g}^*) = p(\mathbf{g}^*|\kappa_d^*, \lambda_k) = N(\mathbf{g}^*; 0, \lambda_k^{-1}), \tag{3.9}$$

where $\gamma = \frac{\alpha}{D}$.

Unfortunately, the rate constant of the Poisson prior tends to be so small that new features are very rarely proposed, resulting in slow mixing. To remedy this we modify the proposal distribution for κ_d^* and introduce two tunable parameters, π and λ.

$$J(\kappa_d^*) = (1-\pi)\text{Poisson}\left(\kappa_d^*; \lambda\gamma\right) + \pi\mathbf{1}(\kappa_d^* = 1) \tag{3.10}$$

Thus the Poisson rate is multiplied by a factor λ, and a spike at $\kappa_d^* = 1$ is added with mass π.

The proposal is accepted with probability $\min\left(1, a_{\xi\to\xi^*}\right)$ where

$$a_{\xi\to\xi^*} = \frac{P(\xi^*|\text{rest}, Y)J(\xi|\xi^*)}{P(\xi|\text{rest}, Y)J(\xi^*|\xi)} = \frac{P(Y|\xi^*, \text{rest})P(\kappa_d^*|\alpha)J(\kappa_d)}{P(Y|\xi, \text{rest})J(\kappa_d^*)P(\kappa_d|\alpha)} = a_l \cdot a_p, \tag{3.11}$$

where

$$a_l = \frac{P(Y|\xi^*, \text{rest})}{P(Y|\xi, \text{rest})}, \qquad a_p = \frac{P(\kappa_d^*|\alpha)J(\kappa_d)}{J(\kappa_d^*)P(\kappa_d|\alpha)}. \tag{3.12}$$

To calculate a_l we need the collapsed likelihood under the new proposal:

$$P(Y_{d:}|\xi^*,-) = \prod_{n=1}^{N} \int P(Y_{dn}|\xi^*,\mathbf{x}'_n,-)P(\mathbf{x}'_n)d\mathbf{x}'$$
$$= \prod_{n=1}^{N} \frac{1}{(2\pi\psi_d^{-1})^{\frac{1}{2}}|\mathbf{M}^*|^{\frac{1}{2}}} \exp\left(\frac{1}{2}(\mathbf{m}_n^{*T}\mathbf{M}\mathbf{m}_n^* - \psi_d^{-1}\hat{E}_{dn}^2)\right) \tag{3.13}$$

where $\mathbf{M}^* = \psi_d^{-1}\mathbf{g}^*\mathbf{g}^{*T} + I_{\kappa_d}$ and $\mathbf{m}_n^* = \mathbf{M}^{*-1}\psi_d^{-1}\mathbf{g}^*\hat{E}_{dn}$ where the residuals $\hat{E}_{dn}$ are with all singletons for row d switched off, ie. $\hat{E}_{dn} = Y_{dn} - \sum_{k:m_{-d,k}>0} G_{dk}X_{kn}$. Note that $Y_{d:}$ denotes taking a "slice" of a matrix, in this case the d-th row. Equivalently, the collapsed likelihood for ξ, the current set of singletons for row d is

$$P(Y_{d:}|\xi,-) = \prod_{n=1}^{N} \frac{1}{(2\pi\psi_d^{-1})^{\frac{1}{2}}|\mathbf{M}|^{\frac{1}{2}}} \exp\left(\frac{1}{2}(\mathbf{m}_n^{T}\mathbf{M}\mathbf{m}_n - \psi_d^{-1}\hat{E}_{dn}^2)\right) \tag{3.14}$$

where $\mathbf{M} = \psi_d^{-1}\mathbf{g}\mathbf{g}^T + I_{\kappa_d}$ and $\mathbf{m}_n = \mathbf{M}^{-1}\psi_d^{-1}\mathbf{g}\hat{E}_{dn}$ with $\mathbf{g} = \{G_{dk} : m_{-d,k} = 0\}$ the current values of $\mathbf{G}$ for the singletons of row d. Note that while it might be tempting to calculate $P(Y_{d:}|\xi,-)$ conditioning on the current factor values for the existing singletons $\{X_{k:} : m_{-d,k} = 0\}$, this is incorrect since the numerator and denominator in the MH acceptance ratio are then calculated with different parts of the model collapsed. Substituting the likelihood terms in Equations 3.14 and 3.13 into the Equation 3.12 for the ratio of likelihood terms, a_l, gives

$$a_l = \frac{|\mathbf{M}^*|^{-\frac{N}{2}} \exp\left(\frac{1}{2}\sum_n \mathbf{m}_n^{*T}\mathbf{M}^*\mathbf{m}_n^*\right)}{|\mathbf{M}|^{-\frac{N}{2}} \exp\left(\frac{1}{2}\sum_n \mathbf{m}_n^{T}\mathbf{M}\mathbf{m}_n\right)} \tag{3.15}$$

We found that appropriate scheduling of the sampler improved mixing, particularly with respect to adding new features. The final scheme we settled on is described in Algorithm 1.

IBP hyperparameter. We can choose to sample the IBP strength parameter α, with conjugate Gamma(a_α, b_α) prior (note that we use the inverse scale parameterisation of the Gamma distribution). The conditional prior of Equation (2.12),

acts as the likelihood term and the posterior update is as follows:

$$P(\alpha|\mathbf{Z}) \propto P(\mathbf{Z}|\alpha)P(\alpha) = \text{Gamma}\left(\alpha; K_+ + a_\alpha, b_\alpha + H_D\right) \tag{3.16}$$

where K_+ is the number of active sources and H_D is the D-th harmonic number.

Latent variables. The remaining sampling steps are standard, but are included here for completeness. Sampling the columns of the latent variable matrix $\mathbf{X}$ for each $n \in [1, \ldots, N]$ we have

$$P(\mathbf{x}_n|-) \propto P(\mathbf{y}_n|\mathbf{x}_n, -)P(\mathbf{x}_n) = \mathcal{N}\left(\mathbf{x}_n; \boldsymbol{\mu}_n, \boldsymbol{\Lambda}\right) \tag{3.17}$$

where $\boldsymbol{\Lambda} = \mathbf{G}^T\boldsymbol{\psi}^{-1}\mathbf{G} + I$ and $\boldsymbol{\mu}_n = \boldsymbol{\Lambda}^{-1}\mathbf{G}^T\boldsymbol{\psi}^{-1}\mathbf{y}_n$. Note that since $\boldsymbol{\Lambda}$ does not depend on t we only need to compute and invert it once per iteration. Calculating $\boldsymbol{\Lambda}$ is order $O(K^2D)$, and calculating its Cholesky decomposition is $O(K^3)$. Calculating $\boldsymbol{\mu}_n$ is order $O(KD)$ and must be calculated for all N $\mathbf{x}_n$'s, a total of $O(NKD)$. Thus sampling $\mathbf{X}$ is order $O(K^2D + K^3 + NKD)$.

Factor covariance. If the mixture coefficient variances are constrained to be equal, we have $\lambda_k = \lambda \sim \text{Gamma}(a_\lambda, b_\lambda)$. The posterior update is then given by $\lambda|G \sim \text{Gamma}(a_\lambda + \frac{\sum_k m_k}{2}, b_\lambda + \sum_{d,k} G_{dk}^2)$.

However, if the variances are allowed to be different for each column of $\mathbf{G}$, we set $\lambda_k \sim \text{Gamma}(a_\lambda, b_\lambda)$, and the posterior update is given by $\lambda_k|G \sim \text{Gamma}(a_\lambda + \frac{m_k}{2}, b_\lambda + \sum_d G_{dk}^2)$. In this case we may also wish to share power across factors, in which case we also sample b_λ. Putting a Gamma prior on b_λ such that $b_\lambda \sim \text{Gamma}(a_{\lambda 0}, b_{\lambda 0})$, the posterior update is $b_\lambda|\lambda_k \sim \text{Gamma}(a_{\lambda 0} + cK, b_{\lambda 0} + \sum_{k=1}^K \lambda_k)$

Noise variance. The additive Gaussian noise can be constrained to be isotropic, in which case the inverse variance is given a Gamma prior: $\psi_d^{-1} = \psi^{-1} \sim \text{Gamma}(a_\psi, b_\psi)$ which gives the posterior update $\psi^{-1}|- \sim \text{Gamma}(a_\psi + \frac{ND}{2}, b_\psi + \sum_{d,n} E_{dn}^2)$ (where $\mathbf{E}$ is again the matrix of residuals).

However, if the noise is only assumed to be independent, then each dimension has a separate variance, whose inverse is given a Gamma prior: $\psi_d^{-1} \sim$

Gamma(a_ψ, b_ψ) which gives the posterior update $\psi_d^{-1}|- \sim$ Gamma$(a + \frac{N}{2}, b + \sum_n E_{dn}^2)$. If b_ψ is given prior distribution Gamma$(a_{\psi 0}, b_{\psi 0})$ the Gibbs update is $b_\psi|- \sim$ Gamma$(a_{\psi 0} + aD, b_{\psi 0} + \sum_{d=1}^{D} \psi_d^{-1})$.

Algorithm 1 One iteration of the NSFA sampler

for $d = 1$ to D **do**
  for $k = 1$ to K **do**
    if $m_{-d,k} > 0$ **then**
      Sample Z_{dk} using Equations 3.6 and 3.5
    end if
  end for
  Propose having κ_d^* singleton features, according to Equation 3.7
  Accept singletons according to Equation 3.11
end for
for $n = 1$ to N **do**
  Sample $X_{:n}$ using Equation 3.17
end for
Sample α, ϕ, λ_g as detailed above.

3.2.1 Getting it right

To validate our sampling algorithm for NSFA we follow the joint distribution testing method of Geweke [2004]. There are two ways to sample from the joint distribution, $P(Y, \theta)$ over parameters, θ and data, Y defined by a probabilistic model such as NSFA. The first we will refer to "marginal-conditional" sampling:

for $m = 1$ to M **do**
  $\theta^{(m)} \sim P(\theta)$
  $Y^{(m)} \sim P(Y|\theta^{(m)})$
end for

Both steps here are straightforward: sampling from the prior followed by sampling from the likelihood model. The second way, refered to as "successive-conditional" sampling, proceeds as follows:

$\theta^{(1)} \sim P(\theta)$
$Y^{(1)} \sim P(Y|\theta^{(1)})$

for $m = 2$ to M **do**

$\theta^{(m)} \sim Q(\theta|\theta^{(m-1)}, Y^{(m-1)})$

$Y^{(m)} \sim P(Y|\theta^{(m)})$

end for

where Q represents a single (or multiple) iteration(s) of our MCMC sampler. To validate our sampler we can then check, either informally or using hypothesis tests, whether the samples drawn from the joint $P(Y, \theta)$ in these two different ways appear to have come from the same distribution.

We apply this method to our NSFA sampler with just $N = D = 2$, and all hyperparameters fixed as follows: $\alpha = 2, \beta = 0, \psi_d = 1, \lambda_k = 1$. We draw 10^4 samples using both the marginal-conditional and successive-conditional procedures, the latter for both methods of sampling new features (see Section 3.2). We can look at various characteristics of the samples, but we focus here on the number of active features. The distribution of the number of features under the correct successive-conditional sampler matches that under the marginal-conditional sampler almost perfectly, but the old method where all singleton features of the current row are deleted before proposing adding new features results in significantly fewer features being active (see Figure 3.3). It is surprising that what seems like a small change can have such a large effect on the equilibrium distribution: the expected numbers of features is 3, but the incorrect sampler gives an empirical average of 1.21. Under the correct successive-conditional sampler the average number of features is 3.0034: a hypothesis test did not reject the null hypothesis that the means of the two distributions are equal. While this cannot completely guarantee correctness of the algorithm and code, 10^4 samples is a large number for such a small model and thus gives strong evidence that our algorithm is correct.

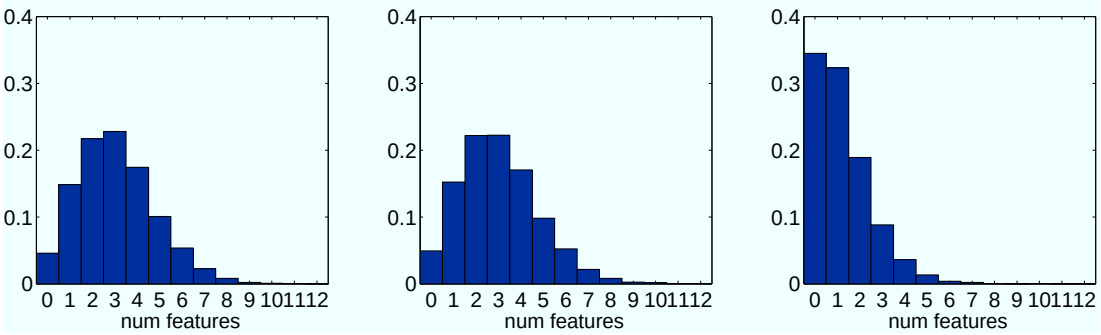


Figure 3.3: Joint distribution test results. Histograms of number of features for 10^4 samples drawn from the NSFA model in different ways. *Left:* from the prior with $\alpha = 2, D = 2$. *Middle:* the successive-conditional procedure, using correct sampling of new features. *Right:* the successive-conditional procedure, with deleting all singleton features before proposing adding new features.

3.3 Results

We compare the following models:

- FA - Bayesian Factor Analysis, see for example Kaufman & Press [1973] or Rowe & Press [1998]
- AFA - Factor Analysis with ARD prior to determine active sources
- FOK - The sparse Factor Analysis method of Fokoue [2004], Fevotte & Godsill [2006] and Archambeau & Bach [2009]
- SPCA - The Sparse PCA method of Zou *et al.* [2006]
- BFRM - Bayesian Factor Regression Model of West *et al.* [2007].
- SFA - Sparse Factor Analysis, using the finite IBP
- NSFA - The proposed model: Nonparametric Sparse Factor Analysis

We use one synthetic dataset and three real biological datasets.

3.3.1 Synthetic data

Since generating a connectivity matrix $\mathbf{Z}$ from the IBP itself would clearly bias towards our model, we instead use the $D = 100$ gene by $K = 16$ factor *E. Coli* connectivity matrix derived in Kao *et al.* [2004] from RegulonDB and current literature. We ignore whether the connection is believed to be up or down regulation, resulting in a binary matrix $\mathbf{Z}$. We generate random datasets with $N = 100$ samples by drawing the non-zero elements of $\mathbf{G}$ (corresponding to the elements of $\mathbf{Z}$ which are non-zero), and all elements of $\mathbf{X}$, from a zero mean unit variance Gaussian, calculating $\mathbf{Y} = \mathbf{GX} + \mathbf{E}$, where $\mathbf{E}$ is Gaussian white noise with variance set to give a signal to noise ratio of 10.

As a metric for evaluating performance, we introduce the reconstruction error, E_r as

$$E_r(G, \hat{G}) = \frac{1}{DK} \sum_{k=1}^{K} \min_{\hat{k} \in \{1,..,\hat{K}\}} \sum_{d=1}^{D} (G_{dk} - \hat{G}_{d\hat{k}})^2, \tag{3.18}$$

where $\hat{G}, \hat{K}$ are the inferred quantities. Although we minimize over permutations, we do not minimize over rotations since, as noted in Fokoue [2004], the sparsity of the prior stops the solution being rotation invariant. We average this error over the last 10 samples of the MCMC run. This error function does not penalize inferring extra spurious factors, so we will investigate this possibility separately. The precision and recall of active elements of the $\mathbf{Z}$ achieved by each algorithm (after thresholding for the non-sparse algorithms) were investigated but are omitted here since the results are consistent with the reconstruction error.

The reconstruction error for each method with different numbers of latent features is shown in Figure 3.4. Ten random datasets were used and for the sampling methods (all but SPCA) the results were averaged over the last ten samples out of 1000. Unsurprisingly, plain Factor Analysis (FA) performs the worst, with increasing overfitting as the number of factors is increased. For $\hat{K} = 20$ the variance is also very high, since the four spurious features fit noise. Using an ARD prior on the features (AFA) improves the performance, and overfitting no longer occurs. The reconstruction error is actually less for $\hat{K} = 20$, but this is an artefact due to the reconstruction error not penalizing additional spurious features

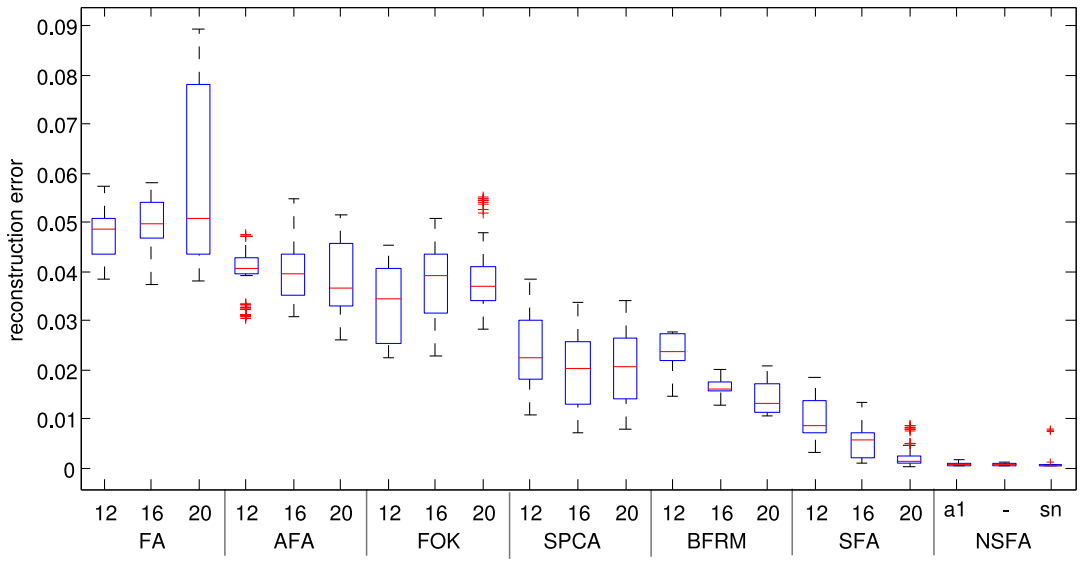


Figure 3.4: Boxplot of reconstruction errors for simulated data derived from the *E. Coli* connectivity matrix of Kao *et al.* [2004]. Ten datasets were generated and the reconstruction error calculated for the last ten samples from each algorithm. Numbers refer to the number of latent factors used, K. *a1* denotes fixing $\alpha = 1$. *sn* denotes sharing power between noise dimensions.

in the inferred $\mathbf{G}$. The Sparse PCA (SPCA) of Zou *et al.* [2006] shows improved reconstruction compared to the non-sparse methods (FA and AFA) but does not perform as well as the Bayesian sparse models. Sparse factor analysis (SFA), the finite version of the full infinite model, performs very well. The Bayesian Factor Regression Model (BFRM) performs significantly better than the ARD factor analysis (AFA), but not as well as our sparse model (SFA). It is interesting that for BFRM the reconstruction error decreases significantly with increasing $\hat{K}$, suggesting that the default priors may actually encourage too much sparsity for this dataset. Fokoue's method (FOK) only performs marginally better than AFA, suggesting that this "soft" sparsity scheme is not as effective at finding the underlying sparsity in the data. Overfitting is also seen, with the error increasing with $\hat{K}$. This could potentially be resolved by placing an appropriate per factor ARD-like prior over the scale parameters of the Gamma distributions controlling the precision of elements of $\mathbf{G}$. Finally, the Non-parametric Sparse Factor Analysis (NSFA) proposed here and in Rai & Daumé III [2008] performs very well, with reconstruction errors an order of magnitude smaller than the competing methods. With fixed $\alpha = 1$ (a1) or inferring α we see very similar performance. Using the soft coupling (sn) variant which shares power between dimensions when fitting the noise variances gives a lower median error, which is reasonable in this example since the noise was in fact isotropic, but has a few of outlying solutions with somewhat poorer reconstruction error.

Since the reconstruction error does not penalize spurious factors it is important to check that NSFA is not scoring well simply by inferring many additional factors. Histograms for the number of latent features inferred by the nonparametric sparse model for a typical synthetic dataset are shown in Figure 3.5. This represents an approximate posterior over K. For both fixed $\alpha = 1$ and inferred $\alpha = 2.78 \pm 0.66$ the posterior for K is concentrated on the true value $K = 16$, with minimal bias: $\mathbb{E}K = 16.08$ with $\alpha = 1$ and $\mathbb{E}K = 16.28$ with learnt α. The variance of the posterior on K is slightly increased when learning alpha, as we would expect, increasing from a standard deviation of 0.294 to 0.536. These results suggest our results are reasonably robust to the choice of α. The posteriors exhibit negative skew: values of K less than 16 are very unlikely since this involves removing a useful feature, but the sampler occasionally "imagines"

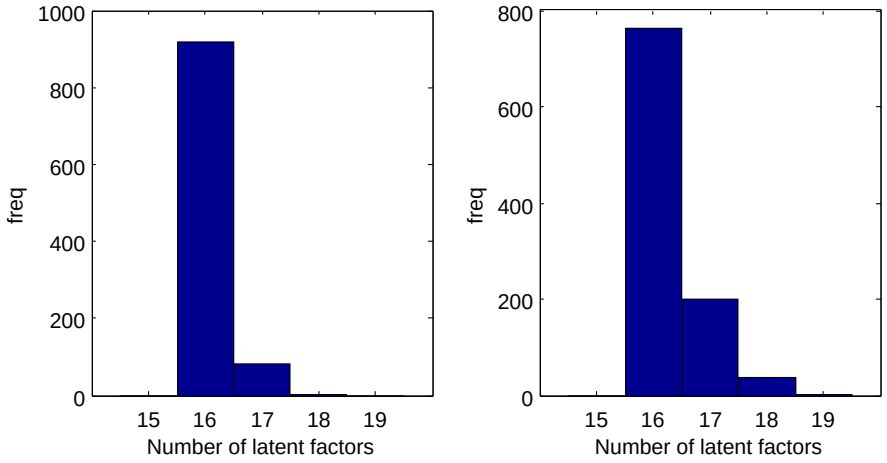


Figure 3.5: Histograms of the number of latent features inferred by the nonparametric sparse FA sampler for the last 1000 samples out of 10,000. *Left:* With $\alpha = 1$. *Right:* Inferring α.

features in the added Gaussian noise. For some of the random datasets, the posterior on K concentrates on values less than the true value 16, maybe because elements of $\mathbf{Z}$ which are 1 are masked by very small corresponding values of $\mathbf{G}$. This hypothesis is supported by the results of a similar experiment where $\mathbf{G}$ was set equal to $\mathbf{Z}$ (with no additional Gaussian weight). In this case, the sampler always converged to at least 16 features, but would also sometimes infer spurious features from noise (results not shown).

3.3.2 Convergence

NSFA can sometimes suffer from slow convergence if the number of new features is drawn from the prior. Figure 3.6 shows how the different proposal distributions for the number of unique features in the d-th row, κ_d, effect how quickly the sampler reaches a sensible number of features. If we use the prior as the proposal distribution, the burn in period is quite long, taking around 400 iterations to reach equilibrium. If a mass of 0.1 is added at $\kappa_d = 1$, then the sampler creates almost all (15) features in around 100 iterations, but then takes some time (roughly another 500 iterations) to properly incorporate the 16th feature, presumably because other variables in the model are still burning in. The best performance in this case seems to be given by setting the factor $\lambda = 10$ in Equa-

tion 3.10, where 16 features are reached after only 200 iterations. Increasing λ further greatly decreases the acceptance ratio of adding new features, especially after the features are added initially. Although only 1000 iterations are shown here, all these experiments were run for 10,000 iterations to confirm convergence.

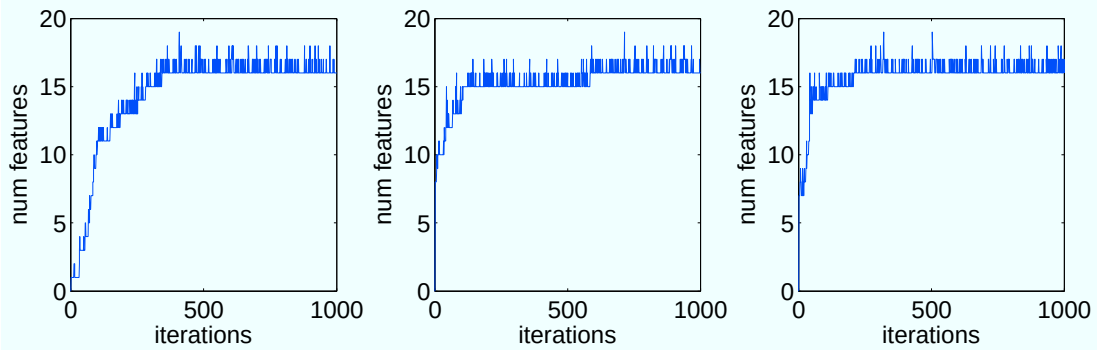
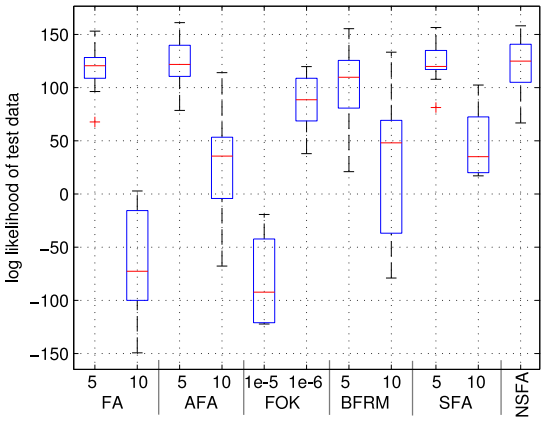


Figure 3.6: The effect of different proposal distributions for the number of new features. *Left:* Prior. *Middle:* Prior plus $0.1\mathbb{I}(\kappa = 1)$. *Right:* Factor $\lambda = 10$.

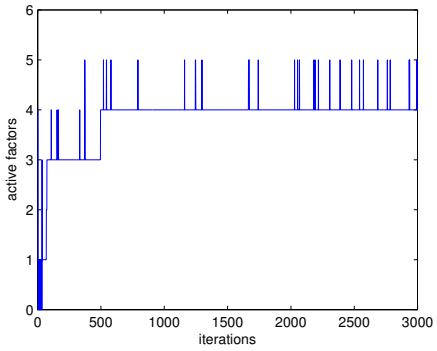
3.3.3 *E. Coli* time-series dataset from Kao *et al.* [2004]

To assess the performance of each algorithm on biological data where no ground truth is available, we calculated the log likelihood of heldout test data under the predictive distribution given by the fitted model. Ten percent of entries from $\mathbf{Y}$ were removed at random, ten times, to give ten datasets for inference. We do not use mean square error as a measure of predictive performance because of the large variation in the signal to noise ratio across gene expression level probes.

The test log likelihood achieved by the various algorithms on the *E. Coli* dataset from Kao *et al.* [2004], including 100 genes at 24 time-points, is shown in Figure 3.7a. On this simple dataset incorporating sparsity doesn't improve predictive performance. Overfitting the number of latent factors does damage performance, although using the ARD or sparse prior alleviates the problem. Based on predictive performance of the finite models, five is a sensible number of features for this dataset: the NSFA model infers a median number of 4 features, with some probability of there being 5, as shown in Figure 3.7b.



(a) Log likelihood of test data under each model based on the last 100 MCMC samples. The boxplots show variation across 10 different random splits of the data into training and test sets.



(b) Number of active latent features during a typical MCMC run of the NSFA model.

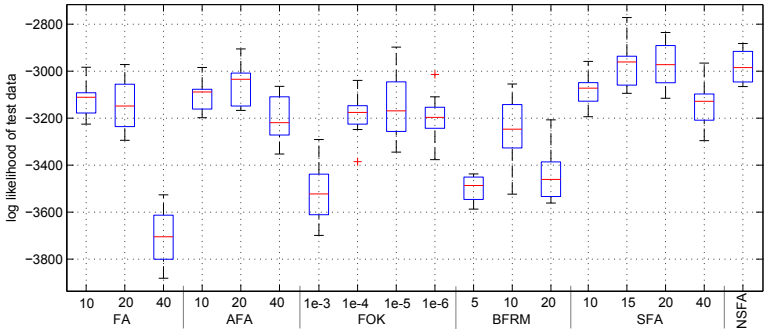
Figure 3.7: Results on *E. Coli* time-series dataset from Kao *et al.* [2004] ($N = 24, D = 100$, 3000 MCMC iterations).

3.3.4 Breast cancer dataset from West *et al.* [2007]

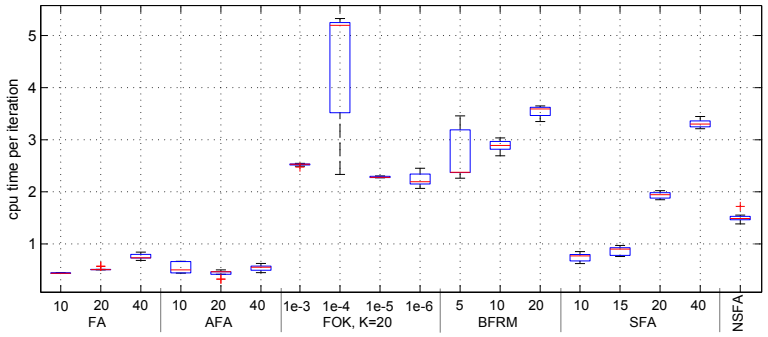
We assess these algorithms in terms of *predictive performance* on the breast cancer dataset of West *et al.* [2007], including 226 genes across 251 individuals. We find that all the finite models are sensitive to the choice of the number of factors, K. The samplers were found to have converged after around 1000 samples according to standard multiple chain convergence measures, so 3000 MCMC iterations were used for all models. The predictive log likelihood was calculated using the final 100 MCMC samples. Figure 3.8a shows test set log likelihoods for 10 random divisions of the data into training and test sets. Factor analysis (FA) shows significant overfitting as the number of latent features is increased from 20 to 40. Using the ARD prior prevents this overfitting (AFA), giving improved performance when using 20 features and only slightly reduced performance when 40 features are used. The sparse finite model (SFA) shows an advantage over AFA in terms of predictive performance as long as underfitting does not occur: performance is comparable when using only 10 features. However, the performance of SFA is sensitive to the choice of the number of factors, K. The performance of the sparse nonparametric model (NSFA) is comparable to the sparse finite model

when an appropriate number of features is chosen, but avoids the time consuming model selection process. Fokoue's method (FOK) was run with $K = 20$ and various settings of the hyperparameter d which controls the overall sparsity of the solution. The model's predictive performance depends strongly on the setting of this parameter, with results approaching the performance of the sparse models (SFA and NSFA) for $d = 10^{-4}$. The performance of BFRM on this dataset is noticeably worse than the other sparse models.

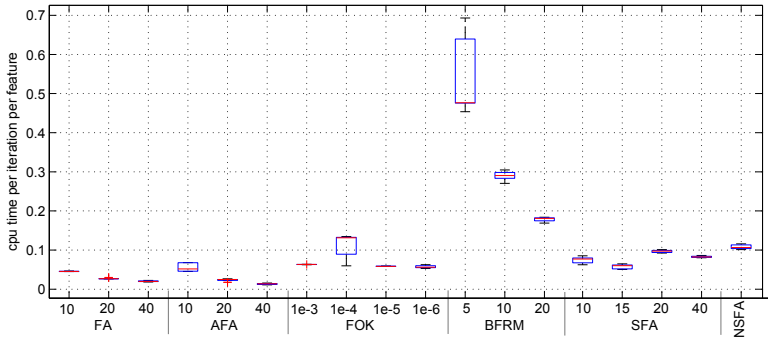
We now consider the *computation cost* of the algorithms. As described in Section 3.2, sampling $\mathbf{Z}$ and $\mathbf{G}$ takes order $O(NKD)$ operations per iteration, and sampling $\mathbf{X}$ takes $O(K^2D + K^3 + KND)$. However, for the relatively small sample sizes, N encountered for datasets 1 and 2 the main computational cost is sampling the non-zero elements of $\mathbf{G}$, which takes $O((1-s)DK)$ where s is the sparsity of the model. Figure 3.8c shows the mean CPU time per iteration divided by the number of features at that iteration. Naturally, straight FA is the fastest, taking only around $0.025s$ per iteration per feature. The value increases slightly with increasing K, suggesting that here the $O(K^2D + K^3)$ calculation and inversion of $\boldsymbol{\lambda}$, the precision of the conditional on $\mathbf{X}$, must be contributing. The computational cost of adding the ARD prior is negligible (AFA). The CPU time per iteration is just over double for the sparse finite model (SFA), but the cost actually decreases with increasing K, because the sparsity of the solution increases to avoid overfitting. There are fewer non-zero elements of $\mathbf{G}$ to sample per feature, so the CPU time *per feature* decreases. The CPU time per iteration per feature for the non-parametric sparse model (NSFA) is somewhat higher than for the finite model because of the cost of the feature birth and death process. However, Figure 3.8b shows the absolute CPU time per iteration, where we see that the nonparametric model is only marginally more expensive than the finite model of appropriate size $\hat{K} = 15$ and cheaper than choosing an unnecessarily large finite model (SFA with $K = 20, 40$). Fokoue's method (FOK) has comparable computational performance to the sparse finite model, but interestingly has increased cost for the optimal setting of $d = 10^{-4}$. The parameter space for FOK is continuous, making search easier but requiring a normal random variable for every element of $\mathbf{G}$. BFRM pays a considerable computational cost for both the hierarchical sparsity prior and the DP prior on $\mathbf{X}$. SPCA was not run on this



(a) Predictive performance: log likelihood of test (the 10% missing) data under each model based on the last 100 MCMC samples. Higher values indicate better performance. The boxplots show variation across 10 different random splits of the data into training and test sets.



(b) CPU time (in seconds) per iteration, averaged across the 3000 iteration run.



(c) CPU time (in seconds) per iteration divided by the number of features at that iteration, averaged across all iterations.

Figure 3.8: Results on breast cancer dataset ($N = 251, D = 226$, 3000 MCMC iterations).

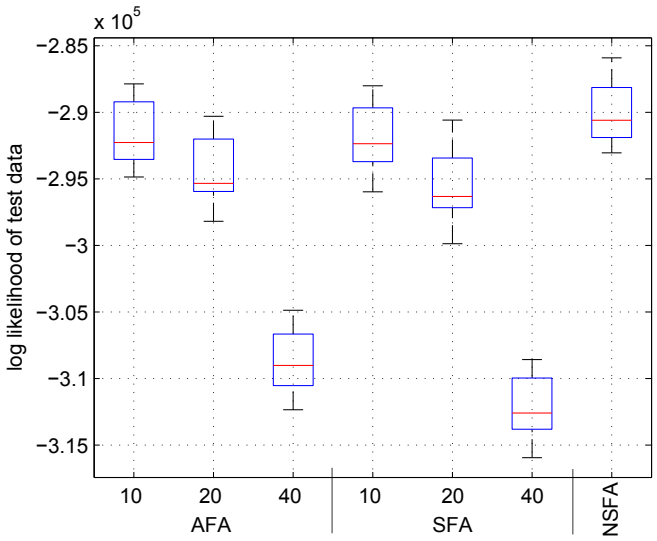


Figure 3.9: Test set log likelihoods on Prostate cancer dataset from Yu & et al. Landsittel [2004], including 12557 genes across 171 individuals (1000 MCMC iterations).

dataset but results on the synthetic data in Section 3.3.1 suggest it is somewhat faster than the sampling methods, but not hugely so. The computational cost of SPCA is $ND^2 + mO(D^2K + DK^2 + D^3)$ in the $N > D$ case (where m is the number of iterations to convergence) and $ND^2 + mO(D^2K + DK^2)$ in the $D > N$ case taking the limit $\lambda \to \infty$. In either case an individual iteration of SPCA is more expensive than one sampling iteration of NSFA (since $K < D$) but fewer iterations will generally be required to reach convergence of SPCA than are required to ensure mixing of NSFA.

3.3.5 Prostate cancer dataset from Yu & et al. Landsittel [2004]

The predictive performance of AFA, FOK and NSFA on the prostate cancer dataset of Yu & et al. Landsittel [2004], for ten random splits into training and test data, is shown in Figure 3.9. The boxplots show variation from ten random splits into training and test data. The large number of genes ($D = 12557$ across $N = 171$ individuals) in this dataset makes inference slower, but the problem is

manageable since the computational complexity is linear in the number of genes. Despite the large number of genes, the appropriate number of latent factors, in terms of maximizing predictive performance, is still small, around 10 (NSFA infers a median of 12 factors). This may seem small relative to the number of genes, especially in comparison to the breast cancer and *E. Coli* datasets. However it should be noted that the genes included in those datasets are those capturing the most variability, whereas here *all* genes on the microarray are included, the majority of which are not expected to have significant signal. Surprisingly, SFA actually performed slightly worse on this dataset than AFA. Both are highly sensitive to the number of latent factors chosen. NSFA however gives better predictive log likelihoods than either finite model for any fixed number of latent factors K. Running 1000 iterations of NSFA on this dataset takes under 8 hours. BFRM and FOK were impractically slow to run on this dataset.

3.4 Discussion

We have seen that in the *E. Coli*, breast cancer and prostate cancer datasets that sparsity can improve predictive performance, as well as providing a more easily interpretable solution. Using the IBP to provide sparsity is straightforward, and allows the number of latent factors to be inferred within a well defined theoretical framework. This has several advantages over manually choosing the number of latent factors. Choosing too few latent factors damages predictive performance, as seen for the breast cancer dataset. Although choosing too many latent factors can be compensated for by using appropriate ARD-like priors, we find this is typically more computationally expensive than the birth and death process of the IBP. Manual model selection is an alternative but is time consuming. Finally on the prostate cancer dataset we show that running NSFA on full gene expression datasets with 10,000+ genes is feasible so long as the number of latent factors remains relatively small. In Doshi-Velez *et al.* [2009a] we show that it is possible to derive a parallel sampling algorithm for such models. Each node in the cluster owns a subset of the data, and information is propagated by message passing between the nodes.

Our results corroborate the findings of Mohamed *et al.* [2011], in the case of

a Gaussian likelihood, that spike-and-slab priors give superior predictive performance compared to "soft" sparsity schemes such as using student-t priors (FOK) or L1 regularisation (SPCA). Mohamed *et al.* [2011] did not consider learning the latent dimensionality K, although their results clearly suggest this would be beneficial (which our results confirm). Combining the methodology of Mohamed *et al.* [2011] to model different data types with the techniques presented here to infer the latent dimensionality could be a fruitful line of investigation.

An interesting direction for future research is how to incorporate prior knowledge, for example if certain transcription factors are known to regulate specific genes. Incorporating this knowledge could both improve the performance of the model and improve interpretability by associating latent variables with specific transcription factors. Another possibility is incorporating correlations in the Indian Buffet Process, which has been proposed for simpler models [Courville *et al.*, 2009; Doshi-Velez & Ghahramani, 2009b]. This would be appropriate in a gene expression setting where multiple transcription factors might be expected to share sets of regulated genes due to common motifs. Unfortunately, performing MCMC in all but the simplest of these models suffers from slow mixing.

Chapter 4

Pitman Yor Diffusion Trees

A proportion of the work in this chapter was published in Knowles & Ghahramani [2011b]. In this chapter we introduce the Pitman Yor Diffusion Tree (PYDT), a Bayesian non-parametric prior over tree structures which generalises the Dirichlet Diffusion Tree [Neal, 2001] and removes the restriction to binary branching structure. The generative process is described and shown to result in an exchangeable distribution over data points. We prove some theoretical properties of the model including showing its construction as the continuum limit of a nested Chinese restaurant process model. We then present two inference methods: a collapsed MCMC sampler which allows us to model uncertainty over tree structures, and a computationally efficient greedy Bayesian EM search algorithm (full details of which are given in Chapter 7). Both algorithms use message passing on the tree structure. The utility of the model and algorithms is demonstrated on synthetic and real world data, both continuous and binary.

Tree structures play an important role in machine learning and statistics. Learning a tree structure over data points gives a straightforward picture of how objects of interest are related. Trees are easily interpreted and intuitive to understand. Sometimes we may know that there is a true hierarchy underlying the data: for example species in the tree of life or duplicates of genes in the human genome, known as paralogs. Typical mixture models, such as Dirichlet Process mixture models, have independent parameters for each component. We might expect for example that certain clusters are similar, being sub-groups of some larger group. By learning this hierarchical similarity structure, the model

can share statistical strength between components to make better estimates of parameters using less data.

Classical hierarchical clustering algorithms employ a bottom up "agglomerative" approach [Duda *et al.*, 2001] based on distances which hides the statistical assumptions being made. Heller & Ghahramani [2005] use a principled probabilistic model in lieu of a distance metric but simply view the hierarchy as a tree consistent mixture over partitions of the data. If instead a full generative model for both the tree structure and the data is used [Blei *et al.*, 2010; Neal, 2003a; Teh *et al.*, 2008; Williams, 2000] Bayesian inference machinery can be used to compute posterior distributions over the tree structures themselves.

An advantage of generative probabilistic models for trees is that they can be used as a building block for other latent variable models [Adams *et al.*, 2010; Rai & Daumé III, 2008]. We could use this technique to build topic models with hierarchies on the topics, or hidden Markov models where the states are hierarchically related. Greedy agglomerative approaches can only cluster latent variables *after* inference has been done and hence they cannot be used in a principled way to aid inference in the latent variable model.

Both heuristic and generative probabilistic approaches to learning hierarchies have focused on learning binary trees. Although computationally convenient this restriction may be undesirable: where appropriate, arbitrary trees provide a more interpretable, clean summary of the data. Some recent work has aimed to address this Adams *et al.* [2010]; Blundell *et al.* [2010], which we discuss in Section 4.1.

The Dirichlet Diffusion Tree (DDT) introduced in Neal [2003a], and reviewed in Section 4.2, is a simple yet powerful generative model which specifies a distribution on binary trees with multivariate Gaussian distributed variables at the leaves. The DDT is a Bayesian nonparametric prior, and is a generalization of Dirichlet Process mixture models [Antoniak, 1974; Rasmussen, 2000]. The DDT can be thought of as providing a very flexible density model, since the hierarchical structure is able to effectively fit non-Gaussian distributions. Indeed, in Adams *et al.* [2008] the DDT was shown to significantly outperform a Dirichlet Process mixture model in terms of predictive performance, and in fact slightly outperformed the Gaussian Process Density Sampler. The DDT also formed part of the winning strategy in the NIPS 2003 feature extraction challenge [Guyon

et al., 2005]. The DDT is thus both a mathematically elegant nonparametric distribution over hierarchies and provides state-of-the-art density estimation performance.

We introduce the Pitman Yor Diffusion Tree (PYDT), a generalization of the DDT to trees with arbitrary branching structure. While allowing atoms in the divergence function of the DDT can in principle be used to obtain multifurcating branch points [Neal, 2003a], our solution is both more flexible and more mathematically and computationally tractable. An interesting property of the PYDT is that the implied distribution over tree structures corresponds to the multifurcating Gibbs fragmentation tree [McCullagh *et al.*, 2008], a very general process generating exchangeable and consistent trees (here consistency can be understood as coherence under marginalization of subtrees).

This chapter is organised as follows. Section 4.1 briefly introduces some relevant work and background material on the DDT. In Section 4.3 we describe the generative process corresponding to the PYDT. In Section 4.4 we derive the probability of a tree and show some important properties of the process. Section 4.5 describes our hierarchical clustering models utilising the PYDT. In Section 4.6 we present an MCMC sampler. We defer to Chapter 7 for details of a greedy EM algorithm for the PYDT (and DDT), which we developed in Knowles *et al.* [2011a]. We present results demonstrating the utility of the PYDT in Section 4.7.

4.1 Related work

Most hierarchical clustering methods, both distance based [Duda *et al.*, 2001] and probabilistic [Heller & Ghahramani, 2005; Teh *et al.*, 2008], have focused on the case of binary branching structure. In Bayesian hierarchical clustering [Heller & Ghahramani, 2005] the traditional bottom-up agglomerative approach is kept but a principled probabilistic model is used to find subtrees of the hierarchy. Bayesian evidence is then used as the metric to decide which node to incorporate in the tree. An extension where the restriction to binary trees is removed is proposed in Blundell *et al.* [2010]. They use a greedy agglomerative search algorithm based on various possible ways of merging subtrees. As for Heller & Ghahramani [2005] the lack of a generative process prohibits modelling uncertainty over tree structures.

Non-binary trees are possible in the model proposed in Williams [2000] since each node independently picks a parent in the layer above, but it is necessary to pre-specify the number of layers and number of nodes in each layer. Their attempts to learn the number of nodes/layers were in fact detrimental to empirical performance. Unlike the DDT or PYDT, the model in Williams [2000] is parametric in nature, so its complexity cannot automatically adapt to the data.

The nested Chinese restaurant process has been used to define probability distributions over tree structures. In Blei *et al.* [2010] each data point is drawn from a mixture over the parameters on the path from the root to the data point, which is appropriate for mixed membership models but not standard clustering. It is possible to use the nested CRP for hierarchical clustering, but either a finite number of levels must be pre-specified, some other approach of deciding when to stop fragmenting must be used, or chains of infinite length must be integrated over Steinhardt & Ghahramani [2012]. We will show in Section 4.4.5 that the DDT and PYDT can be reconstructed as the continuum limits of particular nested CRP models.

An alternative to the PYDT to obtain unbounded trees is given by Adams *et al.* [2010], which is closely related to the nested CRP. They use a nested stick-breaking representation to construct the tree, which is then endowed with a diffusion process. At each node there is a latent probability of the current data point stopping, and so data live at internal nodes of the tree, rather than at leaves as in the PYDT. Despite being computationally appealing, this construction severely limits how much the depth of the tree can adapt to data [Steinhardt & Ghahramani, 2012].

Kingman's coalescent [Kingman, 1982; Teh *et al.*, 2008] is similar to the Dirichlet Diffusion Tree in spirit although the generative process is defined going backwards in time as datapoints coalesce together, rather than forward in time as for the DDT. Kingman's coalescent is the dual process to the Dirichlet diffusion tree, in the following sense. Imagine we sample a partition of $[n]$ from the Chinese restaurant process with hyperparameter α, coalesce this partition for a small time dt, and then "fragment" the resulting partition according to the DDT with constant rate function for time dt. The final partition will be CRP distributed with concentration α, showing that the DDT fragmentation has "undone" the

effect of the coalescent process. This duality is used in Teh *et al.* [2011] to define a partition valued stochastic process through time. Although the generalisation of Kingman's coalescent to arbitrary branching structures has been studied in the probability literature under the name Λ-coalescent [Pitman, 1999; Sagitov, 1999], it has not to our knowledge been used as a statistical model conditioned on data.

4.2 The Dirichlet Diffusion Tree

The Dirichlet Diffusion Tree was introduced in Neal [2003a] as a top-down generative model for trees over N datapoints $x_1, x_2, \cdots, x_N \in \mathbb{R}^D$. We will describe the generative process for the data in terms of a diffusion process in fictitious "time" on the unit interval. The observed data points (or latent variables) correspond to the locations of the diffusion process at time $t = 1$. The first datapoint starts at time 0 at the origin in a D-dimensional Euclidean space and follows a Brownian motion with variance σ^2 until time 1. If datapoint 1 is at position $x_1(t)$ at time t, the point will reach position $x_1(t + dt) \sim \mathrm{N}(x_1(t), \sigma^2 I dt)$ at time $t + dt$. It can easily be shown that $x_1(t) \sim \mathrm{Normal}(0, \sigma^2 I t)$. The second point x_2 in the dataset also starts at the origin and initially follows the path of x_1. The path of x_2 will diverge from that of x_1 at some time T_d after which x_2 follows a Brownian motion independent of $x_1(t)$ until $t = 1$, with $x_i(1)$ being the i-th data point. In other words, the infinitesimal increments for the second path are equal to the infinitesimal increments for the first path for all $t < T_d$. After T_d, the increments for the second path $\mathrm{N}(0, \sigma^2 I dt)$ are independent. The probability of diverging in an interval $[t, t + dt]$ is determined by a "divergence function" $a(t)$ (see Equation 4.9 below) which is analogous to the hazard function in survival analysis.

The generative process for datapoint i is as follows. Initially $x_i(t)$ follows the path of the previous datapoints. If at time t the path of $x_i(t)$ has not diverged, it will diverge in the next infinitesimal time interval $[t, t + dt]$ with probability

$$\frac{a(t)dt}{m} \tag{4.1}$$

where m is the number of datapoints that have previously followed the current

path. The division by m is a reinforcing aspect of the DDT: the more datapoints follow a particular branch, the more likely subsequent datapoints will not diverge off this branch (this division is also required to ensure exchangeability). If x_i does not diverge before reaching a previous branching point, the previous branches are chosen with probability proportional to how many times each branch has been followed before. This reinforcement scheme is similar to the Chinese restaurant process [Aldous, 1983]. For the single data point $x_i(t)$ this process is iterated down the tree until divergence, after which $x_i(t)$ performs independent Brownian motion until time $t = 1$. The i-th observed data point is given by the location of this Brownian motion at $t = 1$, i.e. $x_i(1)$.

For the purpose of this chapter we use the divergence function $a(t) = \frac{c}{1-t}$, with "smoothness" parameter $c > 0$. Larger values $c > 1$ give smoother densities because divergences typically occur earlier, resulting in less dependence between the datapoints. Smaller values $c < 1$ give rougher more "clumpy" densities with more local structure since divergence typically occurs later, closer to $t = 1$. We refer to Neal [2001] for further discussion of the properties of this and other divergence functions. Figure 4.1 illustrates the Dirichlet diffusion tree process for a dataset with $N = 4$ datapoints.

The probability of generating the tree, latent variables and observed data under the DDT can be decomposed into two components. The first component specifies the distribution over the tree structure and the divergence times. The second component specifies the distribution over the specific locations of the Brownian motion when the tree structure and divergence times are given.

Before we describe the functional form of the DDT prior we will need two results. First, the probability that a new path does not diverge between times $s < t$ on a segment that has been followed m times by previous data-points can be written as

$$P(\text{not diverging}) = \exp\left[(A(s) - A(t))/m\right], \tag{4.2}$$

where $A(t) = \int_0^t a(u)du$ is the cumulative rate function. For our divergence function $A(t) = -c\log(1-t)$. Second, the DDT prior defines an exchangeable distribution: the order in which the datapoints were generated does not change

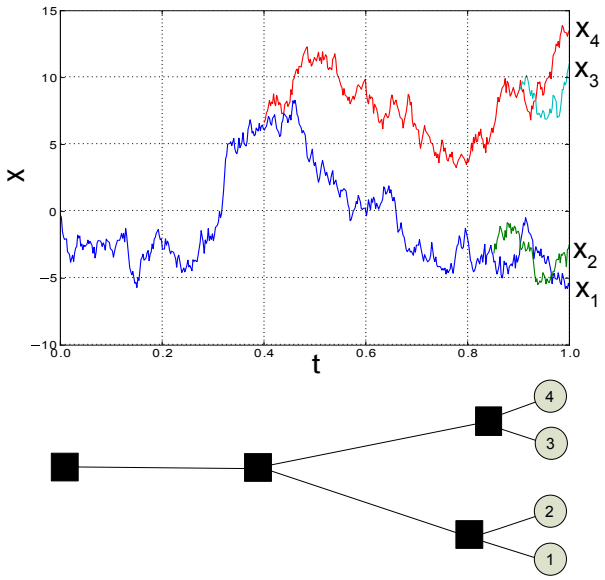


Figure 4.1: A sample from the Dirichlet Diffusion Tree with $N = 4$ datapoints. Top: the location of the Brownian motion for each of the four paths. Bottom: the corresponding tree structure. Each branch point corresponds to an internal tree node.

the joint density. See Neal [2003a] for a proof.

We now consider the tree as a set of segments $\mathcal{S}(\mathcal{T})$ each contributing to the joint probability density. The tree structure $\mathcal{T}$ contains the counts of how many datapoints traversed each segment. Consider an arbitrary segment $[ab] \in \mathcal{S}(\mathcal{T})$ from node a to node b with corresponding locations x_a and x_b and divergence times t_a and t_b, where $t_a < t_b$. Let $m(b)$ be the number of leaves under node b, i.e. the number of datapoints which traversed segment $[ab]$. Let $l(b)$ and $r(b)$ be the number of leaves under the left and right child of node b respectively, so that $l(b) + r(b) = m(b)$.

By exchangeability we can assume that it was the second path which diverged at b. None of the subsequent paths that passed through a diverged before time t_b (otherwise $[ab]$ would not be a contiguous segment). The probability of this

happening is

$$P(t_b|[ab],t_a) = \overbrace{\frac{a(t_b)}{1}}^{\text{2nd branch diverges}} \prod_{i=1}^{m(b)-1} \overbrace{\exp[(A(t_a)-A(t_b))/i]}^{\text{(i+1)th branch does not diverge before b}} \tag{4.3}$$

$$= a(t_b)\exp\left[(A(t_a)-A(t_b))H_{m(b)-1}\right], \tag{4.4}$$

where $H_n = \sum_{i=1}^{n} 1/i$ is the nth harmonic number. This expression factorizes into a term for t_a and t_b. Collecting such terms from the branches attached to an internal node i the factor for t_i for the divergence function $a(t) = c/(1-t)$ is

$$\begin{aligned} &a(t_i)e^{[A(t_i)(H_{l(i)-1}+H_{r(i)-1}-H_{m(i)-1})]} \\ &= c(1-t_i)^{cJ_{l(i),r(i)}-1}, \end{aligned} \tag{4.5}$$

where $J_{l,r} = H_{r+l-1} - H_{l-1} - H_{r-1}$.

Each path that went through x_b, except the first and second, had to choose to follow the left or right branch. Again, by exchangeability, we can assume that all $l(b)-1$ paths took the left branch first, then all $r(b)-1$ paths chose the right branch. The probability of this happening is

$$P([ab]) = \frac{(l(b)-1)!(r(b)-1)!}{(m(b)-1)!}. \tag{4.6}$$

Finally, we include a term for the diffusion locations:

$$P(x_b|x_a,t_a,t_b) = \mathrm{N}(x_b; x_a, \sigma^2(t_b-t_a)). \tag{4.7}$$

The full joint probability for the DDT is now a product of terms for each segment

$$P(x,t,\mathcal{T}) = \prod_{[ab]\in\mathcal{S}(\mathcal{T})} P(x_b|x_a,t_a,t_b)P(t_b|[ab],t_a)P([ab]). \tag{4.8}$$

4.3 Generative process for the PYDT

The PYDT generative process is analogous to that for the DDT, but altered to allow arbitrary branching structures. Firstly, the probability of diverging from a

branch having previously been traversed by m data points in interval $[t, t+dt]$ is given by

$$\frac{a(t)\Gamma(m-\alpha)dt}{\Gamma(m+1+\theta)} \tag{4.9}$$

where $\Gamma(.)$ is the standard Gamma function, and $0 \leq \alpha \leq 1, \theta \geq -2\alpha$ are parameters of the model (see Section 4.4.2 for further discussion of allowable parameter ranges). When $\theta = \alpha = 0$ we recover binary branching and the DDT expression in Equation 4.1. Secondly, if x_i does not diverge before reaching a previous branching point, it may either follow one of the previous branches, or diverge at the branch point (adding one to the degree of this node in the tree). The probability of following one of the existing branches k is

$$\frac{b_k - \alpha}{m + \theta} \tag{4.10}$$

where b_k is the number of samples which previously took branch k and m is the total number of samples through this branch point so far. The probability of diverging at the branch point and creating a new branch is

$$\frac{\theta + \alpha K}{m + \theta} \tag{4.11}$$

where K is the current number of branches from this branch point. By summing Equation 4.10 over $k = \{1, \ldots, K\}$ with Equation 4.11 we get 1, since $\sum_k b_k = m$, as required. This reinforcement scheme is analogous to the Pitman Yor process [Pitman & Yor, 1997; Teh, 2006] version of the Chinese restaurant process [Aldous, 1983].

4.3.1 Sampling the PYDT in practice

It is straightforward to sample from the PYDT prior. This is most easily done by sampling the tree structure and divergence times first, followed by the divergence locations. We will need the inverse cumulative divergence function, e.g. $A^{-1}(y) = 1.0 - \exp(-y/c)$ for the divergence function $a(t) = c/(1-t)$.

Each point starts at the root of the tree. The cumulative distribution function

for the divergence time of the i-th sample is

$$C(t) = 1 - \exp\left\{-A(t)\frac{\Gamma(i-1-\alpha)}{\Gamma(i+\theta)}\right\} \tag{4.12}$$

We can sample from this distribution by drawing $U \sim \text{Uniform}[0,1]$ and setting

$$t_d = C^{-1}(U) := A^{-1}\left(-\frac{\Gamma(i+\theta)}{\Gamma(i-1-\alpha)}\log(1-U)\right) \tag{4.13}$$

If t_d is actually past the next branch point, we diverge at this branch point or choose one of the previous paths with the probabilities defined in Equations 4.11 and 4.10 respectively. If we choose one of the existing branches then we must again sample a divergence time. On an edge from node a to b previously traversed by $m(b)$ data points, the cumulative distribution function for a new divergence time is

$$C(t) = 1 - \exp\left\{-[A(t) - A(t_a)]\frac{\Gamma(m(b)-\alpha)}{\Gamma(m(b)+1+\theta)}\right\} \tag{4.14}$$

which we can sample as follows

$$t_d := A^{-1}\left(A(t_a) - \frac{\Gamma(m(b)+1+\theta)}{\Gamma(m(b)-\alpha)}\log(1-U)\right) \tag{4.15}$$

We do not actually need to be able to evaluate $A(t_a)$ since this will necessarily have been calculated when sampling t_a. If $t_d > t_b$ we again choose whether to follow an existing branch or diverge according to Equations 4.11 and 4.10.

Given the tree structure and divergence times sampling the locations simply involves a sweep down the tree sampling $x_b \sim N(x_a, \sigma^2(t_b - t_a)I)$ for each branch $[ab]$.

4.4 Theory

Now we present some important properties of the PYDT generative process.

4.4.1 Probability of a tree

We refer to branch points and leaves of the tree as nodes. The probability of generating a specific tree structure with associated divergence times and locations at each node can be written analytically since the specific diffusion path taken between nodes can be ignored. We will need the probability that a new data point does not diverge between times $s < t$ on a branch that has been followed m times by previous data-points. This can straightforwardly be derived from Equation 4.9:

$$P\left(\begin{array}{c}\text{not diverging}\\ \text{in } [s,t]\end{array}\right) = \exp\left[(A(s)-A(t))\frac{\Gamma(m-\alpha)}{\Gamma(m+1+\theta)}\right], \tag{4.16}$$

where $A(t) = \int_0^t a(u)du$ is the cumulative rate function.

Consider the tree of $N = 4$ data points in Figure 4.2. The probability of obtaining this tree structure and associated divergence times is:

$$\begin{aligned} & e^{-A(t_a)\frac{\Gamma(1-\alpha)}{\Gamma(2+\theta)}}\frac{a(t_a)\Gamma(1-\alpha)}{\Gamma(2+\theta)} \\ & \times e^{-A(t_a)\frac{\Gamma(2-\alpha)}{\Gamma(3+\theta)}}\frac{1-\alpha}{2+\theta}e^{[A(t_a)-A(t_b)]\frac{\Gamma(1-\alpha)}{\Gamma(2+\theta)}}\frac{a(t_b)\Gamma(1-\alpha)}{\Gamma(2+\theta)} \\ & \times e^{-A(t_a)\frac{\Gamma(3-\alpha)}{\Gamma(4+\theta)}}\frac{\theta+2\alpha}{3+\theta}. \end{aligned} \tag{4.17}$$

The first data point does not contribute to the expression. The second point contributes the first line: the first term results from not diverging between $t = 0$ and t_a, the second from diverging at t_a. The third point contributes the second line: the first term comes from not diverging before time t_a, the second from choosing the branch leading towards the first point, the third term comes from not diverging between times t_a and t_b, and the final term from diverging at time t_b. The fourth and final data point contributes the final line: the first term for not diverging before time t_a and the second term for diverging at branch point a.

Although not immediately obvious, we will see in Section 4.4.3, the tree probability in Equation 4.17 is invariant to reordering of the data points.

The component of the joint probability distribution resulting from the branch-

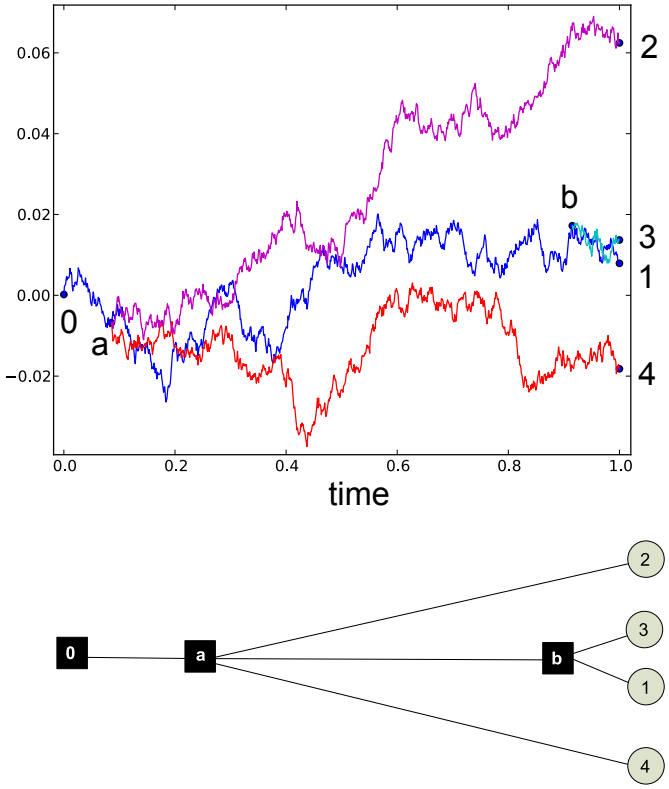


Figure 4.2: A sample from the Pitman-Yor Diffusion Tree with $N = 4$ datapoints and $a(t) = 1/(1-t), \theta = 1, \alpha = 0$. Top: the location of the Brownian motion for each of the four paths. Bottom: the corresponding tree structure. Each branch point corresponds to an internal tree node.

ing point and data locations for the tree in Figure 4.2 is

$$\begin{aligned} &N(x_a; 0, \sigma^2 t_a) N(x_b; x_a, \sigma^2(t_b - t_a)) \\ \times &N(x_1; x_b, \sigma^2(1-t_b)) N(x_2; x_a, \sigma^2(1-t_a)) \\ \times &N(x_3; x_b, \sigma^2(1-t_b)) N(x_4; x_a, \sigma^2(1-t_a)) \end{aligned} \tag{4.18}$$

where we see there is a Gaussian term associated with each branch in the tree.

4.4.2 Parameter ranges and branching degree

There are several valid ranges of the parameters (θ, α):

- $0 \leq \alpha < 1$ and $\theta > -2\alpha$. This is the general multifurcating case with arbitrary branching degree which we will be most interested in (although in

fact we will often restrict further to $\theta > 0$). $\alpha < 1$ ensures the probability of going down an existing branch is non-negative in Equation 4.10. $\theta > -2\alpha$ and $\alpha > 0$ together ensure that the probability of forming a new branch is non-negative for any K in Equation 4.11.

- $\alpha < 0$ and $\theta = -\kappa\alpha$ where $\kappa \in \mathbb{Z}$ and $\kappa \geq 3$. Here κ is the maximum number of children a node can have since the probability of forming a new branch at a node with $K = \kappa$ existing branches given by Equation 4.11 will be zero. We require $\alpha < 0$ to ensure the probability of following an existing branch is always positive.

- $\alpha < 1$ and $\theta = -2\alpha$. This gives binary branching, and specifically the DDT for $\alpha = \theta = 0$. Interestingly however we see that this gives a parameterised family of priors over binaries trees, which was in fact proposed by MacKay & Broderick [2007].

There is another degenerate case which is of little interest for statistical modeling: with $\alpha = 1$ we always have instantaneous divergence at time $t = 0$ (since the numerator in Equation 4.9 contains the term $\Gamma(m - \alpha)$) so every data point is independent.

Consider the parameter range $0 \leq \alpha < 1$ and $\theta > -2\alpha$. By varying θ we can move between flat (large θ) and hierarchical clusterings (small θ), as shown in Figure 4.3 (here we have fixed $\alpha = 0$).

Now consider the binary branching parameter range $\alpha < 1$ and $\theta = -2\alpha$ which is a special case of the PYDT but still generalises the DDT. As mentioned in MacKay & Broderick [2007] the parameter α controls the balance of the tree. In fact, the reinforcing scheme here can be considered as the result of marginalising out a latent variable, p_b at every internal node, b with prior, $p_b \sim \text{Beta}(-\alpha, -\alpha)$. For $\alpha = 0$ this is a uniform distribution. For α close to 1 the distribution will concentrate towards point masses at 0 and 1, i.e. towards $(\delta(0) + \delta(1))/2$, so that one branch will be greatly preferred over the other, making the tree more unbalanced. As $\alpha \to -\infty$ the mass of the beta distribution concentrates towards a point mass at 0.5 encouraging the tree to be more balanced. A simple measure

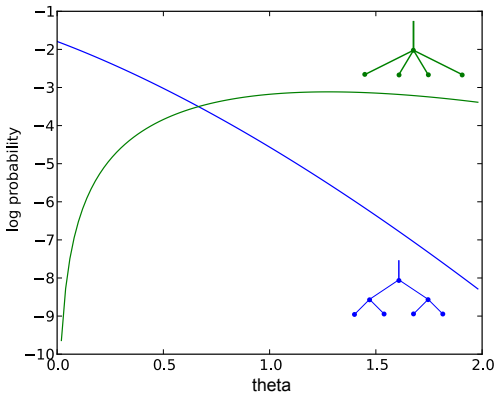


Figure 4.3: The effect of varying θ on the log probability of two tree structures (i.e. the product of the terms in Equation 4.23 over the segments in the tree), indicating the types of tree preferred. Small $\theta < 1$ favors binary trees while larger values of θ favors higher order branching points.

of the imbalance of tree is given by Colless's I [Colless, 1982], given by

$$I = \frac{2}{(n-1)(n-2)} \sum_{a \in \mathcal{T}} |l(a) - r(a)| \tag{4.19}$$

where n is the number of leaves, a ranges over all internal nodes in the tree, and $l(a)$ and $r(a)$ are the number of data points that followed the left and right branches respectively. An alternative is the normalised number of unbalanced nodes in a tree, J [Rogers, 1996], i.e.

$$J = \frac{1}{(n-2)} \sum_{a \in \mathcal{T}} (1 - \mathbb{I}[l(a) = r(a)]) \tag{4.20}$$

where $\mathbb{I}$ is the indicator function. As expected, in Figure 4.4 we see that both measures of tree imbalance increase with α, with the biggest effects occuring in the interval $[0, 1]$.

4.4.3 Exchangeability

Exchangeability is both a key modelling assumption and a property that greatly simplifies inference. We show that analogously to the DDT, the PYDT defines

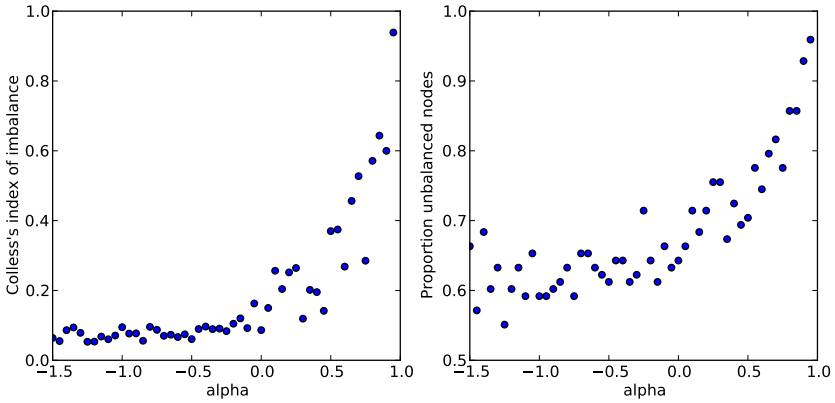


Figure 4.4: Two measures of tree imbalance for samples from the binary Pitman-Yor Diffusion Tree with $\theta = -2\alpha$ for varying α. Generated trees had $N = 100$ and the tree structure itself is invariant to the divergence function. **Left**: Colless's index of balance, see Equation 4.19. **Right**: Proportion of unbalanced nodes, see Equation 4.20.

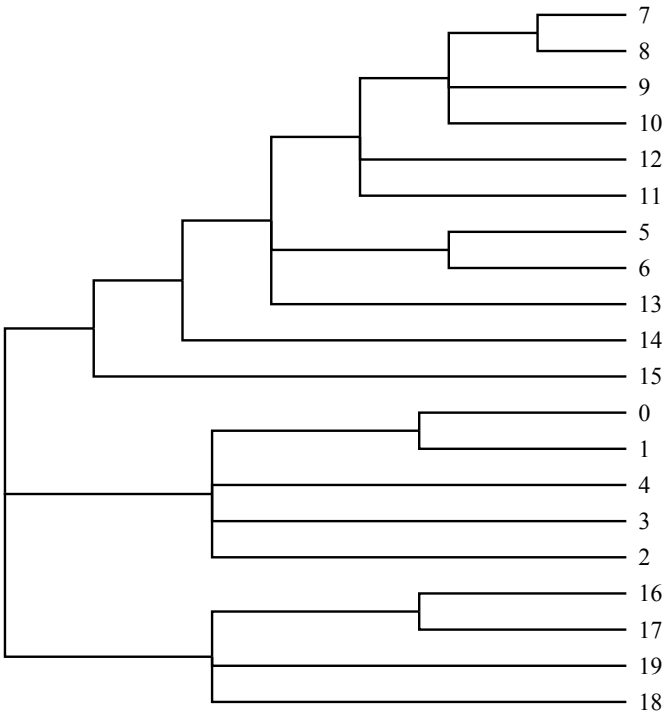
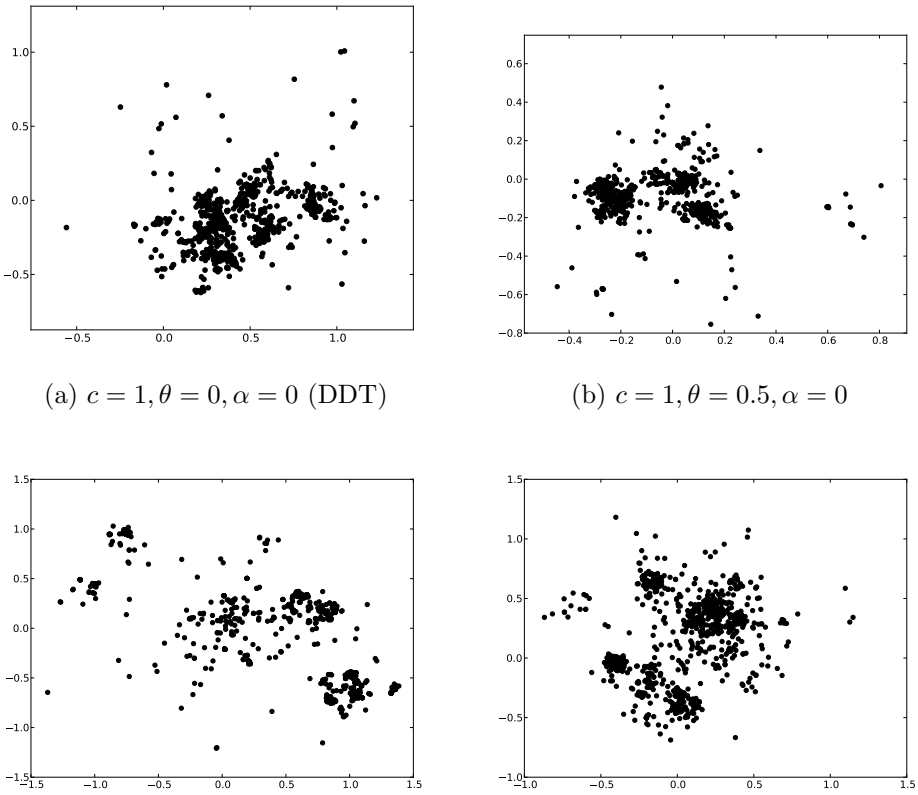


Figure 4.5: A sample from the Pitman-Yor Diffusion Tree with $N = 20$ datapoints and $a(t) = 1/(1-t), \theta = 1, \alpha = 0$ showing the branching structure including non-binary branch points.



(a) $c = 1, \theta = 0, \alpha = 0$ (DDT)

(b) $c = 1, \theta = 0.5, \alpha = 0$

(c) $c = 1, \theta = 1, \alpha = 0$

(d) $c = 3, \theta = 1.5, \alpha = 0$

Figure 4.6: Samples from the Pitman-Yor Diffusion Tree with $N = 1000$ datapoints in $D = 2$ dimensions and $a(t) = c/(1-t)$. As θ increases more obvious clusters appear.

a infinitely exchangeable distribution over the data points. We first need the following lemma.

Lemma 1. *The probability of generating a specific tree structure, divergence times, divergence locations and corresponding data set is invariant to the ordering of data points.*

Proof. The probability of a draw from the PYDT can be decomposed into three components: the probability of the underlying tree structure, the probability of the divergence times given the tree structure, and the probability of the divergence locations given the divergence times. We will show that none of these components depend on the ordering of the data. Consider the tree, $\mathcal{T}$ as a set of edges, $\mathcal{S}(\mathcal{T})$ each of which we will see contributes to the joint probability density. The tree structure $\mathcal{T}$ contains the counts of how many datapoints traversed each edge. We denote an edge by $[uv] \in \mathcal{S}(\mathcal{T})$, which goes from node u to node v with corresponding locations x_u and x_v and divergence times t_u and t_v. Let the final number of branches from v be K_v, and the number of samples which followed each branch be $\{n_k^v : k \in [1 \dots K_v]\}$. The total number of datapoints which traversed edge $[uv]$ is $m(v) = \sum_{j=1}^{K_v} n_k^v$. Denote by $\mathcal{S}'(\mathcal{T}) = \{[uv] \in \mathcal{S}(\mathcal{T}) : m(v) \geq 2\}$ the set of all edges traversed by $m \geq 2$ samples (for divergence functions which ensure divergence before time 1 this is the set of all edges not connecting to leaf nodes).

Probability of the tree structure. For segment $[uv]$, let i be the index of the sample which diverged to create the branch point at v. The first $i-1$ samples did not diverge at v so only contribute terms for not diverging (see Equation 4.24 below). From Equation 4.9, the probability of the i-th sample having diverged to form the branch point is

$$\frac{a(t_v)\Gamma(i-1-\alpha)}{\Gamma(i+\theta)}. \tag{4.21}$$

We now wish to calculate the probability of final configuration of the branch point. Following the divergence of sample i there are $K_v - 2$ samples that form new branches from the same point, which from Equation 4.11 we see contribute $\theta + (k-1)\alpha$ to the numerator for $k \in \{3, \dots, K_v\}$. Let c_l be the number of samples having previously followed path l, so that c_l ranges from 1 to $n_l^v - 1$,

which by Equation 4.10 contributes a term $\prod_{c_l=1}^{n_l^v-1}(c_l-\alpha)$ to the numerator for $l=2,...,K_v$. Note that c_1 only ranges from $i-1$ to n_1^v-1, thereby contributing a term $\prod_{c_1=i-1}^{n_1^v-1}(c_1-\alpha)$. The j-th sample contributes a factor $j-1+\theta$ to the denominator, regardless of whether it followed an existing branch or created a new one, since the denominator in Equations 4.11 and 4.10 are equal. The factor associated with this branch point is then:

$$\begin{aligned}
&\frac{\prod_{k=3}^{K_v}[\theta+(k-1)\alpha]\prod_{c_1=i-1}^{n_1^v-1}(c_1-\alpha)\prod_{l=2}^{K_v}\prod_{c_l=1}^{n_l^v-1}(c_l-\alpha)}{\prod_{j=i+1}^{m(v)}(j-1+\theta)} \\
&= \frac{\prod_{k=3}^{K_v}[\theta+(k-1)\alpha]\prod_{l=1}^{K_v}\prod_{c_l=1}^{n_l^v-1}(c_l-\alpha)}{\prod_{j=i+1}^{m(v)}(j-1+\theta)\prod_{c_1=1}^{i-2}(c_1-\alpha)} \\
&= \frac{\prod_{k=3}^{K_v}[\theta+(k-1)\alpha]\Gamma(i+\theta)\prod_{l=1}^{K_v}\Gamma(n_l^v-\alpha)}{\Gamma(m(v)+\theta)\Gamma(i-1-\alpha)\Gamma(1-\alpha)^{K_v-1}}.
\end{aligned} \tag{4.22}$$

Multiplying by the contribution from data point i in Equation 4.21 we have

$$\frac{a(t_v)\prod_{k=3}^{K_v}[\theta+(k-1)\alpha]\prod_{l=1}^{K_v}\Gamma(n_l^v-\alpha)}{\Gamma(m(v)+\theta)\Gamma(1-\alpha)^{K_v-1}}. \tag{4.23}$$

Each segment $[uv]\in\mathcal{S}'(\mathcal{T})$ contributes such a term. Since this expression does not depend on the ordering of the branching events (that is, on the index i), the overall factor does not either. Note that since $a(t_v)$ is a multiplicative factor we can think of this as part of the probability factor for the divergence times.

Probability of divergence times. The $m(v)-1$ points that followed the first point along this path did not diverge before time t_v (otherwise $[uv]$ would not be an edge), which from Equation 4.16 we see contributes a factor

$$\begin{aligned}
&\prod_{i=1}^{m(v)-1}\exp\left[(A(t_u)-A(t_v))\frac{\Gamma(i-\alpha)}{\Gamma(i+1+\theta)}\right] \\
&= \exp\left[(A(t_u)-A(t_v))H_{m(v)-1}^{\theta,\alpha}\right],
\end{aligned} \tag{4.24}$$

where we define $H_n^{\theta,\alpha}=\sum_{i=1}^{n}\frac{\Gamma(i-\alpha)}{\Gamma(i+1+\theta)}$. All edges $[uv]\in\mathcal{S}'(\mathcal{T})$ contribute the

expression in Equation 4.24, resulting in a total contribution

$$\prod_{[uv]\in\mathcal{S}'(\mathcal{T})} \exp\left[(A(t_u) - A(t_v))H^{\theta,\alpha}_{m(v)-1}\right]. \tag{4.25}$$

This expression does not depend on the ordering of the datapoints.

Probability of node locations. Generalizing Equation 4.18 it is clear that each edge contributes a Gaussian factor, resulting an overall factor:

$$\prod_{[uv]\in\mathcal{S}(\mathcal{T})} \mathrm{N}(x_v; x_u, \sigma^2(t_v - t_u)I). \tag{4.26}$$

The overall probability of a specific tree, divergence times and node locations is given by the product of Equations 4.23, 4.25 and 4.26, none of which depend on the ordering of the data. □

The term $\prod_{k=3}^{K_v}[\theta + (k-1)\alpha]$ in Equation 4.23 can be calculated efficiently depending on the value of α. For $\alpha = 0$ we have $\prod_{k=3}^{K_v}\theta = \theta^{K_v-2}$. For $\alpha \neq 0$ we have

$$\begin{aligned}\prod_{k=3}^{K_v}[\theta + (k-1)\alpha] &= \alpha^{K_v-2}\prod_{k=3}^{K_v}[\theta/\alpha + (k-1)] \\ &= \frac{\alpha^{K_v-2}\Gamma(\theta/\alpha + K_v)}{\Gamma(\theta/\alpha + 2)}.\end{aligned} \tag{4.27}$$

It is also useful to note that the factor for the divergence times in Equation 4.25 itself factorizes into a term for t_u and t_v. Collecting such terms from the branches attached to an internal node v the factor for t_v for the divergence function $a(t) = c/(1-t)$ is

$$\begin{aligned}P(t_v|\mathcal{T}) &= a(t_v)\exp\left[A(t_v)\left(\sum_{k=1}^{K_v} H^{\theta,\alpha}_{n^v_k-1} - H^{\theta,\alpha}_{m(v)-1}\right)\right] \\ &= c(1-t_v)^{cJ^{\theta,\alpha}_{\mathbf{n}^v}-1}\end{aligned} \tag{4.28}$$

where $J^{\theta,\alpha}_{n^v} = H^{\theta,\alpha}_{\sum_{k=1}^K n^v_k - 1} - \sum_{k=1}^K H^{\theta,\alpha}_{n^v_k-1}$ with $\mathbf{n} \in \mathbb{N}^K$ being the number of datapoints having gone down each branch. Equation 4.28 is the generalisation of

Equation 4.5 for the DDT to the PYDT. It is interesting to note that a priori the divergence times are independent apart from the constraint that branch lengths must be non-negative.

Theorem 1. *The Pitman-Yor Diffusion Tree defines an infinitely exchangeable distribution over data points.*

Proof. Summing over all possible tree structures, and integrating over all branch point times and locations, by Lemma 1 we have exchangeability for any finite number of datapoints, N. As a virtue of its sequential generative process, the PYDT is clearly projective (see Section 1.1). Being exchangeable and projective, the PYDT is infinitely exchangeable. □

Corollary 1. *There exists a prior ν on probability measures on $\mathbb{R}^D$ such that the samples $x_1, x_2, \ldots$ generated by a PYDT are conditionally independent and identically distributed (iid) according to $\mathcal{F} \sim \nu$, that is, we can represent the PYDT as*

$$PYDT(x_1, x_2, \ldots) = \int \left(\prod_i \mathcal{F}(x_i) \right) d\nu(\mathcal{F}).$$

Proof. Since the PYDT defines an infinitely exchangeable process on data points, the result follows directly by de Finetti's Theorem [Hewitt & Savage, 1955]. □

Another way of expressing Corollary 1 is that data points $x_1, \ldots, x_N$ sampled from the PYDT could equivalently have been sampled by first sampling a probability measure $\mathcal{F} \sim \nu$, then sampling $x_i \sim \mathcal{F}$ iid for all i in $\{1, \ldots, N\}$. For divergence functions such that $A(1)$ is infinite, divergence will necessarily occur before time $t = 1$, so that there is zero probability of two data points having the same location, i.e. the probability measure $\mathcal{F}$ is continuous almost surely. Note that in general we cannot guarantee that $\mathcal{F}$ will be absolutely continuous (the condition required for a density to exist).

4.4.4 Relationship to the DDT

The PYDT is a generalisation of the Dirichlet diffusion tree:

Lemma 2. *The PYDT reduces to the Diffusion Dirichlet Tree [Neal, 2001] in the case $\theta = \alpha = 0$.*

Proof. This is clear from the generative process: for $\theta = \alpha = 0$ there is zero probability of branching at a previous branch point (assuming continuous cumulative divergence function $A(t)$). The probability of diverging in the time interval $[t, t + dt]$ from a branch previously traversed by m datapoints becomes:

$$\frac{a(t)\Gamma(m-0)dt}{\Gamma(m+1+0)} = \frac{a(t)(m-1)!dt}{m!} = \frac{a(t)dt}{m}, \tag{4.29}$$

as for the DDT. □

It is straightforward to confirm that the DDT probability factors are recovered when $\theta = \alpha = 0$. In this case $K_v = 2$ since non-binary branch points have zero probability, so Equation 4.23 reduces as follows:

$$\frac{a(t_v)\prod_{l=1}^{K_v=2}\Gamma(n_l^v - 0)}{\Gamma(m(v)+0)} = \frac{a(t_v)(n_1^b-1)!(n_2^b-1)!}{(m(v)-1)!}, \tag{4.30}$$

as for the DDT. Equation 4.25 also reduces to the DDT expression since

$$H_n^{0,0} = \sum_{i=1}^{n}\frac{\Gamma(i-0)}{\Gamma(i+1+0)} = \sum_{i=1}^{n}\frac{(i-1)!}{i!} = \sum_{i=1}^{n}\frac{1}{i} = H_n, \tag{4.31}$$

where H_n is the n-th Harmonic number.

4.4.5 The continuum limit of a nested CRP

The PYDT can be derived as the limiting case of a specific nested Chinese Restaurant Process [Blei *et al.*, 2004] model (nCRP). We will first show how to construct the Dirichlet Diffusion Tree as the limit of a simple nCRP model. We then modify this model so that the limiting process is instead the PYDT.

The Chinese Restaurant Process defines a distribution over partitions of the natural numbers $\mathbb{N}$. Define a partition of $[n] = \{1, ..., n\}$ as a collection of disjoints sets (blocks or clusters) $\{B_k : k = 1, .., K\}$ such that $\cup_{k=1}^{K} B_k = [n]$. The CRP is constructed iteratively for $n = 1, 2, ...$ Data point n joins an existing block k

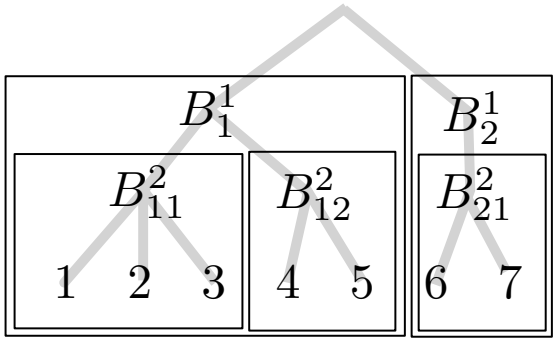


Figure 4.7: A hierarchical partitioning of the integers $\{1, ..., 7\}$ showing the underlying tree structure.

with probability

$$\frac{|B_k|}{\theta + n - 1} \tag{4.32}$$

and forms its own new block with probability

$$\frac{\theta}{\theta + n - 1}. \tag{4.33}$$

The nested CRP gives a distribution over hierarchical partitions. Denote the K blocks in the first level as $\{B_k^1 : k = 1, ..., K\}$. We can now imagine partitioning the elements in each first level block, B_k^1, according to independent CRPs. Denote the blocks in the second level partitioning of B_k^1 as $\{B_{kl}^2 : l = 1, ..., K_k\}$. We can recurse this construction for as many iterations S as we please, forming a S deep hierarchy of blocks B. Each element belongs to just a single block at each level, and the partitioning forms a tree structure: consider the unpartitioned set $[n]$ as the root, with children B_k^1. Each B_k^1 then has children B_{kl}^2, and so on down the tree, see Figure 4.7. Note that nodes with only a single child are allowed under this construction. An example draw from an $S = 10$ level nested CRP is shown in Figure 4.8. It is certainly possible to work with this model directly (see Blei *et al.* [2004, 2010] and more recently Steinhardt & Ghahramani [2012]), but there are disadvantages, such as having to choose the depth S, or avoid this in some more convoluted way: Adams *et al.* [2010] use a stick breaking representation of the nCRP with unbounded depth S augmented with a probability of stopping

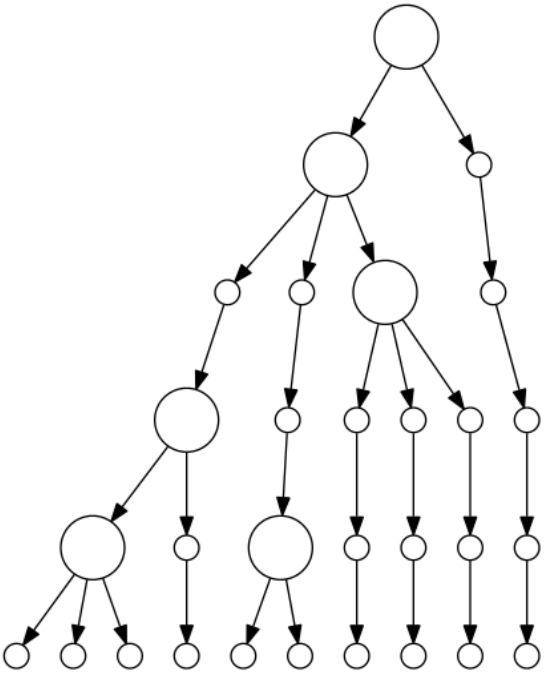


Figure 4.8: A draw from a $S = 10$-level nested Chinese restaurant process.

at each internal node, and Steinhardt & Ghahramani [2012] allow $S \to \infty$ but integrate over countably infinite chains of nodes with only one child. While the later approach is appealing for discrete data where any bounded diffusion process on the infinitely deep hierarchy concentrates towards a point mass, this is not appropriate for continuous data for example. The approach used by the DDT and PYDT of embedding in continuous time is more apt in this domain.

Theorem 2. *Associate each level s in an S-level nCRP with "time" $t_s = \frac{s-1}{S} \in [0, 1)$, and let the concentration parameter at level s be $a(t_s)/S$, where $a : [0, 1] \mapsto \mathbb{R}^+$. Taking the limit $S \to \infty$ recovers the Dirichlet Diffusion Tree [Neal, 2003a] with divergence function $a(t)$.*

Intuitively, any connected chains of nodes with only one child in the nCRP will become branches in the DDT. Nodes in the nCRP which do have multiple children become branch points in the DDT, but we find that these will always be binary splits.

Proof. From Equation 4.33 the probability of forming a new branch (block) at a

node on a chain of nodes with only single children (a single block) at level s is (from the definition of the CRP)

$$\frac{a(t_s)/S}{m + a(t_s)/S}, \tag{4.34}$$

where m is the number of previous data points that went down this chain. This behaves as $a(t_s)/(Sm)$ as S becomes large. Informally associating the time interval $1/S$ with the infinitesimal time interval dt directly yields the DDT divergence probability $a(t)dt/m$. More formally, we aim to show that the distribution over divergence times is given by the DDT in the limit $S \to \infty$. The number of nodes k in a chain starting at level b until divergence is distributed as

$$\underbrace{\frac{a(t_{b+k})/S}{m + a(t_{b+k})/S}}_{\text{prob new block at level b+k}} \prod_{i=1}^{k-1} \underbrace{\left(1 - \frac{a(t_{b+i})/S}{m + a(t_{b+i})/S}\right)}_{\text{prob not forming new block at level b+i}}, \tag{4.35}$$

where $t_b = \frac{b}{S+1}$ is the "time" of the branch point at the top of the chain. For constant $a(.)$ Equation 4.35 is a geometric distribution in k. We now take the limit $S \to \infty$, holding $\frac{k-1}{S} = t - t_b$ and $\frac{b-1}{S} = t_b$ fixed so that we also have $k \to \infty$. We analyse how the product in Equation 4.35 behaves:

$$\begin{aligned}
&\lim_{S\to\infty} \prod_{i=1}^{k-1} \left(1 - \frac{a(t_{b+i})/S}{m + a(t_{b+i})/S}\right) \\
&= \lim_{S\to\infty} \prod_{i=1}^{k-1} \left(1 - \frac{a(t_{b+i})}{Sm}\right) \\
&= \exp\left\{\lim_{k\to\infty} \sum_{i=1}^{k-1} \log\left(1 - \frac{a(t_b + i(t-t_b)/(k-1)))}{m} \frac{t-t_b}{k-1}\right)\right\}
\end{aligned} \tag{4.36}$$

where we have used that $t_{b+i} = t_b + \frac{i-1}{S}$ and $\frac{1}{S} = \frac{t-t_b}{k-1}$. We are able to exchange the order of the exp and lim operations because of the continuity of exp. Now we

use that $\log(1-x) = -x - O(x^2)$ to give

$$\log\left(1 - \frac{a(t_b + i(t-t_b)/(k-1))}{m}\frac{t-t_b}{k-1}\right) = -\frac{a(t_b + i(t-t_b)/(k-1))}{m}\frac{t-t_b}{k-1} - O(k^{-2}) \tag{4.37}$$

which allows us to see that the limiting value of the exponent in Equation 4.36 is simply a Riemann integral

$$\lim_{k\to\infty}\left[\frac{a(t_b + i(t-t_b)/(k-1)))}{m}\frac{t-t_b}{k-1} - O(k^{-2})\right] = \frac{1}{m}\int_{t_b}^{t} a(\tau)d\tau \tag{4.38}$$

Thus taking the limit $S \to \infty$ of Equation 4.35 we find the divergence time, $t = t_b + \frac{k-1}{S}$ is distributed

$$\frac{a(t)}{m}\exp\left\{-\frac{\int_{t_b}^{t} a(\tau)d\tau}{m}\right\} \tag{4.39}$$

as for the Dirichlet Diffusion Tree, the waiting time in a inhomogeneous Poisson process with rate function $a(.)$. In the simple case of constant $a(.) = a$ the geometric distribution becomes an exponential waiting time with parameter a/m.

At existing branch points the probability of going down an existing branch k is $|B_k|/(m + a(t_s)/S)$ which is simply $|B_k|/m$ in the limit $S \to \infty$, recovering the DDT. The probability of a third cluster forming at an existing branch point is given by Equation 4.34 which clearly tends to 0 in the limit, resulting in the binary nature of the DDT. □

An alternative, but more technically involved construction would use a homogeneous (constant) rate $a(.) = a$ and then use the Poisson process mapping theorem [Kingman, 1993] to transform this process into a DDT with arbitrary divergence function $a(.)$.

It was essential in this construction that we drove the concentration parameter to zero as the depth of the tree increases. This avoids complete instantaneous fragmentation of the tree. For any time $\epsilon > 0$ there will be infinitely many levels in the nCRP before time ϵ when we take $S \to \infty$. If the CRPs in these levels

have strictly positive concentration parameters, the tree will have completely fragmented to individual samples before ϵ almost surely. This is clearly undesirable from a modelling perspective since the samples are then independent.

It is interesting that despite the finite level nCRP allowing multifurcating "branch points" the continuum limit taken in Theorem 2 results in binary branch points almost surely. We will show how to rectify this limitation in Theorem 3 where we present the analogous construction for the Pitman-Yor Diffusion Tree. First we mention the possibility of using the two parameter Chinese restaurant process (the urn representation of the Pitman-Yor process [Pitman & Yor, 1997]) in the construction of the DDT in Theorem 2. This in principle does not introduce any additional difficulty. One can imagine a nested two parameter CRP, using an analogous rate function $c(t)$ to give the discount parameter for each level. The problem is that it would still be necessary to avoid instantaneous fragmentation by driving the discount parameters to zero as $S \to \infty$, e.g. by setting the discount parameter at time t to $c(t)/S$. It is straightforward to see that this will again recover the standard DDT, although with rate function $a(t)+c(t)$: the probability of divergence will be $(a(t) + c(t))/(Sm)$ when there is one block, i.e. on a chain, so the logic of Theorem 2 follows; the probability of forming a third cluster at any branch point is $(a(t) + 2c(t))/(Sm)$ which tends to zero as $S \to \infty$; and finally the probability of following a branch k at a branch point is $\frac{b_k - c(t_s)/S}{m + a(t_s)/S}$ which again recovers the DDT factor b_k/m in the limit.

Thus the construction of the DDT in Theorem 2 destroys both the arbitrary branching structure of the underlying finite level nCRP and does not allow the extra flexibility provided by the two parameter CRP. This has ramifications beyond the construction itself: it implies that attempting to use a simple nCRP model in a very deep hierarchy has strong limitations. Either only the first few levels will be used, or the probability of higher order branching events must be made exponentially small. Note that this is not necessarily a problem for discrete data [Steinhardt & Ghahramani, 2012]. Additionally, the two parameter generalisation cannot be used to any advantage.

To obtain the multifurcating PYDT rather than the binary DDT we will modify the construction above.

Associate level s of an S-level nested partitioning model with time

$$t_s = (s-1)/S.$$

For a node at level s with only $K = 1$ cluster, let the probability of forming a new cluster be $\frac{a'(m,s)/S}{m+a'(m,s)/S}$ where

$$a'(m,s) = ma(t_s)\frac{\Gamma(m-\alpha)}{\Gamma(m+1+\theta)}, \tag{4.40}$$

where $0 \leq \alpha < 1, \theta > -2\alpha$ are hyperparameters. At an existing branch point (i.e. if the number of existing clusters is $K \geq 2$) then let the probabilities be given by the two parameter CRP, i.e. the probability of joining an existing cluster k is

$$\frac{b_k - \alpha}{m+\theta}, \tag{4.41}$$

where b_k is the number of samples in cluster k and m is the total number of samples through this branch point so far. The probability of diverging at the branch point and creating a new branch is

$$\frac{\theta + \alpha K}{m+\theta}, \tag{4.42}$$

where K is the current number of clusters from this branch point.

Theorem 3. *In the limit $S \to \infty$ the construction above becomes equivalent to the PYDT with rate function $a(t)$, concentration parameter θ and discount parameter α.*

Proof. Showing the correct distribution for the divergence times is analogous to the proof for Theorem 2. The probability of divergence from a chain at any level s behaves as $\frac{a'(m,s)}{Sm}$ as $S \to \infty$. The number of nodes k in a chain starting at level b until divergence is distributed:

$$\frac{a'(m,b+k)}{Sm}\prod_{i=1}^{k-1}\left(1 - \frac{a'(m,b+i)}{Sm}\right) = \frac{a(t_{b+k})\Gamma(m-\alpha)}{S\Gamma(m+1+\theta)}\prod_{i=1}^{k-1}\left(1 - \frac{a(t_{b+i})\Gamma(m-\alpha)}{S\Gamma(m+1+\theta)}\right). \tag{4.43}$$

Following the proof of Theorem 2 in the limit $S \to \infty$ this becomes

$$\frac{\Gamma(m-\alpha)}{S\Gamma(m+1+\theta)}a(t)\exp\left\{-\frac{\Gamma(m-\alpha)}{\Gamma(m+1+\theta)}\int_{t_b}^{t}a(\tau)d\tau\right\}. \tag{4.44}$$

Since Equations 4.11 and 4.41, and Equations 4.10 and 4.42 are the same, it is straightforward to see that the probabilities for higher order branching events are exactly as for the PYDT, i.e. given by Equation 4.23. □

Note that the finite level model of Theorem 3 is not exchangeable until we take the limit $S \to \infty$. Every node at level s with only $K = 1$ cluster contributes a factor

$$\prod_{i=1}^{m-1}\left(1-\frac{a'(i,s)/S}{j+a'(i,s)/S}\right), \tag{4.45}$$

where $a'(.)$ is defined in Equation 4.40 and m is the total number of samples having passed through this node. This factor does not depend on the order of the data points. Now consider a node with $K \geq 2$ clusters at level s. Assume the i-th sample diverged to create this branch point initially. The first $i-1$ samples did not diverge, the first contributing no factor, and the subsequent $i-2$ contributing a total factor

$$\prod_{j=2}^{i-1}\left(1-\frac{a'(j,s)/S}{m+a'(j,s)/S}\right). \tag{4.46}$$

Although this factor tends to 1 as $S \to \infty$, for finite S it depends on i. The probability of the i-th sample diverging to form the branch point is

$$\frac{a'(i,s)/S}{m+a'(i,s)/S} = \frac{a(t_s)}{S+a'(i,s)/i}\frac{\Gamma(i-\alpha)}{\Gamma(i+1+\theta)}. \tag{4.47}$$

The probability contributed by the samples after i is exactly the same as Equation 4.22 in Lemma 1, given by

$$\frac{\prod_{k=3}^{K_b}[\theta+(k-1)\alpha]\Gamma(i+\theta)\prod_{l=1}^{K_b}\Gamma(n_l^b-\alpha)}{\Gamma(m(b)+\theta)\Gamma(i-1+\alpha)}. \tag{4.48}$$

Multiplying this by Equation 4.47 we obtain

$$\frac{a(t_s)}{S + a'(i,s)/i} \frac{\prod_{k=3}^{K_b}[\theta + (k-1)\alpha] \prod_{l=1}^{K_b} \Gamma(n_l^b - \alpha)}{\Gamma(m(b) + \theta)}. \tag{4.49}$$

It is easy enough to see that we will recover the correct expression for the PYDT in the limit $S \to \infty$, using $1/S \to dt$. However, for finite S this factor, and the factor in Equation 4.46, depend on i, so we do not have exchangeability. It is interesting that the PYDT is therefore a case where the continuum limit has more attractive characteristics than the finite model.

4.5 Hierarchical clustering model

To use the PYDT as a hierarchical clustering model we must specify a likelihood function for the data given the leaf locations of the PYDT, and priors on the hyperparameters. We use a Gaussian observation model for multivariate continuous data and a probit model for binary vectors. We use the divergence function $a(t) = c/(1-t)$ and specify the following priors on the hyperparameters:

$$\theta \sim \mathrm{G}(a_\theta, b_\theta), \qquad \alpha \sim \mathrm{Beta}(a_\alpha, b_\alpha), \tag{4.50}$$

$$c \sim \mathrm{G}(a_c, b_c), \qquad 1/\sigma^2 \sim \mathrm{G}(a_{\sigma^2}, b_{\sigma^2}), \tag{4.51}$$

where $G(a, b)$ is a Gamma distribution with shape, a and rate, b. In all experiments we used $a_\theta = 2, b_\theta = .5, a_\alpha = 1, b_\alpha = 1, a_c = 1, b_c = 1, a_{\sigma^2} = 1, b_{\sigma^2} = 1$.

4.6 Inference

We propose two inference algorithms: an MCMC sampler and a more computationally efficient greedy EM algorithm. Both algorithms marginalize out the locations of internal nodes using belief propagation, and are capable of learning the hyperparameters c, σ^2, θ and α if desired.

4.6.1 MCMC sampler

We construct an MCMC sampler to explore the posterior over the tree structure, divergence times and hyperparameters. To sample the structure and divergence times our sampler uses moves that detach and reattach subtrees. A subtree is chosen uniformly at random to be detached (the subtree may be a single leaf node). To propose a new position in the tree for the detached subtree, we follow the procedure for generating a new sample on the remaining tree. The subtree is attached wherever divergence occurred, which may be on a segment, in which case a new parent node is created, or at an existing internal node, in which case the subtree becomes a child of that node. If divergence occurred at a time later than the divergence time of the root of the subtree we must repeat the procedure until this is not the case. The marginal likelihood of the new tree is calculated, marginalizing over the internal node locations, and excluding the structure and divergence time contribution since this is accounted for by having sampled the new location according to the prior. The ratio to the marginal likelihood for the original tree gives the Metropolis factor used to determine whether this move is accepted. Unfortunately it is not possible to slice sample the position of the subtree as in Neal [2003a] because of the atoms in the prior at each branch point.

Smoothness hyperparameter, c. From Equation 4.28 the conditional posterior for c is

$$\mathrm{G}\left(a_c + |\mathcal{I}|, b_c + \sum_{i \in \mathcal{I}} J_{n^i}^{\theta,\alpha} \log\left(1 - t_i\right)\right), \tag{4.52}$$

where $\mathcal{I}$ is the set of internal nodes of the tree.

Data variance, σ^2. It is straightforward to sample $1/\sigma^2$ given divergence locations. Having performed belief propagation it is easy to jointly sample the divergence locations using a pass of backwards sampling. From Equation 4.26

the Gibbs conditional for the precision $1/\sigma^2$ is then

$$\mathrm{G}(a_{\sigma^2}, b_{\sigma^2}) \prod_{[ab]\in\mathcal{S}(\mathcal{T})} \mathrm{G}\left(D/2+1, \frac{||x_a - x_b||^2}{2(t_b - t_a)}\right), \tag{4.53}$$

where $||\cdot||$ denotes Euclidean distance.

Pitman-Yor hyperparameters, θ and α. We use slice sampling [Neal, 2003b] to sample θ and α. We reparameterize in terms of the logarithm of θ and the logit of α to extend the domain to the whole real line. The terms required to calculate the conditional probability are those in Equations 4.23 and 4.25.

4.6.2 Greedy Bayesian EM algorithm

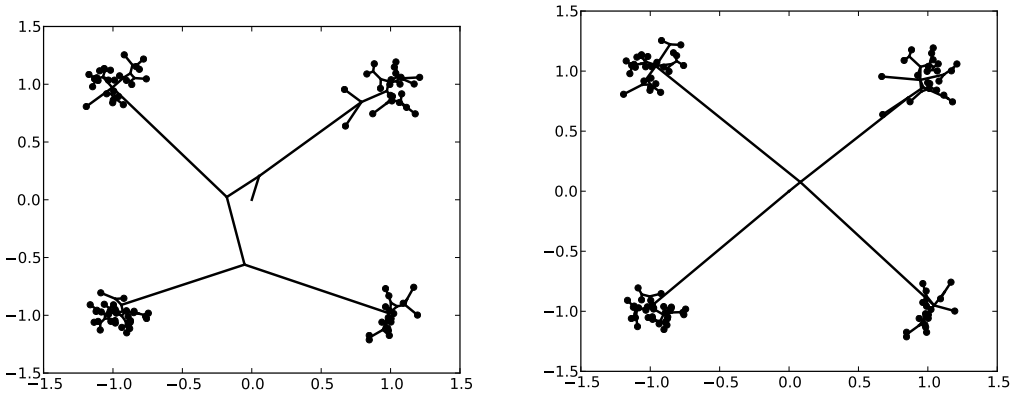
As an alternative to MCMC here we use a Bayesian EM algorithm to approximate the marginal likelihood for a given tree structure, which is then used to drive a greedy search over tree structures. This algorithm is presented in detail for both the DDT and PYDT in Chapter 7.

4.6.3 Likelihood models

Connecting our PYDT module to different likelihood models is straightforward: we use a Gaussian observation model and a probit model for binary vectors. The MCMC algorithm slice samples auxiliary variables and the EM algorithm uses EP [Minka, 2001b] on the probit factor, implemented using the runtime component of the Infer.NET framework Minka *et al.* [2010].

4.7 Results

We present results on synthetic and real world data, both continuous and binary.



(a) DDT (b) PYDT

Figure 4.9: Optimal trees learnt by the greedy EM algorithm for the DDT and PYDT on a synethic dataset with $D = 2, N = 100$.

4.7.1 Synthetic data

We first compare the PYDT to the DDT on a simple synthetic dataset with $D = 2, N = 100$, sampled from the density

$$f(x, y) = \frac{1}{4} \sum_{\bar{x} \in \{-1,1\}} \sum_{\bar{y} \in \{-1,1\}} N(x; \bar{x}, 1/8) N(y; \bar{y}, 1/8)$$

The optimal trees learnt by 100 iterations of the greedy EM algorithm are shown in Figure 4.9. While the DDT is forced to arbitrarily choose a binary branching structure over the four equi-distant clusters, the PYDT is able to represent the more parsimonious solution that the four clusters are equally dependent. Both models find the fine detail of the individual cluster samples which may be undesirable; investigating whether learning a noise model for the observations alleviates this problem is a subject of future work.

4.7.2 Density modeling

In Adams *et al.* [2008] the DDT was shown to be an excellent density model on a $D = 10, N = 228$ dataset of macaque skull measurements, outperforming a kernel

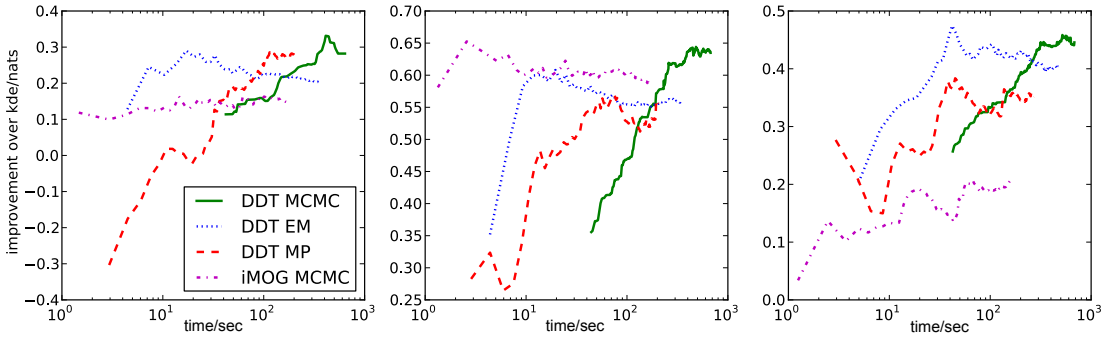


Figure 4.10: Density modelling of the $D = 10, N = 200$ macaque skull measurement dataset of Adams *et al.* [2008]. *Top*: Improvement in test predictive likelihood compared to a kernel density estimate. *Bottom*: Marginal likelihood of current tree. The shared x-axis is computation time in seconds.

density and Dirichlet process mixture of Gaussians, and sometimes the Gaussian process density sampler proposed in that paper. We compare the PYDT to the DDT on the same dataset, using the same data preprocessing and same three train test splits ($N_{\text{train}} = 200, N_{\text{test}} = 28$) as Adams *et al.* [2008]. The performance using the MCMC sampler is shown in Figure 4.10. The PYDT finds trees with higher marginal likelihood than the DDT, which corresponds to a moderate improvement in predictive performance. The posterior hyperparameters were reasonably consistent across the three train/test splits, with $\theta = 2.3 \pm 0.4$ and $\alpha = 0.23 + 0.08$ averaged over the last 100 samples for the first training split for example. Inference in the PYDT is actually slightly more efficient computationally than in the DDT because on average the smaller number of internal nodes reduces the cost of belief propagation over the divergence locations, which is the bottleneck of the algorithm (being a subroutine of the tree search procedure).

4.7.3 Binary example

To demonstrate the use of an alternative observation model we use a probit observation model in each dimension to model 102-dimensional binary feature vectors relating to attributes (e.g. being warm-blooded, having two legs) of 33 animal species from Tenenbaum & Kemp [2008]. The MAP tree structure learnt using EM, as shown in Figure 4.11, is intuitive, with subtrees corresponding to

land mammals, aquatic mammals, reptiles, birds, and insects (shown by colour coding). Note that penguins cluster with aquatic species rather than birds, which is not surprising since the data includes attributes such as "swims", "flies" and "lives in water".

4.8 Conclusion

We have introduced the Pitman-Yor Diffusion Tree, a Bayesian nonparametric prior over tree structures with arbitrary branching structure at each branch point. We have shown the PYDT defines an infinitely exchangeable distribution over data points. We demonstrated an MCMC sampler and Bayesian EM with greedy search, both using message passing on the tree structure. In ongoing work we are investigating more advanced MCMC methods. Quantitatively we have shown a modest improvement relative to the DDT on a density estimation task. However, we see improved interpretability as the key benefit of removing the restriction to binary trees, especially since hierarchical clustering is typically used as a data exploration tool. Qualitatively, we have shown the PYDT can find simpler, more interpretable representations of data than the DDT.

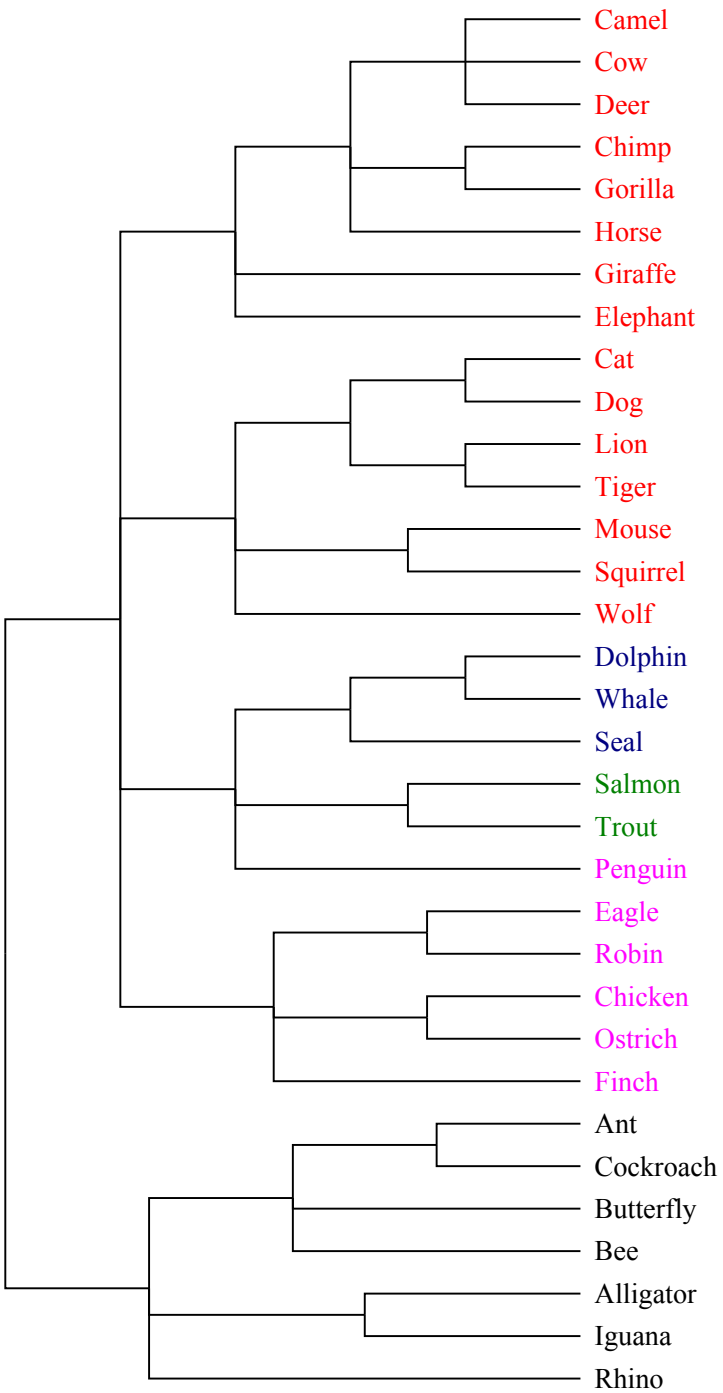


Figure 4.11: Tree structure learnt for the animals dataset of Tenenbaum & Kemp [2008]. Contrast this to Figure 7.6 which shows the solution under the DDT.

Chapter 5

Background: Message passing algorithms

In this chapter we give some background on message passing algorithms to set the scene for the novel algorithm introduced in Chapter 6. Hopefully this will also serve as a useful reference for practitioners wishing to use these methods. While the methods and results presented in this chapter have been available in the literature for some years, there has been little attempt to provide a coherent review.

5.1 Factor graphs

In fully Bayesian inference no distinction is made between latent variables and model parameters. Both are unknown quantities whose posterior distribution given data is of interest. Our aim is to approximate some distribution $p(\mathbf{x})$, represented as a factor graph $p(\mathbf{x}) = \frac{1}{Z} f(\mathbf{x}) = \frac{1}{Z} \prod_a f_a(\mathbf{x}_a)$ where factor f_a is a function of all $x \in \mathbf{x}_a$. Note that $p(\mathbf{x})$ might be a conditional distribution, i.e. $p(\mathbf{x}) = p(\mathbf{x}|\mathcal{D})$ where $\mathcal{D}$ is some observed data. The factor graph for a specific simple example:

$$f(x, y_1, y_2) = p_0(x) f_1(x, y_1) f_2(x, y_2) \tag{5.1}$$

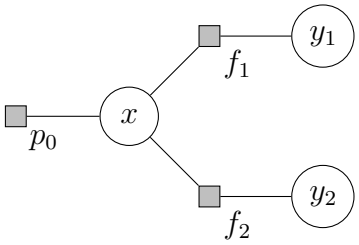


Figure 5.1: The factor graph for the simple model in Equation 5.1. Circles denote random variables and squares denote factors. Factors are connected by edges to variables on which they have a function dependence.

is shown in Figure 5.1. Circles denote random variables and squares denote factors. A factor f_a is connected by an edge to variable x_i if $x_i \in \mathbf{x}_a$, i.e. if f_a is a function of x_i. Although simple, this factor graph has the structure of many useful models, for example Gaussian Process regression [Rasmussen & Williams, 2006] would have $p_0(x) = N(x; \mathbf{0}, \mathbf{K})$ and, $f_i(x, y_i) = N(y_i; x_i, \sigma^2)$ where $x \in \mathbb{R}^N$ is the latent function evaluated at the N training points, $y_i \in \mathbb{R}$ are the observations, $\mathbf{K}$ is the Gram matrix and σ^2 is the noise variance.

5.2 Belief propagation/Sum-product algorithm

Given a factor graph representation of a model, $p(\mathbf{x})$ we typically want to compute either marginal distributions of some $x \in \mathbf{x}$ or calculate the partition function/marginal likelihood, $Z = \int f(\mathbf{x}) d\mathbf{x}$. For tree structured models with appropriate functional forms (e.g. purely discrete or Gaussian models), both these objectives can be achieved efficiently using the sum product algorithm, also known as belief propagation [Pearl & Shafer, 1988] (the sum-product algorithm can be viewed as somewhat more general than BP as it operates on the factor graph itself, but we will refer to them interchangeably). Belief propagation (BP) involves propagating "messages" (distributions which may or may not be normalised) across the factor graph. From variable x_i to factor f_a we send the message

$$m_{i \to a}(x_i) = \prod_{b \in \text{ne}(i) \neg a} m_{b \to i}(x_i) \tag{5.2}$$

where $\mathrm{ne}(i)$ denotes the neighbours of i, and $\neg a$ denotes exclusion of a. From factor f_a to variable x_i the message is

$$m_{a\to i}(x_i) = \int f_a(x_a) \prod_{j\in \mathrm{ne}(a)\neg i} [m_{j\to a}(x_j)dx_j] \tag{5.3}$$

For a tree structured factor graph one can think of $m_{a\to i}(x_i)$ as representing our belief about variable x_i conditional on the information in the subgraph attached to factor f_a. It is then intuitively reasonably that we obtain the marginal distribution for x_i by multiplying together all the independent belief contributions from each subgraph attached to x_i:

$$q_i(x_i) = \prod_{b\in \mathrm{ne}(i)} m_{b\to i}(x_i) \tag{5.4}$$

This explains why BP is only exact for tree structured graphs: on graphs with cycles the belief messages sent from each factor to x_i are not in general independent.

The BP equations can look quite intimidating but should not be: they follow from the basic rules of probability. Consider for example the factor graph in Figure 5.1. The message from p_0 to x is simply $m_{p_0\to x}(x) = p_0(x)$ itself, since p_0 depends on no other variables, i.e. $\mathrm{ne}(p_0)\neg a = \emptyset$. The message from x to f_1 is simply $m_{x\to f_1}(x) = m_{p_0\to x}(x)m_{f_1\to x}(x)$, and from f_1 to x is $m_{f_1\to x}(x) = \int f_1(x, y_1)m_{y_1\to f_1}(y_1)dy_1$. In the case of observed y_1 we can think of $m_{y_1\to f_1}(y_1)$ as being a point mass, $\delta(y_1 - y_1^{\mathrm{obs}})$, so the message to x will simplify:

$$\begin{aligned} m_{f_1\to x}(x) &= \int f_1(x, y_1)m_{y_1\to f_1}(y_1)dy_1 \\ &= f_1(x, y_1^{\mathrm{obs}}) \end{aligned} \tag{5.5}$$

We see that for typical factor graphs used in probabilistic modelling the BP messages are actually quite simple.

5.3 Exponential family distributions

It may seem confusing that the messages are functions of random variables: how can we represent the infinite degrees of freedom of a function algorithmically? In practice one chooses models such that the messages have convenient parametric forms, typically being distributions in the exponential family, for example Gaussian or discrete distributions, although there are exceptions (see for example Sudderth *et al.* [2003] where mixtures of Gaussians are used). An exponential family distribution is one that can be written as

$$f(x;\phi) = \exp(\phi^T \mathbf{u}(x) - \kappa(\phi)), \tag{5.6}$$

where $\mathbf{u}(\cdot)$ is a vector of sufficient statistics, ϕ is the vector of natural parameters and $\kappa(\cdot)$ is the log partition function. It is straightforward to derive the expected sufficient statistics (also sometimes referred to as the "moments" by analogy to the Gaussian case) as

$$\langle \mathbf{u}(x) \rangle_f = \kappa'(\phi), \tag{5.7}$$

where $\kappa'(\cdot)$ is the first derivative of $\kappa(\cdot)$. Exponential family distributions arise naturally as the maximum entropy distributions under constraints on the moments, and many common distributions used in statistical modelling, such as Gaussian ($u(x) = [x, -\frac{1}{2}x^2]^T$), Gamma ($u(x) = [x, \log x]^T$), and Beta ($u(x) = [\log x, \log(1-x)]^T$), are in the exponential family. Others however are not: common examples are the student-t and log-normal distributions. A key advantage of using exponential families is the property of closure under multiplication:

$$f(x;\phi_1)f(x;\phi_2) = \exp\left(\kappa(\phi_1+\phi_2) - \kappa(\phi_1) - \kappa(\phi_2)\right) f(x;\phi_1+\phi_2) \tag{5.8}$$

If the messages used in BP are all of the same exponential family form then calculating the variable to factor message in Equation 5.2 simply involves summing natural parameters.

5.4 Limitations of BP

Two main problems can arise which prevent the use of BP. Firstly, the graph may have cycles. In this case the exact junction tree algorithm [Jensen *et al.*, 1994] may still be applicable, but has run-time exponential in the *treewidth* of the graph. The treewidth is the number of largest number of graph nodes mapped into a single node in the junction tree. An alternative is loopy belief propagation [Frey & MacKay, 1998], where the standard BP equations are simply iterated in the hope of achieving a reasonable solution. It has been shown that the fixed points of loopy BP correspond to saddle points of the Bethe free energy, an approximation to the true Gibbs free energy which ignores terms resulting from cycles [Yedidia *et al.*, 2003]. Loopy BP is not guaranteed to converge, but has been applied with great success for certain problems such as decoding turbocodes [Berrou *et al.*, 1993]. Guaranteed convergent alternatives are available [Yuille, 2002] but are typically much slower.

Secondly, the messages sent to a variable may not permit straightforward multiplication, and even if they do, the integration required in Equation 5.3 may be challenging. If the messages are not of exponential family form, or are from different exponential families, then the variable to factor messages will be an inconvenient form and for certain models the complexity of the messages can grow exponentially in the number of iterations. This problem motivates somehow constraining all messages to a variable to have the same (exponential family) form. This is the idea behind Assumed Density Filtering [Boyen & Koller, 1998; Lauritzen, 1992; Maybeck, 1979] and Expectation Propagation [Minka, 2001c].

5.5 Expectation Propagation

Expectation Propagation (EP) can be motivated in a number of ways. Following our narrative so far, it can be thought of as a somehow "optimal" way to approximate a message of arbitrary functional form using a distribution from a parametric family of our choosing. To explain in what way this approximate message will be optimal we need to introduce the Kullback—Leibler (KL) divergence

between two distributions p and q:

$$KL(p||q) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}. \tag{5.9}$$

The KL divergence is non-negative and zero if and only if $p = q$. EP can be thought of as attempting to approximately minimise the global KL divergence $KL(p||q)$ where $p(\mathbf{x})$ is the true posterior and $q(\mathbf{x})$ is some approximation to $p(\mathbf{x})$[1]. Since our model is a product of factors $p(\mathbf{x}) = \frac{1}{Z} \prod_a f_a(\mathbf{x}_a)$ the approximation will be a product of approximations to these factors: $q(x) = \frac{1}{Z} \prod_a \tilde{f}_a(\mathbf{x}_a)$. Minimising the KL divergence can itself be motivated from a decision theoretic perspective: it is equivalent to finding the approximate posterior, q which minimises the Bayes' risk where the "action" is estimating the posterior density of $\mathbf{x}$ and the loss is the difference in log densities between the truth and the approximation. As discussed below in Section 5.12, EP can also be derived as fixed point scheme designed to find stationary points of the Bethe free energy under relaxed constraints relative to loopy BP [Heskes & Zoeter, 2002]. The global KL divergence $KL(p||q)$ can be written

$$KL\left(\prod_a f_a(\mathbf{x}_a) \middle|\middle| \prod_a \tilde{f}_a(\mathbf{x}_a)\right). \tag{5.10}$$

Minimising this functional with respect to all $\tilde{f}_a$ is intractable for most models of interest. Instead for each factor in turn EP aims to find a KL minimising approximation to $f_a(\mathbf{x}_a)$ under the assumption that the approximation to the other factors is good, i.e. that $\prod_{b \neq a} \tilde{f}_b(\mathbf{x}_b) \approx \prod_{b \neq a} f_b(\mathbf{x}_b)$. This gives the following update:

$$\tilde{f}_a(\mathbf{x}_a) = \arg\min_{s \in \mathcal{E}_a} \mathrm{KL}\left(f_a(\mathbf{x}_a) \prod_{b \neq a} \tilde{f}_b(\mathbf{x}_b) \middle|\middle| s(\mathbf{x}_a) \prod_{b \neq a} \tilde{f}_b(\mathbf{x}_b)\right), \tag{5.11}$$

where $\mathcal{E}_a$ denotes the space of exponential family distributions we have chosen for factor a. We now wish to translate the update in Equation 5.11 into message

[1] It is a common misconception that EP actually minimises the global KL: this is only a motivation for the algorithm, EP does not in general achieve this goal.

passing formalism. We will assume a fully factorised posterior approximation $q(\mathbf{x}) \propto \prod_i q_i(x_i)$ so that each approximate factor will also factorise:

$$\tilde{f}_a(\mathbf{x}_a) = \prod_{i \in \text{ne}(a)} m_{a \to i}(x_i). \tag{5.12}$$

The variable to factor messages are calculated exactly as for BP (see Equation 5.2):

$$m_{i \to a}(x_i) = \prod_{b \neq a} m_{b \to i}(x_i). \tag{5.13}$$

We can now write the product of approximate factors as

$$\prod_{b \neq a} \tilde{f}_b(\mathbf{x}_b) = \prod_{b \neq a} \prod_{i \in \text{ne}(a)} m_{a \to i}(x_i) = \prod_{i \in \text{ne}(a)} m_{i \to a}(x_i) \tag{5.14}$$

Substituting into Equation 5.11 we have

$$\prod_{i \in \text{ne}(a)} m_{a \to i}(x_i) = \arg\min_{s \in \mathcal{E}_a} \text{KL}\left(f_a(\mathbf{x}_a) \prod_{i \in \text{ne}(a)} m_{i \to a}(x_i) \middle\| \prod_{i \in \text{ne}(a)} s_i(x_i) m_{i \to a}(x_i) \right) \tag{5.15}$$

where $s(\mathbf{x}_a)$ is now constrained to be fully factorised, i.e. $s(\mathbf{x}_a) = \prod_{i \in \text{ne}(a)} s_i(x_i)$. This optimisation is in fact independent for each message $m_{a \to i}(x_i)$ so we have

$$m_{a \to i}(x_i) = \arg\min_{s_i \in \mathcal{E}_i} \text{KL}\left(\int f_a(\mathbf{x}_a) \prod_{j \in \text{ne}(a)} m_{j \to a}(x_j) dx_j \middle\| s_i(x_i) m_{i \to a}(x_i) \right). \tag{5.16}$$

where $\mathcal{E}_i$ is the exponential family chosen for variable i. Finally it should be clear that we can write Equation 5.16 as

$$\begin{aligned} m_{a\to i}(x_i) &= \frac{1}{m_{i\to a}(x_i)} \arg\min_{s_i\in\mathcal{E}_i} \mathrm{KL}\left(f_a(\mathbf{x}_a) \prod_{j\in \mathrm{ne}(a)} m_{j\to a}(x_j) \middle\| s_i(x_i) \right) \\ &= \frac{1}{m_{i\to a}(x_i)} \mathrm{proj}\left[\int f_a(\mathbf{x}_a) \prod_{j\in \mathrm{ne}(a)} m_{j\to a}(x_j) d\mathbf{x}_{a\neg i} \right], \end{aligned} \tag{5.17}$$

where we have introduced the *proj* operator which projects a distribution onto the closest exponential family distribution in terms of KL divergence. It is straightforward to show that this is found by moment matching, i.e. if $s(x) = \exp\{\mathbf{u}(x)^T\phi - \kappa(\phi)\}$ then choose ϕ such that

$$\langle u_l(x)\rangle_s = \langle u_l(x) f_a(\mathbf{x}_a)\rangle_{\prod_{j\in \mathrm{ne}(a)} m_{j\to a}(x_j)}, \tag{5.18}$$

for each l. Equation 5.17 reduces to the standard BP message in the case where the BP message would have been in $\mathcal{E}_i$, the chosen exponential family for x_i.

5.5.1 A note on the convergence of EP

We know that BP is guaranteed to converge on tree structured graphical models. Expectation propagation is a generalisation of BP, for which no convergence guarantees are known. Intuitively, if the graph is tree structured, and the factors requiring approximation are in some sense "close" to conjugate, then there should be a good chance that EP will converge. Here we formalise in what sense the factors must be close to conjugate to at least give a guarantee of local convergence, i.e. that given an initialisation sufficiently close to the optimum, the algorithm will converge.

We will consider the very simple factor graph model

$$f(x) = f_1(x) f_2(x), \tag{5.19}$$

but note that it is possible to extend this idea to more general models. We will work in the exponential family $q(x|\theta) = \exp\left(\theta^T u(x) - \kappa(\theta)\right)$, where $u(x) \in \mathbb{R}^d$

is the vector of sufficient statistics. Let the messages from f_1 and f_2 to x have natural parameters m_1 and m_2 respectively. The posterior has natural parameter $\theta = m_1 + m_2$. Now consider the EP update for m_1. Since dividing two exponential family distributions corresponds to subtracting their natural parameters, from Equation 5.17 the new value of m_1 is given by

$$m_1' = \text{proj}\left[f_1(x)e^{m_2^T u(x) - \kappa(m_2)}\right] - m_2, \tag{5.20}$$

where by a slight abuse of notation we now consider the output of the *proj* operator to be the natural parameters of the distribution. Let

$$g(\phi) = \int u(x)q(x|\phi)dx = \langle u(x)\rangle_\phi = \kappa'(\phi), \tag{5.21}$$

be the bijective function mapping natural parameters to sufficient statistics. The projection operator can be written as follows:

$$\text{proj}\left[F(x)\right] = g^{-1}\left(\frac{\int u(x)F(x)dx}{\int F(x)dx}\right), \tag{5.22}$$

for some arbitrary function, $F(.)$. The update for m_1 can be written

$$m_1' = g^{-1}\left(\int u(x)\tilde{F}_1(x, m_2)dx\right) - m_2 =: G_1(m_2) \tag{5.23}$$

where $G_1(m_2)$ is the function mapping m_2 to the new value of m_1 and we have defined

$$\begin{aligned} F_1(x, m_2) &:= f_1(x)q(x|m_2) \\ &= f_1(x)e^{m_2^T u(x) - \kappa(m_2)} \end{aligned} \tag{5.24}$$

$$\tilde{F}_1(x, m_2) := \frac{F_1(x, m_2)}{\int F_1(x, m_2)dx}. \tag{5.25}$$

Analogously we can write down the update for m_2:

$$m_2' = G_2(m_1') := g^{-1}\left(\frac{\int u(x)F_2(x, m_1')dx}{\int F_2(x, m_1)dx}\right) - m_1', \tag{5.26}$$

where $F_2(x, m_1) = f_2(x)q(x|m_1)$. A full EP iteration can be thought of as the following mapping of m_2:

$$m_2' = G_2(G_1(m_2)) =: G(m_2). \tag{5.27}$$

To check if $G(.) : \mathbb{R}^d \to \mathbb{R}^d$ will converge at least locally we can look at the Taylor expansion around a fixed point, m^*, where of course $m^* = G(m^*)$. Let m^t be the value of m_2 at iteration t and $\epsilon^t := m^t - m^*$ be the error. Then

$$\begin{aligned} m^{t+1} := G(m^t) &= G(m^*) + G'(m^*)\epsilon^t + O(|\epsilon^t|^2) \\ &\approx m^* + G'(m^*)\epsilon^t \\ \Rightarrow \epsilon^{t+1} = m^{t+1} - m^* &= G'(m^*)\epsilon^t + O(|\epsilon^t|^2) \end{aligned} \tag{5.28}$$

Clearly if the determinant of the Jacbobian, $|G'(.)| < 1$ then $\lim_{t\to\infty} \epsilon^t = \mathbf{0}$, so we will consider when this is the case. Note that convergence in some other norm could also be used [Wang & Titterington, 2006]. We have

$$G'(m_2) = G_1'(m_2)G_2'(G_1(m_2)) \Rightarrow |G'(m_2)| = |G_1'(m_2)||G_2'(G_1(m_2))| \tag{5.29}$$

so we can attempt to show that $|G_1'(.)| < 1$ and $|G_2'(.)| < 1$. Let

$$h(m_2) = \int u(x)\tilde{F}_1(x, m_2)dx = \langle u(x) \rangle_{\tilde{F}_1}, \tag{5.30}$$

so that we can write

$$\begin{aligned} G_1(m_2) &= g^{-1}\left(h(m_2)\right) - m_2 \\ \Rightarrow G_1'(m_2) &= g'(G_1(m_2) + m_2)^{-1}h'(m_2) - I, \end{aligned} \tag{5.31}$$

where we have used that $\frac{df^{-1}(x)}{dx} = f'(f^{-1}(y))^{-1}$. The updated natural parameters for the posterior are $\theta' := G_1(m_2) + m_2 = m_1' + m_2$, so we have $G_1'(m_2) = g'(\theta')^{-1}h'(m_2) - I$. Using the fact that

$$\frac{\partial F_1}{\partial m_2} = (u(x) - \kappa'(m_2))F_1(x, m_2), \tag{5.32}$$

it is straightforward to show that

$$h'(m_2) = \int u(x)u(x)^T \tilde{F}_1(x, m_2)dx - h(m_2)h(m_2)^T, \tag{5.33}$$

which is readily interpreted as the covariance of $u(x)$ under $\tilde{F}$. Now

$$\begin{aligned} g'(\theta') &= \kappa''(\theta') = \text{cov}[u(x)|\theta'] \\ &= \int u(x)u(x)^T e^{\theta'^T u(x) - \kappa(\theta')}dx - \langle u(x)|\theta'\rangle\langle u(x)^T|\theta'\rangle \end{aligned} \tag{5.34}$$

where

$$\langle u(x)|\theta'\rangle = \int u(x)e^{\theta'^T u(x) - \kappa(\theta')}dx \tag{5.35}$$

This is equal to $h(m_2)$ for $\theta' = m_1' + m_2$ since this was the criteria for choosing m_1'. We now have

$$G_1'(m_2) = \text{cov}[u(x)|\theta']^{-1}\text{cov}[u(x)|\tilde{F}_1] - I \tag{5.36}$$

We are interested in the determinant of this expression, which can be written

$$|G_1'(m_2)| = \frac{\left|\text{cov}(u(x)|\tilde{F}_1) - \text{cov}(u(x)|\theta')\right|}{|\text{cov}(u(x)|\theta')|} \tag{5.37}$$

by a simple manipulation. This quantity can be thought of as a measure of how well $\tilde{F}_1(., m_2)$ is approximated by the moment matched exponential family $q(x;\theta)$. If f_1 is conjugate then this measure will be zero, otherwise it will be positive. For a Gaussian approximation, this quantity will be a function of the skew and kurtosis of $\tilde{F}_1$. We are interested in whether this quantity is bounded above by 1, in other words does it hold that

$$\left|\text{cov}[u(x)|\tilde{F}_1] - \text{cov}[u(x)|\theta']\right| \leq |\text{cov}[u(x)|\theta']| \tag{5.38}$$

Consider a univariate posterior which we will attempt to approximate using a

Gaussian. Define the skew, s as

$$s = \frac{\langle (x-m)^3 \rangle}{v^{\frac{3}{2}}} \tag{5.39}$$

and the kurtosis, k as

$$k = \frac{\langle (x-m)^4 \rangle}{v^2} - 3, \tag{5.40}$$

where the mean, $m = \langle x \rangle$ and the variance, $v = \langle x^2 \rangle - m^2$. Note that $k = s = 0$ for a Gaussian distribution. For the Gaussian distribution, $u(x) = [x, \quad -\frac{1}{2}x^2]^T$ it is straightforward (if tedious) to show that

$$\mathrm{cov}[u(x)|\tilde{F}_1] = \begin{bmatrix} v & -\frac{1}{2}(v^{\frac{3}{2}}s + 2mv) \\ . & \frac{1}{4}(kv^2 + 2v^2 + 4v^{\frac{3}{2}}sm + 4m^2v) \end{bmatrix} \tag{5.41}$$

where . simply denotes that this element is equal to the other cross term by symmetry of the matrix. Clearly setting $k = s = 0$ we have

$$\mathrm{cov}[u(x)|\theta'] = \begin{bmatrix} v & -mv \\ . & \frac{1}{2}v^2 + m^2v \end{bmatrix} \tag{5.42}$$

So

$$\mathrm{cov}(u(x)|\tilde{F}_1) - \mathrm{cov}[u(x)|\theta'] = \begin{bmatrix} 0 & -\frac{1}{2}v^{\frac{3}{2}}s \\ . & \frac{1}{4}kv^2 + v^{\frac{3}{2}}sm \end{bmatrix} \tag{5.43}$$

Finally

$$\left| \mathrm{cov}[u(x)|\tilde{F}_1] - \mathrm{cov}[u(x)|\theta'] \right| = -\frac{1}{4}v^3s^2 \tag{5.44}$$

and

$$|\mathrm{cov}[u(x)|\theta']| = \frac{1}{2}v^3 \tag{5.45}$$

so

$$|G_1'(m_2)| = -\frac{1}{2}s^2 \tag{5.46}$$

Thus our requirement that $|G_1'(m_2)| < 1$ becomes a simple condition on the skew $|s| < \sqrt{2}$ of $\tilde{F}_1$, the normalised product of the true factor f_1 and the other approximate factors. Unfortunately in practice checking this condition is difficult, but it does give quantitative foundation to the qualitative statement that EP tends to converge better for factors it can approximate well. Note that if we have $|s| > \sqrt{2}$ we cannot say that the algorithm will diverge, only that we cannot guarantee convergence.

5.6 α-divergences/Fractional BP/Power EP

It is possible to generalise EP to α-divergence message passing [Minka, 2005], where instead of locally minimising the KL divergence we minimise the α-divergence:

$$D_\alpha(p||q) = \frac{1}{\alpha(1-\alpha)} \int [1 - p(\mathbf{x})^\alpha q(\mathbf{x})^{(1-\alpha)}] d\mathbf{x}. \tag{5.47}$$

The updates are then as follows, analogously to EP:

$$\tilde{f}_a(\mathbf{x}_a) := \arg\min_s D_\alpha(f_a(\mathbf{x}_a) m_{x\to a}(\mathbf{x}_a) || s(\mathbf{x}_a) m_{x\to a}(\mathbf{x}_a)). \tag{5.48}$$

We will not discuss general α-divergence message passing in detail here, but instead mention it as a bridge from EP to variational Bayes. Taking the limit $\alpha \to 0$ gives the KL divergence used in EP: $KL(p||q)$. Taking the limit $\alpha \to 1$ gives the KL divergence in the other direction, $KL(q||p)$, which is the divergence measure minimised by Variational Bayes and Variational Message Passing (see below), and is the only setting of α for which iteratively minimising the local divergence exactly minimises the global divergence. Related methods are Power EP [Minka, 2004], an alternative algorithm to minimise α-divergence with slightly different messages but the same fixed points as Equation 5.48, and Fractional BP [Wiegerinck & Heskes, 2002], the equivalent extension of standard BP. These

ideas motivate hybrid EP/VB algorithms, which have been successfully applied in various models [Ding *et al.*, 2010; Guiver & Snelson, 2009; Stern *et al.*, 2009] and an example of which is demonstrated in Chapter 7.

5.7 Variational Bayes

As mentioned above, α-divergence message passing with $\alpha = 0$ corresponds to the variational Bayes (VB) approximation [Attias, 2000; Beal & Ghahramani, 2006]. For completeness, and because it is normally presented using somewhat different notation, we review the standard derivation of VB here. Recall that the normalised distribution of interest is $p(\mathbf{x}) = \frac{f(\mathbf{x})}{Z}$. We use Jensen's inequality to lower bound the marginal likelihood:

$$\begin{aligned} \log Z &= \log \int_{\mathbf{x}} f(\mathbf{x}) d\mathbf{x} = \log \int_{\mathbf{x}} q(\mathbf{x}) \frac{f(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} \\ &\geq \int_{\mathbf{x}} q(\mathbf{x}) \log \frac{f(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} =: \mathcal{F}[q]. \end{aligned} \tag{5.49}$$

We can ask what error we are making between the true Z and the lower bound (known as the free energy) $\mathcal{F}[q]$:

$$\log Z - \mathcal{F}[q] = \int_{\mathbf{x}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} d\mathbf{x} = KL(q||p). \tag{5.50}$$

In general we can evaluate $\mathcal{F}[q]$ but not the KL itself, since this would require knowing Z. By maximising the lower bound $\mathcal{F}[q]$ we will minimise the KL divergence

$$KL(q||p) = \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} d\mathbf{x} = -H[q(\mathbf{x})] - \int q(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x}, \tag{5.51}$$

where $H[q(\mathbf{x})] = -\int q(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{x}$ is the entropy. Recall that the KL divergence is strictly positive for $q \neq p$ and equal to 0 only for $q = p$. As a result finding the general q which minimises the KL divergence is no easier than the original inference task, which we assume is intractable. The usual strategy therefore is to place simplifying constraints on q, the most popular, due to its simplicity, being

the mean field approximation.

5.8 Mean-field approximation

The mean-field approximation assumes a fully-factorised variational posterior

$$q(\mathbf{x}) = \prod_i q_i(x_i), \tag{5.52}$$

where $q_i(x_i)$ is an approximation to the marginal distribution of x_i (note however x_i might in fact be vector valued, e.g. with multivariate normal q_i). The variational approximation $q(\mathbf{x})$ is chosen to minimise the Kullback-Leibler divergence $KL(q||p)$ (or equivalently maximise $\mathcal{F}[q]$). It can be shown [Attias, 2000] that if the functions $q_i(x_i)$ are *unconstrained* then minimising this functional can be achieved by coordinate descent, setting

$$\begin{aligned} q_i(x_i) &:= \exp \int \log p(\mathbf{x}) \prod_{j \neq i} q_j(x_j) dx_j \\ &= \exp \langle \log p(\mathbf{x}) \rangle_{\neg q_i(x_i)}, \end{aligned} \tag{5.53}$$

iteratively for each i, where $\langle ... \rangle_{\neg q_i(x_i)}$ implies marginalisation of all variables except x_i.

5.9 Variational Message Passing on factor graphs

Variational Message Passing [Winn & Bishop, 2006] is an efficient algorithmic implementation of the mean-field approximation which leverages the fact that the mean-field updates only require local operations on the factor graph. VMP was originally presented as operating on the directed acyclic graph of a Bayesian network, but here we describe VMP operating directly on the factor graph to emphasise the similarity with BP and EP. As a result of the mean field approximation, the variational distribution $q(\mathbf{x})$ factorises into approximate factors $\tilde{f}_a(\mathbf{x}_a)$. As a result of the fully factorised approximation, the approximate factors

themselves factorise into messages, i.e.

$$\tilde{f}_a(\mathbf{x}_a) = \prod_{x_i \in \mathbf{x}_a} m_{a \to i}(x_i), \tag{5.54}$$

where the message from factor a to variable i is

$$\begin{aligned} m_{a \to i}(x_i) &= \exp\langle \log f_a(\mathbf{x}_a) \rangle_{\neg q_i(x_i)} \\ &= \exp \int \log f_a(\mathbf{x}_a) \prod_{j \in \mathrm{ne}(a) \neg i} q_j(x_j) dx_j. \end{aligned} \tag{5.55}$$

It is instructive to note the similarity of this message to that for BP:

$$m^{BP}_{a \to i}(x_i) = \int f_a(\mathbf{x}_a) \prod_{j \in \mathrm{ne}(a) \neg i} m_{j \to a}(x_j) dx_j. \tag{5.56}$$

There are two differences: firstly VMP involves integrating with respect to the *logarithm* of the factor and then exponentiating the resulting function, and secondly the messages from variables to factors contain slightly different information. In VMP the message from variable i to factor a is the current variational posterior of x_i, denoted $q_i(x_i)$, i.e.

$$m_{i \to a}(x_i) = q_i(x_i) = \prod_{a \in \mathrm{ne}(i)} m_{a \to i}(x_i). \tag{5.57}$$

Contrast this to the variable to factor message in BP which excludes the contribution from this factor, i.e. $m_{i \to a}(x_i) = q_i(x_i)/m_{a \to i}(x_i)$. This means that the effective state (and therefore memory requirement) of EP/BP is "larger" than for VMP since the later only requires all current marginals q rather than all the messages.

5.10 Conditional conjugacy

Under what conditions will the message in Equation 5.55 be an exponential family distribution? In other words when do we have:

$$\begin{aligned} m_{a\to i}(x_i) &= \exp \int \log f_a(\mathbf{x}_a) \prod_{j\in \mathrm{ne}(a)\neg i} q_j(x_j) dx_j \\ &= \exp\left(\mathbf{u}(x_i)^T \phi - \text{const.}\right). \end{aligned} \tag{5.58}$$

If this is the case then it must be possible to write

$$\log f_a(\mathbf{x}_a) = \mathbf{u}(x_i)^T g(\mathbf{x}_a^{\neg i}) + z(\mathbf{x}_a^{\neg i}) \tag{5.59}$$

where $\mathbf{x}_a^{\neg i}$ denotes the variables in $\mathbf{x}_a$ excluding x_i. In other words the log factor must be a linear combination of the sufficient statistics of the appropriate exponential family distribution. If all the factors connected to variable x_i have this form with respect to x_i then the update for x_i will be straightforward. If this is the case for all variables then we consider this model to have "conditional conjugacy", which is considerably less restrictive than the standard notion of conjugacy. Conditional conjugacy says that conditional on a variable's Markov blanket (i.e. variables with which it shares factors) its distribution is of a tractable exponential family form. This requires the log factors to be linear functions of the sufficient statistics of the exponential family distributions used in the variational posterior.

5.11 Deterministic factors in VMP

Deterministic factors require careful handling in VMP. Consider a factor $f_a(x, y) = \delta(y - f(x))$ where the current variational posteriors on x and y are $q(x)$ and $q(y)$ respectively. Naively applying the VMP update rules and taking care with limits

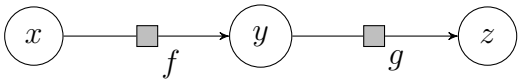


Figure 5.2: The factor graph for a simple model including a deterministic factor.

the message to y will be:

$$\begin{aligned} m_{f\to y}(y) &= \exp \int \log \delta(y - f(x))q(x)dx \\ &= \lim_{v\to 0} \exp \int \log N(y; f(x), v)q(x)dx \\ &= \lim_{v\to 0} N(y; \langle f(x)\rangle, v) \\ &= \delta(y - \langle f(x)\rangle) \end{aligned} \tag{5.60}$$

where we have represented the delta function as $\delta(x) = \lim_{v\to 0} N(x; 0, v)$ and used the continuity of exp and log to exchange the order of operations. Thus the message to y is a point mass, and we have lost all uncertainty, which is clearly underisable when our aim was to perform Bayesian inference. The same phenomenon occurs for the message to x:

$$\begin{aligned} m_{f\to x}(x) &= \exp \int \log \delta(y - f(x))q(y)dy \\ &= \lim_{v\to 0} \exp \int \log N(f(x); y, v)q(y)dy \\ &= \lim_{v\to 0} N(f(x); \langle y\rangle, v) \\ &= \delta(f(x) - \langle y\rangle) \qquad (5.61) \\ &= \delta(x - f^{-1}(\langle y\rangle)) \qquad (5.62) \end{aligned}$$

As a result deterministic factors are best handled in a slightly non-standard way. Consider the following model: $p(x, y, z) = \delta(y - f(x))g(y, z)$ (as shown in Figure 5.2) with current variational posteriors on x and z being $q(x)$ and $q(z)$ respectively.

The messages we would like to send to x and z are given by marginalising away y to give the model: $g(f(x), z)$. We assume that the "conditional conjugacy"

property holds, which implies that we can write the log factor as

$$\log g(y,z) = (\mathbf{a} + \mathbf{A}\mathbf{u}_y(y))^T(\mathbf{b} + \mathbf{B}\mathbf{u}_z(z)), \tag{5.63}$$

where $\mathbf{a}, \mathbf{b}$ are vectors and $\mathbf{A}, \mathbf{B}$ are matrices. This factor is now stochastic so the standard messages have no pathological behaviour:

$$\begin{aligned} m_{f\to x}(x) &\propto \exp \int \log g(f(x),z) q(z) dz \qquad (5.64)\\ &\propto \exp\left\{\mathbf{u}_y(f(x))^T \mathbf{A}^T \left(\mathbf{b} + \mathbf{B}\langle \mathbf{u}_z(z)\rangle\right)\right\} \\ m_{f\to z}(z) &\propto \exp \int \log g(f(x),z) q(x) dx \\ &\propto \exp\left\{(\mathbf{a} + \mathbf{A}\langle \mathbf{u}_y(f(x))\rangle_{q(x)})^T \mathbf{B}\mathbf{u}_z(z)\right\} \qquad (5.65) \end{aligned}$$

Can we handle the deterministic factor $\delta(y - f(x))$ in a way that will implicitly result in the same messages being sent to x and z as marginalising out y? Instead of representing the approximate posterior on y as a true marginal it will be a "pseudo-marginal". The message from f to y simply must have the moments of $f(x)$. Write the exponential family distribution for y as

$$q(y;\phi) = \exp\left(\mathbf{u}_y(y)^T\phi - \kappa(\phi)\right) \tag{5.66}$$

where $\mathbf{u}_y(y)$ and ϕ are the sufficient statistics and natural parameters of the exponential family respectively (note that requiring an exponential family distribution for y is an extra constraint that not using the marginalised representation introduces). We find ϕ by requiring that

$$\langle \mathbf{u}_y(y)\rangle_{q(y;\phi)} = \langle \mathbf{u}_y(f(x))\rangle_{q(x)} \tag{5.67}$$

This message is then used in calculating the message to z, resulting in the same

message as for the marginalised representation:

$$\begin{aligned} m_{f\to z}(z) &\propto \exp \int q(y;\phi) \log g(y,z) dy \\ &\propto \exp \left\{ (\mathbf{a} + \mathbf{A}\langle \mathbf{u}_y(y) \rangle_{q(y;\phi)})^T \mathbf{B} \mathbf{u}_z(z) \right\} \\ &\propto \exp \left\{ (\mathbf{a} + \mathbf{A}\langle \mathbf{u}_y(f(x)) \rangle_{q(x)})^T \mathbf{B} \mathbf{u}_z(z) \right\} \end{aligned} \tag{5.68}$$

which is the same as Equation 5.65. The message from g to y is the standard VMP message:

$$m_{g\to y}(y) = \exp \int \log g(y,z) q(z) dz \tag{5.69}$$

However, the message from y to f is non-standard because it excludes the contribution of the message $m_{f\to y}$. The message from f to x is then calculated as

$$\begin{aligned} m_{f\to x}(f(x)) = \hat{f}(x) &:= \int \delta(y - f(x)) m_{g\to y}(y) dy \\ &= \exp \int \log g(f(x), z) q(z) dz \end{aligned} \tag{5.70}$$

which is the same as the message for the marginalised representation (Equation 5.64). It is instructive to consider some simple examples.

A *very* simple example. Consider the model $p(x,y) = \delta(y - 2x) N(y; 1, 1)$. Clearly it is trivial to marginalise out y to find that the true posterior for x is $N(x; 1/2, 1/4)$. However, we would like to be able to calculate this answer using VMP, without having to collapse the model manually. The message from the normal factor to y is simply $N(y; 1, 1)$ itself. The message from y to x is calculated as $m_{N\to y}(f(x)) = N(2x; 1, 1)$ which directly gives the correct posterior.

The product factor. A slightly more interesting case, which is of practical relevance, is the product factor, $\delta(y - ab)$ where we assume Gaussian variational

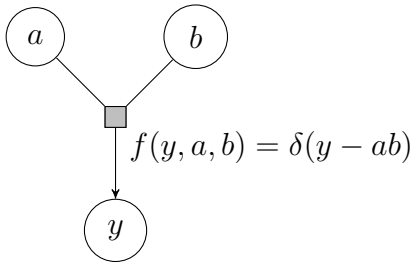


Figure 5.3: The product factor.

posteriors on a and b (Figure 5.3). The message from f to a is:

$$\begin{aligned} m_{f\to a}(a) &= \int m_{y\to f}(ab) q_b(b) db \\ &= \int N(ab, m_y, v_y) N(b, m_b, v_b) db \\ &\propto N\left(a; \frac{m_b m_y}{\langle b^2 \rangle}, \frac{v_y}{\langle b^2 \rangle}\right) \end{aligned} \tag{5.71}$$

where $m_{y\to f}(y) = \prod_{b\in \mathrm{ne}(y)\neg f} m_{f_b\to y}(y)$ and $\langle b^2 \rangle = m_b^2 + v_b$. The message to b should be clear from the symmetry of a and b. Using Equation 5.67 the message to y from f is

$$m_{f\to y}(y) = N(y; m_a m_b, \langle a^2 \rangle \langle b^2 \rangle - m_a^2 m_b^2) \tag{5.72}$$

By specific handling of the deterministic factor f and pseudomarginal y we are able to achieve the same messages as the marginalised representation. This lets us keep uncertainty in our estimates which is fundamental to Bayesian inference, whilst allowing us to conveniently connect a variety of deterministic and stochastic factors in interesting and useful combinations. Importantly the variational Bayes guarantees still hold: we still have a lower bound on the marginal likelihood which increases (or at least does not decrease) every iteration.

5.12 Further topics

Exclusive vs. inclusive KL. The different characteristics of EP and VMP can be explained in terms of the particular KL divergence each algorithm at-

tempts to minimise. EP minimises $KL(p||q) = \int p(x) \log p(x)/q(x) dx$, which is known as an "inclusive" divergence since the approximate q will attempt to include most of the mass of the true distribution p. This can be seen from the KL divergence: q will be severely penalised anywhere $p(x)$ is large and $q(x)$ is small. In contrast, VB minimises $KL(q||p) = \int q(x) \log q(x)/p(x) dx$, an "exclusive" divergence which most heavily penalises regions where $p(x)$ is small and $q(x)$ is not. This behaviour is most pronounced with multimodal $p(x)$, where EP will tend to spread the mass of $q(x)$ over multiple modes, whereas VB will focus on fitting a single mode (hopefully the largest, although each mode will typically represent a different local optimum). As a result VMP can often be more robust, since EP's attempt to cover multiple modes can result in the variance of the approximation diverging [Stern *et al.*, 2009]. VMP is often found to severely underestimate variances (although there are counterexamples, see for example Turner *et al.* [2008]). Thus the choice of algorithm must typically be made depending on the model, or hybrid approaches can be taken [Ding *et al.*, 2010; Guiver & Snelson, 2009; Qi *et al.*, 2005; Stern *et al.*, 2009].

Damping. Although it has been proposed that EP always converges for log concave distributions $p(x)$, there is no formal proof even in this case, and no guarantees at all in general. For complex models, especially multimodal ones, EP will often fail to converge, either entering a limit cycle or diverging. This behaviour can sometimes be alleviated by damping, which increases the basin of attraction of the stable solutions. Instead of simply using the message $m^{\text{new}}_{a \to i}(x_i)$ we use a convex combination with the previous message, i.e. $m^{\text{new}}_{a \to i}(x_i)^{(1-\delta)} m^{\text{old}}_{a \to i}(x_i)^{\delta}$ where $\delta \in [0, 1]$ is the damping factor. The fixed points of the algorithm are unchanged, but there is typically some cost in terms of convergence rate.

Scheduling. For approximate message passing algorithms the order of message updates, known as the schedule, is very important for three reasons. Firstly, algorithms which are not guaranteed to converge are more likely to converge with a better schedule. Secondly, the rate of convergence is greatly effected by the schedule. Finally in situations where there are multiple local optima/fixed points the schedule can effect which optimum is reached. The first two issues are in

fact tightly coupled: a schedule which converges faster is typically more robust as well. While various methods for choosing schedules for specific models have been proposed [Elidan *et al.*, 2006; Sutton & McCallum, 2007; Vila Casado *et al.*, 2007], finding good global schedules in the general case remains an open problem.

EP energy function. Although EP was originally motivated from an algorithmic standpoint, it corresponds to finding fixed points of the Bethe free energy under the weak constraint of having consistent moments (sufficient statistics) rather than the strong constraint of consistent marginals implied by BP [Heskes & Zoeter, 2002; Heskes *et al.*, 2005; Minka, 2001a]. Langragian dual theory shows this can be equivalently be interpreted as finding saddle points of the Lagragian objective, but unfortunately the inner optimisation is not convex. Guaranteed convergent methods derived from this interpretation have been developed [Heskes & Zoeter, 2002] but these double loop algorithms are typically much slower than standard message passing EP.

Structured approximations. The methods described here have typically assumed a fully factorised variational distribution $q(x)$ but often this constraint can be relaxed to improve inference accuracy while maintaining reasonable computational tractability. A typical approach is to use tree structured variational approximations, see for example tree structured VB [Saul & Jordan, 1996], tree EP [Qi & Minka, 2004] and tree re-weighted BP [Wainwright *et al.*, 2003]. More general structured methods include Generalised BP [Yedidia *et al.*, 2004] and structured region graphs [Welling *et al.*, 2005].

Connection to EM. The Expectation-Maximisation algorithm [Dempster *et al.*, 1977] is a popular method to marginalise (the E-step) over certain variables in a model, and maximise (the M-step) over others. The E-step may be performed exactly, e.g. using BP, or approximately using VB (known as "variational EM") or EP [Kim & Ghahramani, 2006; Minka & Lafferty, 2002]. EM arises naturally as a special case of the VB framework by specifying a degenerate point mass variational distribution on the variables to be maximised over.

Statistical properties. An important property of any parameter estimator is consistency: that the estimate converges to the "true" value in the large sample limit. This is true for example for maximum likelihood estimators under relatively mild regularity conditions [Wald, 1949]. In a Bayesian setting we can ask whether the posterior concentrates to a point mass around the true value [Walker & Hjort, 2001]. For approximate inference methods, such as those described above, we can ask two questions: does the mean of the approximate posterior converge to the correct value, and additionally does the variance of the approximation approach the variance of the true posterior asymptotically? Some results are known for specific models: VB applied to a mixture of multivariate normals converges locally to the true mean but underestimates the posterior covariance [Wang & Titterington, 2006]. For mixtures of known densities there have been recent theoretical results confirming that EP does not suffer from the same problem [Titterington, 2011].

Infer.NET [Minka *et al.*, 2010] is a probabilistic programming language which I have been involved in developing at Microsoft Research Cambridge. A user is able to write down a broad class of probabilistic generative models, condition on data and automatically perform inference using EP, VMP or Gibbs sampling.

Chapter 6

Non-conjugate Variational Message Passing

A proportion of the work in this chapter was published in Knowles & Minka [2011]. As described in the previous chapter, Variational Message Passing (VMP) is an algorithmic implementation of the Variational Bayes (VB) method which applies only in the special case of conditionally conjugate exponential family models (see Section 5.10). In this chapter we propose an extension to VMP, which we refer to as Non-conjugate Variational Message Passing (NCVMP) which aims to alleviate this restriction while maintaining modularity, allowing choice in how expectations are calculated, and integrating into an existing message-passing framework: Infer.NET. We demonstrate NCVMP on logistic binary and multinomial classification. In the multinomial case we introduce a novel variational bound for the softmax factor which is tighter than other commonly used bounds whilst maintaining computational tractability.

Unfortunately, VMP is effectively limited to conjugate-exponential models since otherwise the messages become exponentially more complex at each iteration. In conjugate exponential models this is avoided due to the closure of exponential family distributions under multiplication. There are many non-conjugate problems which arise in Bayesian statistics, for example logistic regression or learning the hyperparameters of a Dirichlet.

Previous work extending Variational Bayes to non-conjugate models has fo-

cused on two aspects. The first is how to fit the variational parameters when the VB free form updates are not viable. Various authors have used standard numerical optimization techniques [Blei & Lafferty, 2007; Opper & Archambeau, 2009; Raiko *et al.*, 2007], or adapted such methods to be more suitable for this problem [Honkela *et al.*, 2007, 2010]. A disadvantage of this approach is that the convenient and efficient message-passing formulation is lost.

The second line of work applying VB to non-conjugate models involves deriving lower bounds to approximate the expectations required to calculate the KL divergence. We contribute to this line of work by proposing and evaluating a new bound for the useful softmax factor, which is tighter than other commonly used bounds whilst maintaining computational tractability [Bouchard, 2007; Jaakkola & Jordan, 1996; Khan *et al.*, 2010; Marlin *et al.*, 2011; Saul & Jordan, 1999]. We also demonstrate, in agreement with Wand *et al.* [2010] and Nickisch & Rasmussen [2008], that for univariate expectations such as required for logistic regression, carefully designed quadrature methods can be effective.

Existing methods typically represent a compromise on modularity or performance. To maintain modularity one is effectively constrained to use exponential family bounds (e.g. quadratic in the Gaussian case [Bouchard, 2007; Jaakkola & Jordan, 1996]) which we will show often gives sub-optimal performance. Methods which uses more general bounds, e.g. Blei & Lafferty [2007], must then resort to numerical optimisation, and sacrifice modularity. This is a particular disadvantage for an inference framework such as Infer.NET [Minka *et al.*, 2010] where we want to allow modular construction of inference algorithms from arbitrary deterministic and stochastic factors. We propose a novel message passing algorithm, which we call Non-conjugate Variational Message Passing (NCVMP), which generalises VMP and gives a recipe for calculating messages out of any factor. NCVMP gives much greater freedom in how expectations are taken (using bounds or quadrature) so that performance can be maintained along with modularity.

For conditionally conjugate exponential models the messages to a particular variable x_i, will all be in the same exponential family. Thus calculating the current approximate marginal $q_i(x_i)$ simply involves summing sufficient statistics. If, however, our model is not conjugate-exponential, there will be a variable x_i

which receives incoming messages which are in different exponential families, or which are not even exponential family distributions at all. Thus $q_i(x_i)$ will be some more complex distribution. Computing the required expectations becomes more involved, and worse still the complexity of the messages (e.g. the number of possible modes) may grow exponentially per iteration.

One approach, in the spirit of Expectation Propagation and following Minka [2005], would be to directly minimise the local exclusive KL. However, this in general requires an inner loop to find the KL minimising message, and does not have the attractive VMP feature of being able to connect arbitrary deterministic and stochastic factors (see Section 5.11).

Another possibility would be to project the standard VMP message onto the closest exponential family distribution in terms of KL divergence. From Equation 5.55 the standard VMP message from a factor a to a variable i is

$$m_{a\to i}(x_i) = \exp \int \log f_a(\mathbf{x}_a) \prod_{j \in \text{ne}(a) \neg i} q_j(x_j) dx_j. \tag{6.1}$$

We would then calculate an approximate message:

$$\tilde{m}_{a\to i}(x_i) := \arg\min_{s \in \mathcal{F}} \text{KL}(s(x_i) || m_{a\to i}(x_i)), \tag{6.2}$$

where $\mathcal{F}$ is the appropriate exponential family. There are two problems with this approach. Firstly, it does not reach a minimum of the global KL divergence because the procedure is not equivalent to directly minimising the local KL under the exponential family constraint. Secondly, it is computationally demanding since an inner loop is required in general to minimise the KL divergence in the second step.

The outline of the chapter is as follows. Section 6.1 is the main contribution of the chapter: the Non-conjugate VMP algorithm and its theoretical properties. Sections 6.2—6.5 describe four different applications of NCVMP: inferring the shape of a Gamma prior, modelling heteroskedasticity using the "exp" factor, and binary logistic and multinomial softmax classification models. For the classification models results on synthetic and standard UCI datasets are given in Section 6.6 and some conclusions are drawn in Section 6.7.

6.1 Algorithm and theoretical properties

In this section we give some criteria under which the algorithm was conceived. We set up required notation and describe the algorithm, and prove some important properties. Finally we give some intuition about what the algorithm is doing. The approach we take aims to fulfill certain criteria:

1. provides a recipe for any factor
2. reduces to standard VMP in the case of conjugate exponential factors
3. does not require an inner loop
4. allows modular implementation and combining of deterministic and stochastic factors

The key intuition behind NCVMP is to ensure the gradients of the approximate KL divergence implied by the message match the gradients of the true KL. This means that we will have a fixed point at the correct point in parameter space: the algorithm will be at a fixed point if the gradient of the KL is zero. Perhaps a more intuitive explanation is that NCVMP is iteratively approximating the non-conjugate model by a conjugate model.

We use the following notation: variable x_i has current variational posterior $q_i(x_i;\theta_i)$, where θ_i is the vector of natural parameters of the exponential family distribution q_i. Each factor f_a which is a neighbour of x_i sends a message $m_{a\to i}(x_i;\phi_{a\to i})$ to x_i, where $m_{a\to i}$ is in the same exponential family as q_i, i.e.

$$m_{a\to i}(x_i;\phi) = \exp(\phi^T \mathbf{u}(x_i) - \kappa(\phi)) \tag{6.3}$$

$$q_i(x_i;\theta) = \exp(\theta^T \mathbf{u}(x_i) - \kappa(\theta)), \tag{6.4}$$

where $\mathbf{u}(\cdot)$ are sufficient statistics, and $\kappa(\cdot)$ is the log partition function. We define $C(\theta)$ as the Hessian of $\kappa(\cdot)$ evaluated at θ, i.e.

$$C_{ij}(\theta) = \frac{\partial^2 \kappa(\theta)}{\partial\theta_i \partial\theta_j}. \tag{6.5}$$

It is straightforward to show that $C(\theta) = \text{cov}(\mathbf{u}(x)|\theta)$, i.e. the covariance matrix of the sufficient statistics under q_i, so if the exponential family q_i is identifiable, C will be symmetric positive definite, and therefore invertible. The factor f_a contributes a term

$$S_a(\theta_i) = \int q_i(x_i;\theta_i)\langle \log f_a(\mathbf{x})\rangle_{\neg q_i(x_i)} dx_i, \tag{6.6}$$

to the KL divergence, where we have only made the dependence on θ_i explicit: this term is also a function of the variational parameters of the other variables neighbouring the factor f_a. With this notation in place we are now able to describe the NCVMP algorithm.

Algorithm 2 Non-conjugate Variational Message Passing

```
Initialise all variables to uniform θ_i := 0 ∀i
while not converged do
    for all variables i do
        for all neighbouring factors a ∈ N(i) do
            φ_{a→i} := C(θ_i)^{-1} ∂S_a(θ_i)/∂θ_i
        end for
        θ_i := Σ_{a∈N(i)} φ_{a→i}
    end for
end while
```

To motivate Algorithm 2 we give a rough proof that we will have a fixed point at the correct point in parameter space: the algorithm will be at a fixed point if the gradient of the KL divergence is zero.

Theorem 4. *Algorithm 2 has a fixed point at* $\{\theta_i : \forall i\}$ *if and only if* $\{\theta_i : \forall i\}$ *is a stationary point of the KL divergence* $KL(q||p)$.

Proof. Firstly define the function

$$\tilde{S}_a(\theta;\phi) := \int q_i(x_i;\theta) \log m_{a\to i}(x_i;\phi) dx_i, \tag{6.7}$$

which is an approximation to the function $S_a(\theta)$. Since q_i and $m_{a\to i}$ belong to

the same exponential family we can simplify as follows,

$$\tilde{S}_a(\theta;\phi) = \int q_i(x_i;\theta)(\phi^T \mathbf{u}(x_i) - \kappa(\phi))dx_i = \phi^T \langle \mathbf{u}(x_i)\rangle_\theta - \kappa(\phi) = \phi^T \frac{\partial \kappa(\theta)}{\partial \theta} - \kappa(\phi), \tag{6.8}$$

where $\langle\cdot\rangle_\theta$ implies expectation wrt $q_i(x_i;\theta)$ and we have used the standard property of exponential families that $\langle \mathbf{u}(x_i)\rangle_\theta = \frac{\partial \kappa(\theta)}{\partial \theta}$. Taking derivatives wrt θ we have $\frac{\partial \tilde{S}_a(\theta;\phi)}{\partial \theta} = C(\theta)\phi$. Now, the update in Algorithm 2, Line 5 for $\phi_{a\to i}$ ensures that

$$C(\theta)\phi = \frac{\partial S_a(\theta)}{\partial \theta} \quad \Leftrightarrow \quad \frac{\partial \tilde{S}_a(\theta;\phi)}{\partial \theta} = \frac{\partial S_a(\theta)}{\partial \theta}. \tag{6.9}$$

Thus this update ensures that the gradients wrt θ_i of S and $\tilde{S}$ match. The update in Algorithm 2, Line 7 for θ_i is designed to minimise the *approximate* local KL divergence for x_i:

$$\theta_i := \arg\min_\theta \left(-H[q_i(x_i,\theta)] - \sum_{a\in\mathcal{N}(i)} \tilde{S}_a(\theta;\phi_{a\to i}) \right) \tag{6.10}$$

where $H[.]$ is the entropy, and is analogous to multiplying factor to variable messages together in standard VMP. We can also see this explicitly since

$$\begin{aligned} \frac{\partial}{\partial \theta_i}\left(-H[q_i(x_i,\theta_i)] - \sum_{a\in\mathcal{N}(i)} \tilde{S}_a(\theta_i;\phi_{a\to i}) \right) &= -\frac{\partial H[q_i(x_i,\theta_i)]}{\partial \theta_i} - \sum_{a\in\mathcal{N}(i)} \frac{\partial \tilde{S}_a(\theta_i;\phi_{a\to i})}{\partial \theta_i} \\ &= C(\theta_i)\theta_i - \sum_{a\in\mathcal{N}(i)} C(\theta_i)\phi_{a\to i} \end{aligned} \tag{6.11}$$

Setting this expression equal to zero gives the update in Algorithm 2, Line 7, i.e. $\theta_i := \sum_{a\in\mathcal{N}(i)} \phi_{a\to i}$. If and only if we are at a fixed point of the algorithm, we will have

$$\frac{\partial}{\partial \theta_i}\left(-H[q_i(x_i,\theta_i)] - \sum_{a\in\mathcal{N}(i)} \tilde{S}_a(\theta_i;\phi_{a\to i}) \right) = 0 \tag{6.12}$$

for all variables i. By Equation 6.9, if and only if we are at a fixed point (so that

θ_i has not changed since updating ϕ) we have

$$-\frac{\partial H[q_i(x_i,\theta_i)]}{\partial \theta_i} - \sum_{a\in\mathcal{N}(i)} \frac{\partial S_a(\theta_i)}{\partial \theta_i} = \frac{\partial KL(q||p)}{\partial \theta_i} = 0 \tag{6.13}$$

for all variables i. □

Theorem 4 gives some intuition about what NCVMP is doing. $\tilde{S}_a$ is a conjugate approximation to the true S_a function, chosen to have the correct gradients at the current value of the variational parameter, θ_i. The update at variable x_i for θ_i combines all these approximations from factors involving x_i to get an approximation to the local KL, and then moves θ_i to the minimum of this approximation. Theorem 4 shows that if NCVMP converges to a fixed point then it is at a stationary point of the KL divergence $KL(q||p)$. However, unlike VMP we have no guarantee to decrease $KL(q||p)$ at every step, and indeed we do sometimes encounter convergence problems which require damping to fix: see Section 6.7.

Another important property of Non-conjugate VMP is that it reduces to standard VMP for conditionally conjugate factors (see Section 5.10 for a defintion of conditional conjugacy).

Theorem 5. *If $\langle \log f_a(\mathbf{x})\rangle_{\neg q_i(x_i)}$ as a function of x_i can be written $\mu^T\mathbf{u}(x_i) - c$ where c is a constant, then the NCVMP message $m_{a\to i}(x_i, \phi_{a\to i})$ will be the standard VMP message $m_{a\to i}(x_i, \mu)$.*

Proof. To see this note that

$$\langle \log f_a(\mathbf{x})\rangle_{\neg q_i(x_i)} = \mu^T\mathbf{u}(x_i) - c \quad \Rightarrow \quad S_a(\theta) = \mu^T\langle \mathbf{u}(x_i)\rangle_\theta - c, \tag{6.14}$$

where μ is the expected natural statistic under the messages from the variables connected to f_a other than x_i. We have

$$S_a(\theta) = \mu^T\frac{\partial \kappa(\theta)}{\partial \theta} - c \quad \Rightarrow \quad \frac{\partial S_a(\theta)}{\partial \theta} = C(\theta)\mu, \tag{6.15}$$

so from Algorithm 2, Line 5 we have

$$\phi_{a\to i} := C(\theta)^{-1}\frac{\partial S_a(\theta)}{\partial \theta} = C(\theta)^{-1}C(\theta)\mu = \mu, \tag{6.16}$$

the standard VMP message. □

The update for θ_i in Algorithm 2, Line 7 is the same as for VMP, and Theorem 5 shows that for conjugate factors the messages sent to the variables are the same as for VMP. Thus NCVMP is a generalisation of VMP.

Another appealing property of NCVMP is that like Newton's method, and unlike gradient descent, it is parameterisation invariant. Recall that NCVMP is based on matching gradients at the current posterior $\theta^{(t)}$

$$\left.\frac{\partial \tilde{S}(\theta;\phi)}{\partial \theta}\right|_{\theta=\theta^{(t)}} = \left.\frac{\partial S(\theta)}{\partial \theta}\right|_{\theta=\theta^{(t)}}. \tag{6.17}$$

Now if we reparameterise in terms of ψ with a bijective mapping $\theta = g(\psi)$ then we would derive NCVMP in terms of $S_\psi(\psi) = S(g(\psi))$ and $\tilde{S}_\psi(\psi;\phi) = \tilde{S}(g(\psi);\phi)$.

Theorem 6. *NCVMP is invariant to reparameterisation of S.*

Proof. The NCVMP gradient matching equation in the new parameterisation is

$$\frac{\partial \tilde{S}_\psi(\psi;\phi)}{\partial \psi} = \frac{\partial S_\psi(\psi)}{\partial \psi}. \tag{6.18}$$

Substituting to get the original S function this implies

$$\frac{\partial \tilde{S}(g(\psi);\phi)}{\partial \psi} = \frac{\partial S(g(\psi))}{\partial \psi}. \tag{6.19}$$

Applying the chain rule we have

$$\frac{\partial \tilde{S}(\theta;\phi)}{\partial \theta}\frac{\partial \theta}{\partial \psi} = \frac{\partial S(\theta)}{\partial \theta}\frac{\partial \theta}{\partial \psi}. \tag{6.20}$$

The Jacobian matrix $\frac{\partial \theta}{\partial \psi}$ is full rank since g is bijective, so the original gradient matching scheme is recovered. □

We have shown here that the NCVMP scheme is invariant to the parameterisation of S. It should be noted of course that it is not invariant to reparameterisations of the model which change the structure of the variational posterior itself.

6.1.1 Gaussian variational distribution

As an illustration we describe the NCVMP updates for a Gaussian variational distribution $q(x) = N(x; m, v)$ and approximate factor/message $m_{f\to x} = N(x; m_f, v_f)$. Although these can be derived from the generic formula using natural parameters in Algorithm 2, Line 5 it is mathematically more convenient to use the mean and variance which is valid by Theorem 6.

$$\begin{aligned}\tilde{S}(m,v) &= \int_x N(x;m,v)\log N(x;m_f,v_f)dx \qquad (6.21)\\ &= -.5\log(2\pi v_f) - \frac{(m-m_f)^2+v}{2v_f},\\ \Rightarrow \frac{d\tilde{S}(m,v)}{dm} &= \frac{m_f-m}{v_f}, \qquad \frac{d\tilde{S}(m,v)}{dv} = -\frac{1}{2v_f}. \qquad (6.22)\end{aligned}$$

Matching gradients wrt (m, v) and solving for the natural parameters gives

$$\frac{1}{v_f} = -2\frac{dS(m,v)}{dv}, \qquad \frac{m_f}{v_f} = \frac{m}{v_f} + \frac{dS(m,v)}{dm}. \qquad (6.23)$$

Thus we see that the message $m_{f\to x} = N(x; m_f, v_f)$ has a very simple form in terms of the derivatives of S wrt to the mean and variance.

6.1.2 Alternative derivation

NCVMP can alternatively be derived by assuming the incoming messages to x_i are fixed apart from $m_{a\to i}(x_i;\phi)$ and calculating a fixed point update for this message. We will focus on a particular factor f_a and variable x_i, with the aim of calculating an exponential family message $m_{a\to i}(x_i;\phi)$, parameterised by the natural parameter ϕ. Consider the local KL divergence under exponential family variational posterior $q_i(x_i;\theta_i)$, with natural parameter θ_i. Let $q_i^{\neg a}(\mathbf{x}) := \prod_{b\in \mathrm{ne}(x_i)\neg a} m_{b\to i}(x_i)$ be the "cavity distribution" and $q^{\neg i}(\mathbf{x}^{\neg i}) = \prod_{j\neq i} q_j(x_j)$ the current variational

distribution over variables other than x_i.

$$\begin{aligned}\mathrm{KL}(q_i(x_i;\theta_i)q^{\neg i}(\mathbf{x}^{\neg i})||f_a(\mathbf{x})q^{\neg a}(\mathbf{x})) = &\int q_i(x_i;\theta_i)\log q_i(x_i;\theta_i)dx_i \\ &- \int q_i(x_i;\theta_i)q^{\neg i}(\mathbf{x}^{\neg i})\log f_a(\mathbf{x})d\mathbf{x} \\ &- \int q_i(x_i;\theta_i)q^{\neg i}(\mathbf{x}^{\neg i})\log q^{\neg a}(\mathbf{x})d\mathbf{x} + \text{const.} \end{aligned} \tag{6.24}$$

The cavity distribution itself factorises as $q^{\neg a}(\mathbf{x}) = q_i^{\neg a}(x_i;\theta_i^{\neg a})q^{\neg a,i}(\mathbf{x}^{\neg i})$, where $q_i(x_i;\theta^{\neg a})$ is the product of all the other incoming messages to x_i. The local contribution to the KL is

$$\begin{aligned}\mathrm{KL}(\theta_i) =& - H[q_i(x_i;\theta_i)] \\ &- \int q_i(x_i;\theta_i)\langle\log f_a(\mathbf{x})\rangle_{\sim q_i(x_i)}dx_i \\ &- \int q_i(x_i;\theta_i)\log q_i(x_i;\theta^{\neg a})dx_i + \text{const.} \\ =&\theta_i^T\kappa(\theta_i) - \kappa(\theta_i) - S(\theta_i) - \theta_i^{\neg a}\kappa(\theta_i) + \kappa(\theta_i^{\neg a}) + \text{const.} \end{aligned} \tag{6.25}$$

where we have used the fact that the expectation of the sufficient statistics of an exponential family are given by the derivatives of κ. The variational posterior $q_i(x_i;\theta_i)$ will be updated to $m_{a\to i}(x_i;\phi)q_i(x_i;\theta_i^{\neg a})$, so we have the relationship $\theta_i = \theta_i^{\neg a} + \phi$. We assume that $\theta_i^{\neg a}$ is fixed (which is at least true once the algorithm has converged), so differentiating wrt to θ_i and ϕ is equivalent:

$$\begin{aligned}\frac{\partial \mathrm{KL}}{\partial \phi} = \frac{\partial \mathrm{KL}}{\partial \theta_i} &= C(\theta_i)\theta_i + \kappa(\theta_i) - \kappa(\theta_i) - \frac{\partial S(\theta_i)}{\partial \theta_i} - C(\theta_i)\theta^{\neg a} \\ &= C(\theta_i)\phi - \frac{\partial S(\theta_i)}{\partial \theta_i} \end{aligned} \tag{6.26}$$

where $C(\theta_i)$ is the Hessian of $\kappa(\theta_i)$. Setting this derivative to zero corresponds to a fixed point scheme for ϕ, and recovers the Non-conjugate VMP update for ϕ.

6.1.3 NCVMP as moment matching

The gradient matching operation of NCVMP can be seen as analogous to moment matching in EP. The gradient of the true S is

$$\begin{aligned}\frac{\partial S(\theta)}{\partial \theta} &= \int \frac{\partial q_i(x_i;\theta)}{\partial \theta}\langle \log f_a(\mathbf{x})\rangle_{\neg q_i(x_i)} dx_i \\ &= \int \frac{\partial \log q_i(x_i;\theta)}{\partial \theta} q_i(x_i;\theta)\langle \log f_a(\mathbf{x})\rangle_{\neg q_i(x_i)} dx_i \\ &= \int (\mathbf{u}(x_i) - \langle \mathbf{u}(x_i)\rangle_{q_i(x_i;\theta)}) q_i(x_i;\theta)\langle \log f_a(\mathbf{x})\rangle_{\neg q_i(x_i)} dx_i. \end{aligned} \tag{6.27}$$

Whereas the gradient of the approximate $\tilde{S}$ is

$$\begin{aligned}\frac{\partial \tilde{S}(\theta,\phi)}{\partial \theta} &= \int \frac{\partial q_i(x_i;\theta)}{\partial \theta} \log m_{a\to i}(x_i;\phi) dx_i \\ &= \int \frac{\partial \log q_i(x_i;\theta)}{\partial \theta} q_i(x_i;\theta)\langle \log m_{a\to i}(x_i;\phi) dx_i \\ &= \int (\mathbf{u}(x_i) - \langle \mathbf{u}(x_i)\rangle_{q_i(x_i;\theta)}) q_i(x_i;\theta) \log m_{a\to i}(x_i;\phi) dx_i. \end{aligned} \tag{6.28}$$

We see that matching gradients is equivalent to matching moments of the true and approximate *log* factors, given the current variational posterior. Thus NCVMP extends the applicability of VMP in much the same way that EP extends BP.

6.2 Gamma prior with unknown shape

Consider a variable y which we would like to give a Gamma distribution with unknown (stochastic) shape s and rate r. The factor is

$$f(y,s,r) = \frac{y^{s-1} r^s}{\Gamma(s)} \exp(-ry) =: G(y;s,r). \tag{6.29}$$

The log factor is

$$\log f(y,s,r) = (s-1)\log y + s\log r - \log\Gamma(s) - ry. \tag{6.30}$$

The messages to the output y and rate r are standard:

$$\begin{aligned} m_{f\to y}(y) &= G(y; \langle s\rangle, \langle r\rangle) \\ m_{f\to r}(r) &= G(r; \langle s\rangle + 1, \langle y\rangle), \end{aligned} \tag{6.31}$$

The normal VMP message to s would be

$$m_{f\to s}(s) \propto \frac{\exp\{s[\langle \log y\rangle + \langle \log r\rangle]\}}{\Gamma(s)}, \tag{6.32}$$

This is in fact an exponential family distribution, but unfortunately the partition function is not known analytically. Using non-conjugate VMP we can calculate the appropriate message under the constraint that the shapes s has a Gamma variational distribution. The message to s is then $G(s; c, d)$, with c, d given by

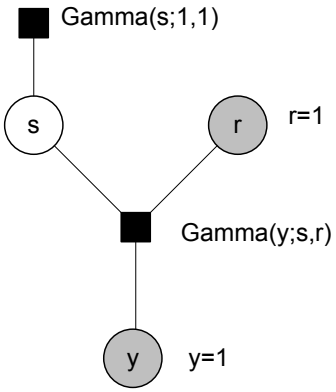
$$\begin{bmatrix} c-1 \\ d \end{bmatrix} = C(\theta)^{-1} \frac{\partial S(\theta)}{\partial \theta}, \tag{6.33}$$

where $\theta = [a, b]^T$ is the natural parameter vector for the current variational posterior on s and $C(\theta)$ is the Hessian of $\kappa(\theta) = \log\Gamma(a) - a\log b$, which is

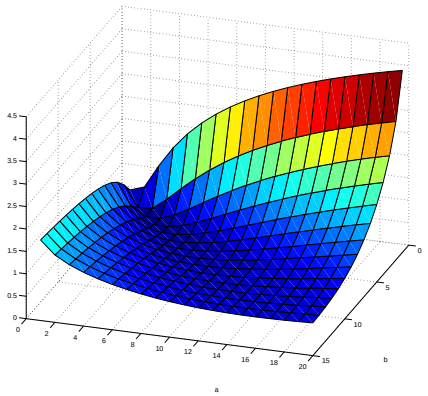
$$C(\theta) = \begin{bmatrix} \psi'(a) & -1/b \\ -1/b & a/b^2 \end{bmatrix}, \tag{6.34}$$

where $\psi'(.)$ is the trigamma function. Calculating $\frac{\partial S(\theta)}{\partial \theta}$ requires quadrature: an efficient and accurate approach is detailed in Appendix 6.A.

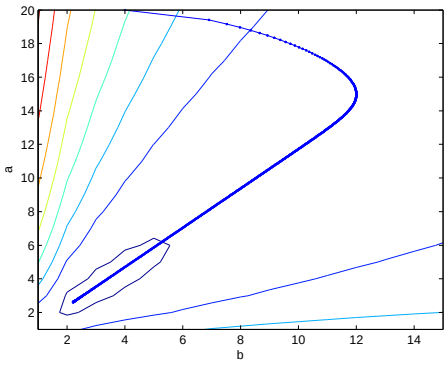
The simplest possible usage of this nonconjugate factor is to fix $r = 1$, $y = 1$, and set a Gamma$(1, 1)$ prior on s (Figure 6.1a). In this case we can analytically calculate the KL divergence between the true posterior of s and a Gamma(a, b) approximation. The KL divergence is plotted as a function of the variational parameters a and b in Figure 6.1b. Gradient descent is very slow due to the flat region for large values of a and b (Figure 6.1c). Nonconjugate VMP performs much better here, reaching the minimum in only a few steps (Figure 6.1c). The computational cost of nonconjugate VMP is the same as gradient descent, requiring only first derivatives of the KL divergence.



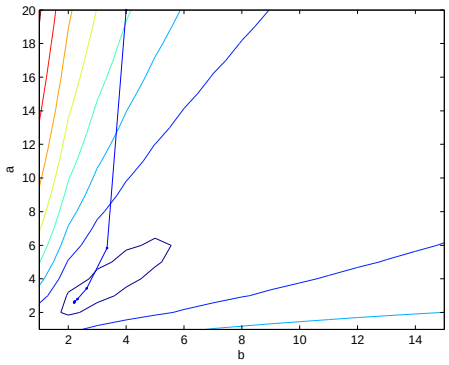
(a) Simplest usage case



(b) KL divergence as a function of the variational parameters



(c) Steps taken by gradient descent (learning rate = 1)



(d) Steps taken by nonconjugate VMP (no damping)

Figure 6.1: Fitting a Gamma distribution using NCVMP. In this example nonconjugate VMP is much faster than gradient descent, at the same computational cost.

6.2.1 Estimating a Gamma distribution

The next simplest application of the Gamma factor is to estimate the shape and rate of an unknown Gamma distribution. Our model is

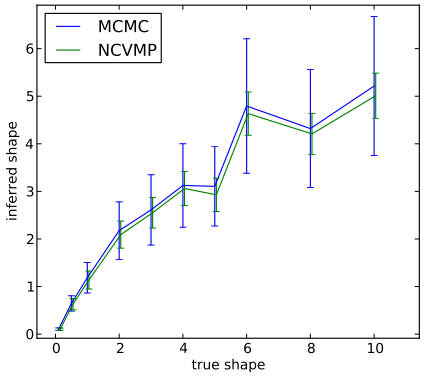
$$\begin{aligned} s &\sim G(1,1) \\ r &\sim G(1,1) \\ x_i &\sim G(s,r) \qquad i.i.d. \quad \forall i = 1,...,N \end{aligned} \tag{6.35}$$

The $\{x_i : i = 1 \dots N\}$ are the observations, and we put Gamma variational distributions on s and r. To test NCVMP in this setting, we generate synthetic data from a Gamma distribution and vary either the true shape (Figure 6.2a) or the sample size (Figure 6.2b). The "true" posterior was found by running 10,000 iterations of slice sampling. The general pattern is that NCVMP is able to find the posterior mean quite accurately, but the variational posterior variance is often two or three times smaller than the true posterior variance. This can be explained by the strong posterior coupling of the shape s and rate r illustrated in Figure 6.2c. The mean field factored approximation, $q(r,s) = q(r)q(s)$ is clearly poor here. Possible solutions would be to reparameterise in terms of the mean s/r and variance s/r^2, since these quantities are less strongly coupled a posteriori (see Figure 6.2d), or to use a multivariate variational posterior on s and r such as the Wishart distribution. These possibilities are left to future research.

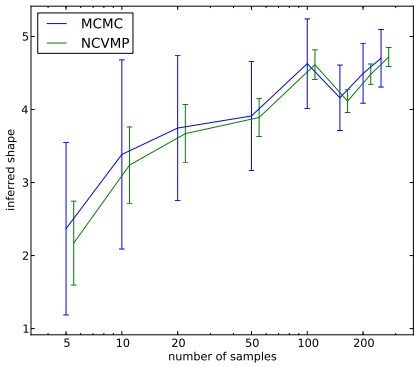
6.3 A deterministic factor: "Exp"

Consider a deterministic factor representing the exponential function, $\delta(y - e^x)$. The exponent x is the parent and the output y is the child (Figure 6.3a). Let the incoming messages be:

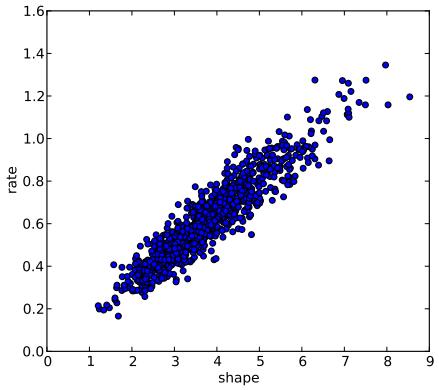
$$\begin{aligned} q(x) &= N(x; m, v), \\ q(y) &= G(y; a_y, b_y), \end{aligned} \tag{6.36}$$



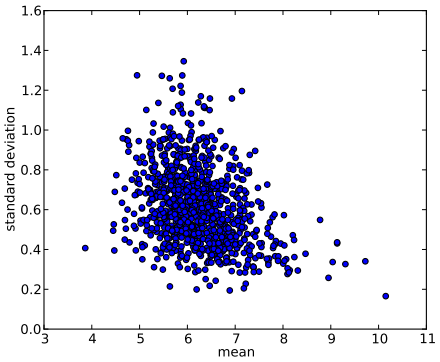
(a) Inferred vs true shape with $N = 50$ samples and rate $r = 1$. Line shows posterior mean, errors bars show $\pm$ one posterior standard deviation.



(b) Inferred shape vs sample size with true shape $s = 5$ and rate $r = 1$. Line shows posterior mean, errors bars show $\pm$ one posterior standard deviation.



(c) Posterior samples of the shape and rate for $N = 50$ samples with shape $s = 5$ and rate $r = 1$



(d) Posterior samples of the mean and variance for $N = 50$ samples with shape $s = 5$ and rate $r = 1$

Figure 6.2: Fitting a Gamma distribution.

where we are assuming that the variational posteriors for x and y will be Gaussian and Gamma respectively. The local KL divergence for this factor as a function of the variational parameters for x is shown in Figure 6.3b, where the message into y from the rest of the model is Gamma$(2, 1)$.

We now calculate the NCVMP messages for this factor. First we calculate the message to the exponent x. As for standard VMP for deterministic factors, we first transform the factor into a "soft constraint" (see Section 5.11) as follows:

$$\begin{aligned} f(x) &= \int \delta(y - e^x) G(y; a_y, b_y) dy \\ &= G(e^x; a_y, b_y). \end{aligned} \tag{6.37}$$

This is the "factor" that we now calculate the message for. As it happens, no quadrature is required, since the expectation of x and e^x under a Gaussian distribution are known:

$$\begin{aligned} S(m, v) &= \int N(x; m, v) \log f(x) dx \\ &= (a_y - 1)m - b_y \exp(m + v/2) \\ &\quad - \log \Gamma(a_y) + a_y \log b_y. \end{aligned} \tag{6.38}$$

From Equation 6.23 the message to x will be $N(x; m_f, v_f)$ where

$$\begin{aligned} \frac{1}{v_f} &= b_y \exp(m + v/2), \\ \frac{m_f}{v_f} &= \frac{m}{v_f} + a_y - 1 - b_y \exp(m + v/2). \end{aligned} \tag{6.39}$$

Since this is a deterministic factor the message to y simply has the expected sufficient statistics of $f(x)$ under $q(x)$, i.e. the Gamma distribution, $G(y; a, b)$ where

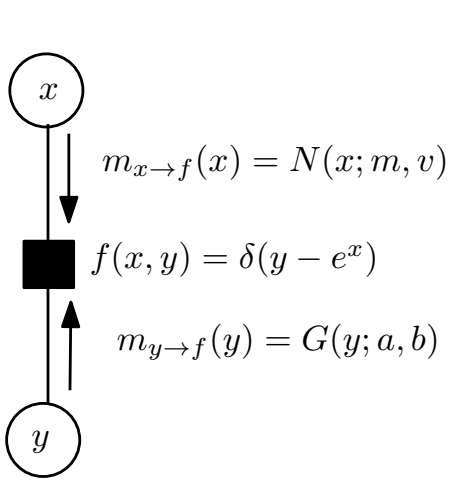
$$\begin{aligned} \langle y \rangle = \langle f(x) \rangle_q &\Rightarrow \frac{a}{b} = \exp(m + v/2), \\ \langle \log y \rangle = \langle \log f(x) \rangle_q &\Rightarrow \psi(a) - \log b = m. \end{aligned} \tag{6.40}$$

To solve for a and b one option would be to substitute $b = a/\exp(m + v/2)$ into

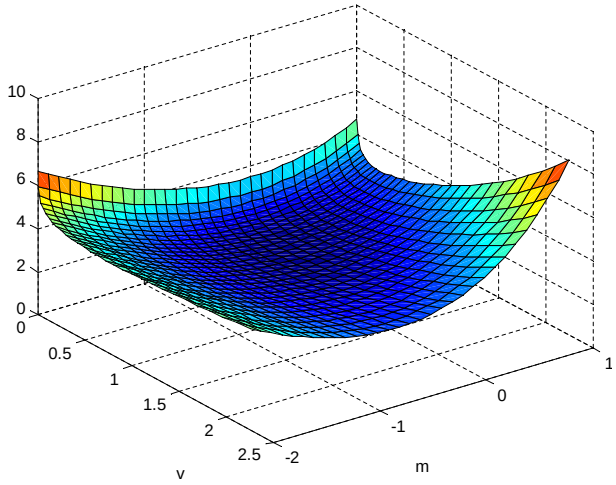
Equation 6.40 to give

$$\xi(a) := \log a - \psi(a) = v/2. \tag{6.41}$$

The inverse of the monotonic function $\xi(a)$ is then required, which could be tabulated. We take an alternative approach, using the generalised Newton iteration described in Minka [2002]. We now demonstrate the application of this factor to incorporating heteroskedasticity in linear and non-linear regression models.



(a) Exp factor.



(b) KL as function of variational parameters of x.

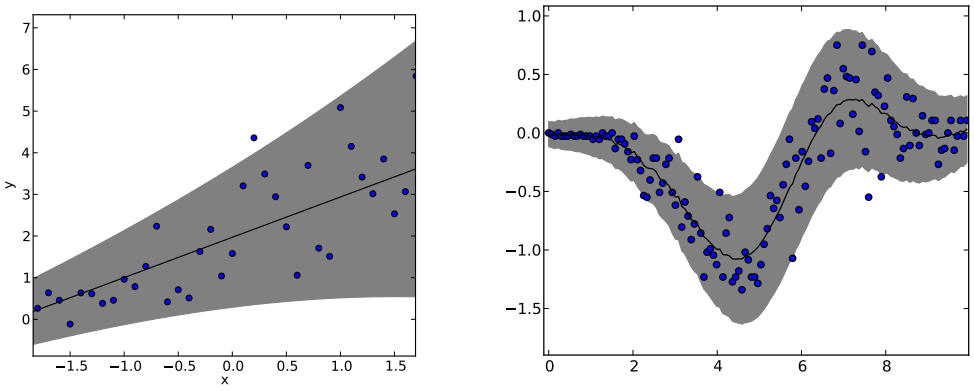
Figure 6.3: Exponential factor.

6.3.1 Heteroskedastic regression.

A common problem for standard regression models is heteroskedasticity: noise variance that is not constant as a function of the inputs. Consider data in pairs $\{(\mathbf{x}_i \in \mathbb{R}^P, y_i \in \mathbb{R}) : i = 1 \dots N\}$, where we wish to model $P(y|\mathbf{x})$. Using the exponential factor introduced above, it is simple to implement the following model

$$\begin{aligned} y_i &= \mathbf{w}^T \mathbf{x}_i + \epsilon_i \\ \epsilon_i &\sim N(0, \exp(\boldsymbol{\beta}^T \mathbf{x}_i)) \end{aligned} \tag{6.42}$$

The result of fitting this model to synthetic data (generated from the model), using NCVMP, is shown in Figure 6.4a. EP could in principle be used for this model, but would require quadrature for both the exponential and normal factors, significantly increasing computational cost. In fact we also find EP can have convergence problems for this model (note that the posterior distribution of $(\mathbf{w}, \beta)$ is not log concave).



(a) Linear model, synthetic data (b) Non-linear GP model, motorcycle dataset

Figure 6.4: Fitted heteroskedastic regression models. Grey regions show two standard deviations of the predictive distribution.

It is straightforward to extend the linear model in Equation 6.42 to a non-linear model using Gaussian processes:

$$\begin{aligned} f &\sim GP(0, k_f) \\ m &\sim N(-1, 1) \\ l &\sim GP(0, k_l) \\ p(t) &\sim \exp\left(l(t) + m\right) \\ y(t) &\sim N(f(t), p(t)^{-1}). \end{aligned} \tag{6.43}$$

Here the functions f and l are draws from GPs with different kernels k_f and k_l since we might expect the noise variance to vary more slowly than the signal. See Rasmussen & Williams [2006] for a thorough introduction to Gaussian processes. f gives the mean of the signal, whereas l represents the log precision of the noise

up to a constant m. This offset is included since otherwise the noise variance is very large under the prior. This model was proposed in Adams & Stegle [2008], where EP was used for inference. In Figure 6.4b we show this model fitted to the well known "motorcycle" dataset [Parker & Rice, 1985] using NCVMP in Infer.NET. We use a squared exponential kernel for both k_f and k_l and optimise the log length scale of each using gradient descent on the variational lower bound.

6.4 Logistic regression models

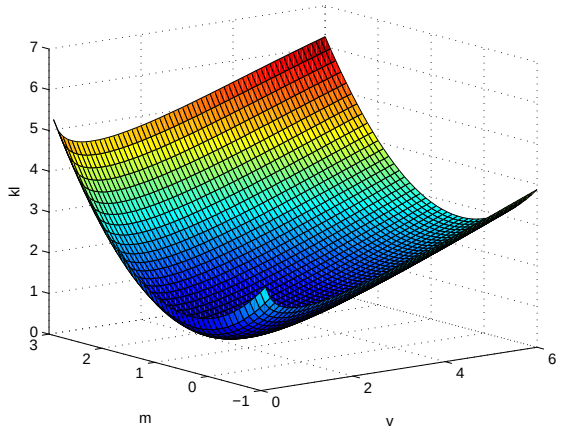


Figure 6.5: KL as a function of variational parameters of x for the Bernoulli-From-Log-Odds factor required for logistic regression.

We illustrate NCVMP on Bayesian binary and multinomial logistic classification. The regression part of the model is standard:

$$g_{kn} = \sum_{d=1}^{D} W_{kd} X_{dn} + m_k \tag{6.44}$$

where g is the auxiliary variable, W is a matrix of weights with standard normal prior, X is the design matrix and m is a per class mean, which is also given a standard normal prior. We use a multivariate normal variational posterior on the rows of W, and a univariate normal variational posterior for m_k. For binary classification we just have $k = 1$, and the observation model is $p(y = 1|g_{1n}) = \sigma(g_{1n})$ where $\sigma(x) = 1/(1 + e^{-x})$ is the logistic function. In the multinomial case

$p(y = k|g_{:n}) = \sigma_k(g_{:n})$ where

$$\sigma_k(\mathbf{x}) = \frac{e^{x_k}}{\sum_l e^{x_l}}, \tag{6.45}$$

is the "softmax" function. The auxiliary variables g are given univariate normal "pseudo"-marginals as described in Section 5.11. The VMP messages for the regression part of the model are standard so we omit the details.

6.4.1 Binary logistic classification

For logistic regression we require the following factor: $f(s, x) = \sigma(x)^s(1-\sigma(x))^{1-s}$ where we assume s is observed. The log factor is $sx - \log(1 + e^x)$. It is useful to note that fitting the logistic factor in isolation is a convex problem wrt to the variational parameters.

Lemma 3. *If $q(x) = N(x; \mu, \sigma^2)$ and $p(x) = f(x, s)N(x; m, v)$ then $KL(q||p)$ is jointly convex in (μ, σ^2).*

Proof. p is log-concave in x, so $h(x) = -\log p(x)$ is convex

$$\begin{aligned} KL(q||p) &= \int N(x; \mu, \sigma^2)h(x)dx - H[q] \\ &= \int \underbrace{N(y; 0, 1)}_{\text{positive}} \underbrace{h(y\sigma + \mu)}_{\text{affine}} dy - \log \sigma + \text{const.}, \end{aligned} \tag{6.46}$$

which is convex in (μ, σ), see for example Chapter 3.2, Boyd & Vandenberghe [2004]. □

There are two problems: we cannot analytically compute expectations wrt to x, and we need to optimise the variational parameters. Jaakkola & Jordan [1996] propose the "quadratic" bound on the integrand

$$\sigma(x) \geq \tilde{\sigma}(x, t) = \sigma(t) \exp\left((x - t)/2 - \frac{\lambda(t)}{2}(x^2 - t^2)\right), \tag{6.47}$$

where $\lambda(t) = \frac{\tanh(t/2)}{t} = \frac{\sigma(t)-1/2}{t}$. It is straightforward to analytically optimise t to make the bound as tight as possible. The bound is conjugate to a Gaussian,

but we will see that its performance can be poor. An alternative proposed in Saul & Jordan [1999] is to bound the integral:

$$\langle \log f(s,x) \rangle_q \geq sm - \frac{1}{2}a^2 v - \log(1 + e^{m+(1-2a)v/2}), \tag{6.48}$$

where m, v are the mean and variance of $q(x)$ and a is a variational parameter which can be optimised using the fixed point iteration

$$a := \sigma(m - (1-2a)v/2). \tag{6.49}$$

We refer to this as the "tilted" bound. This bound is not conjugate to a Gaussian, but we can calculate the NCVMP message $N(m_f, v_f)$, which from Equation 6.23 has parameters:

$$\frac{1}{v_f} = a(1-a), \qquad \frac{m_f}{v_f} = \frac{m}{v_f} + s - a, \tag{6.50}$$

where we have assumed a has been optimised. A final possibility is to use quadrature to calculate the gradients of $S(m, v)$ directly. The NCVMP message then has parameters

$$\frac{1}{v_f} = \frac{\langle x\sigma(x) \rangle_q - m\langle \sigma(x) \rangle_q}{v}, \qquad \frac{m_f}{v_f} = \frac{m}{v_f} + s - \langle \sigma(x) \rangle_q. \tag{6.51}$$

The univariate expectations $\langle \sigma(x) \rangle_q$ and $\langle x\sigma(x) \rangle_q$ can be efficiently computed using Gauss-Hermite or Clenshaw-Curtis quadrature (see Trefethen [2008] for a good comparison of these methods).

6.5 The softmax factor

Consider the deterministic softmax factor

$$f(x,p) = \prod_{k=1}^{K} \delta\left(p_k - \sigma_k(\mathbf{x})\right), \tag{6.52}$$

where x_k are real valued, p is a probability vector with incoming Dirichlet message, $m_{\to p}(p) = \text{Dir}(p; d)$, and recall that $\sigma_k(\mathbf{x}) := e^{x_k} / \sum_l e^{x_l}$. We can integrate out p to give the log factor

$$\log f(x) = \sum_k (d_k - 1)x_k - (d_\cdot - K) \log \sum_l e^{x_l}, \tag{6.53}$$

where we define $d_\cdot := \sum_{k=1}^K d_k$. Let the incoming message from x be the fully factorised variational posterior, $q(x) = \prod_{k=1}^K N(x_k; m_k, v_k)$. The expected log factor is then

$$S(m, v) \leq \int q(x) \log f(x) dx = \sum_{k=1}^K (d_k - 1)m_k - (d_\cdot - K) \left\langle \log \sum_l e^{x_l} \right\rangle_q \tag{6.54}$$

How should we deal with the $\langle \log \sum_l e^{x_l} \rangle$ term? To maintain a lower bound on the marginal likelihood and $S(m, v)$ we require an *upper* bound on $\langle \log \sum_l e^{x_l} \rangle$.

6.5.1 Existing approaches

Log bound. The approach used by Blei & Lafferty [2007] is a linear Taylor expansion of the log, which is accurate for small variances v:

$$\langle \log \sum_i e^{x_i} \rangle \leq \log \sum_i \langle e^{x_i} \rangle = \log \sum_i e^{m_i + v_i/2}, \tag{6.55}$$

which we refer to as the "log" bound. In the statistics literature this is known as the 0-th order delta method. We now have

$$S(m, v) \leq \sum_{k=1}^K (d_k - 1)m_k - (d_\cdot - K) \log \sum_i e^{m_i + v_i/2}. \tag{6.56}$$

The messages are still not conjugate, so some numerical method must be used to learn m and v: while Blei & Lafferty [2007] used LBFGS [Liu & Nocedal, 1989] we will use NCVMP, which from Equation 6.23 has message $\prod_k N(x_k; m_{kf}, v_{kf})$

with

$$\frac{1}{v_{kf}} = (d_\cdot - K)\sigma_k, \qquad \frac{m_{kf}}{v_{kf}} = \frac{m_k}{v_{kf}} + (d_k - 1) - (d_\cdot - K)\sigma_k, \tag{6.57}$$

where $\sigma_k := \sigma_k(\mathbf{m} + \mathbf{v}/2)$.

Quadratic bound. Another bound was proposed by Bouchard [2007]:

$$\log \sum_{k=1}^{K} e^{x_k} \leq a + \sum_{k=1}^{K} \log(1 + e^{x_k - a}), \tag{6.58}$$

where a is a new variational parameter. Combining with Equation 6.47 we get the "quadratic bound" on the integrand, with $K + 1$ variational parameters:

$$\log \sum_{k=1}^{K} e^{x_k} \leq a + \sum_{k=1}^{K} \frac{x_k - a - t_k}{2} + \lambda(t_k)[(x_k - a)^2 - t_k^2] - \log \sigma(-t_k), \tag{6.59}$$

where t are new variational parameters and $\lambda(t) = \frac{1}{2t}\left[\frac{1}{1+e^{-t}} - \frac{1}{2}\right]$. The overall log factor, $\log f(x)$ is then lower bounded by

$$\sum_{k=1}^{K}(d_k - 1)x_k - (d_\cdot - K)\left(a - \sum_{k=1}^{K}\left[\frac{x_k - a - t_k}{2} + \lambda(t_k)[(x_k - a)^2 - t_k^2] - \log \sigma(-t_k)\right]\right). \tag{6.60}$$

This directly gives Gaussian messages, $N(x_k; m_{kf}, v_{kf})$ where

$$\frac{1}{v_{kf}} = 2(d_\cdot - K)\lambda(t_k), \qquad \frac{m_{kf}}{v_{kf}} = (d_k - 1) - (d_\cdot - K)\left(1/2 - 2a\lambda(t_k)\right). \tag{6.61}$$

So modularity can be achieved without NCVMP, but as we will see, results are often poor. The question remains how to find a and $\{t_k\}$, which we defer to Appendix 6.B.

Taylor series approximation. We can use a Taylor series expansion about the mean of x. This will not give a bound, but may be more accurate and is

cheap to compute:

$$\log \sum_i e^{x_i} \approx \log \sum_i e^{m_i} + \sum_i (x_i - m_i)\sigma_i(\mathbf{m}) + \frac{1}{2}\sum_i (x_i - m_i)^2 \sigma_i(\mathbf{m})[1 - \sigma_i(\mathbf{m})] \tag{6.62}$$

We have not included the cross terms of the Hessian because we are using a fully factorised variational posterior for x so these terms would not contribute to the messages to x anyway. Taking expectations we find

$$\langle \log \sum_i e^{x_i} \rangle_q \approx \log \sum_i e^{m_i} + \frac{1}{2}\sum_i v_i \sigma_i(\mathbf{m})[1 - \sigma_i(\mathbf{m})]. \tag{6.63}$$

This approximation is similar in spirit to Laplace's approximation (see MacKay [2003], Chapter 27), except that we calculate the curvature around an approximate mean (calculated using VMP) rather than the MAP. The messages to x_k will be given by:

$$\begin{aligned} \frac{1}{v_{kf}} &= (d_\cdot - K)\sigma_k(\mathbf{m})(1 - \sigma_k(\mathbf{m})) \\ \frac{m_{kf}}{v_{kf}} &= d_k - 1 + \frac{m_k}{v_{kf}} - (d_\cdot - K)\sigma_k(\mathbf{m}) \end{aligned} \tag{6.64}$$

This message will always be proper (have positive variance) but there is no guarantee of global convergence since this approximation is not a bound.

Bohning's bound has the same form as the Taylor series expansion, only with a different approximation to the Hessian matrix H of $\log \sum \exp$, specifically using the bound

$$H \geq \frac{1}{2}(I - \mathbf{1}\mathbf{1}^T/K) =: H_B, \tag{6.65}$$

which results in the following bound on $\log \sum \exp$:

$$\log \sum_i e^{x_i} \leq \log \sum_i e^{m_i} + \sum_i (x_i - m_i)\sigma_i(\mathbf{m}) + \frac{1}{4}\sum_{ij} (x_i - m_i)(x_j - m_j)(\delta_{ij} - \frac{1}{K}). \tag{6.66}$$

In the case of a fully factorised distribution on x taking expectations we have:

$$\langle \log \sum_i e^{x_i} \rangle_q \leq \log \sum_i e^{m_i} + \frac{1}{4} \sum_i \left(1 - \frac{1}{K}\right) v_i \tag{6.67}$$

Analogously to the Taylor series expansion, we have the following message to x_k:

$$\begin{aligned} \frac{1}{v_{kf}} &= \frac{1}{2}(d_\cdot - K)\left(1 - \frac{1}{K}\right) \\ \frac{m_{kf}}{v_{kf}} &= d_k - 1 + \frac{m_k}{v_{kf}} - (d_\cdot - K)\sigma_k(\mathbf{m}) \end{aligned} \tag{6.68}$$

Note here that the variance is constant and depends only on $d_\cdot$ and K, and is always less than or equal to the variance of the message calculated using the Taylor series expansion.

6.5.2 A new bound

Inspired by the univariate "tilted" bound in Equation 6.48 we propose the multivariate tilted bound:

$$\begin{aligned} \langle \log \sum_i e^{x_i} \rangle &= \langle \log e^{\sum_j a_j x_j} e^{-\sum_j a_j x_j} \sum_i e^{x_i} \rangle \\ &\leq \sum_j a_j m_j + \log \sum_i \langle e^{x_i - \sum_j a_j x_j} \rangle \\ &= \sum_j a_j m_j + \log \sum_i e^{m_i - \sum_j a_j m_j + (1-2a_i)v_i/2 + \sum_j a_j^2 v_j/2} \\ &= \frac{1}{2}\sum_j a_j^2 v_j + \log \sum_i e^{m_i + (1-2a_i)v_i/2} =: \mathcal{T}(m, v, a) \end{aligned} \tag{6.69}$$

where $a \in \mathbb{R}^K$ is a new variational parameter and we have used the fact that

$$x_i - \sum_j a_j x_j \sim N\left(m_i - \sum_j a_j m_j, (1-2a_i)v_i + \sum_j a_j^2 v_j\right). \tag{6.70}$$

Setting $a_k = 0$ for all k we recover the log bound in Equation 6.55 (hence this is the "tilted" version). Taking derivatives of the bound $\mathcal{T}$ wrt a_k gives

$$\nabla_{a_k} \mathcal{T}(m, v, a) = a_k v_k - v_k \sigma_k \left[\mathbf{m} + \frac{1}{2}(\mathbf{1} - 2\mathbf{a}) \cdot \mathbf{v} \right] \tag{6.71}$$

where $\cdot$ denotes element-wise multiplication. Setting this expression equal to zero results in the fixed point update

$$\mathbf{a} := \sigma \left[\mathbf{m} + \frac{1}{2}(\mathbf{1} - 2\mathbf{a}) \cdot \mathbf{v} \right] \tag{6.72}$$

This is a $\mathcal{O}(K)$ operation since the denominator of the softmax function is shared. For the softmax factor quadrature is not viable because of the high dimensionality of the integrals. From Equation 6.23 the NCVMP messages using the tilted bound have natural parameters

$$\frac{1}{v_{kf}} = (d_\cdot - K) a_k (1 - a_k), \qquad \frac{m_{kf}}{v_{kf}} = \frac{m_k}{v_{kf}} + d_k - 1 - (d_\cdot - K) a_k, \tag{6.73}$$

where we have assumed $\mathbf{a}$ has been optimised. As an alternative we suggest choosing whether to send the message resulting from the quadratic bound or tilted bound depending on which is currently the tightest, referred to as the "adaptive" method.

6.6 Results

Here we aim to present the typical compromise between performance and modularity that NCVMP addresses. We will see that for both binary logistic and multinomial softmax models achieving conjugate updates by being constrained to quadratic bounds is sub-optimal, in terms of estimates of variational parameters, marginal likelihood estimation, and predictive performance. NCVMP gives the freedom to choose a wider class of bounds, or even use efficient quadrature methods in the univariate case, while maintaining simplicity and modularity.

6.6.1 The logistic factor

We first test the logistic factor methods of Section 6.4.1 at the task of estimating the toy model $\sigma(x)\pi(x)$ with varying Gaussian prior $\pi(x) := N(x; \mu, \sigma^2)$ (see Figure 6.6). We calculate the true mean and variance using quadrature. The quadratic bound has the largest errors for the posterior mean, and the posterior variance is severely underestimated. In contrast, NCVMP using quadrature, while being slightly more computationally costly, approximates the posterior much more accurately: the error here is due only to the VB approximation. Using the tilted bound with NCVMP gives more robust estimates of the variance than the quadratic bound as the prior mean changes. However, both the quadratic and tilted bounds underestimate the variance as the prior variance increases.

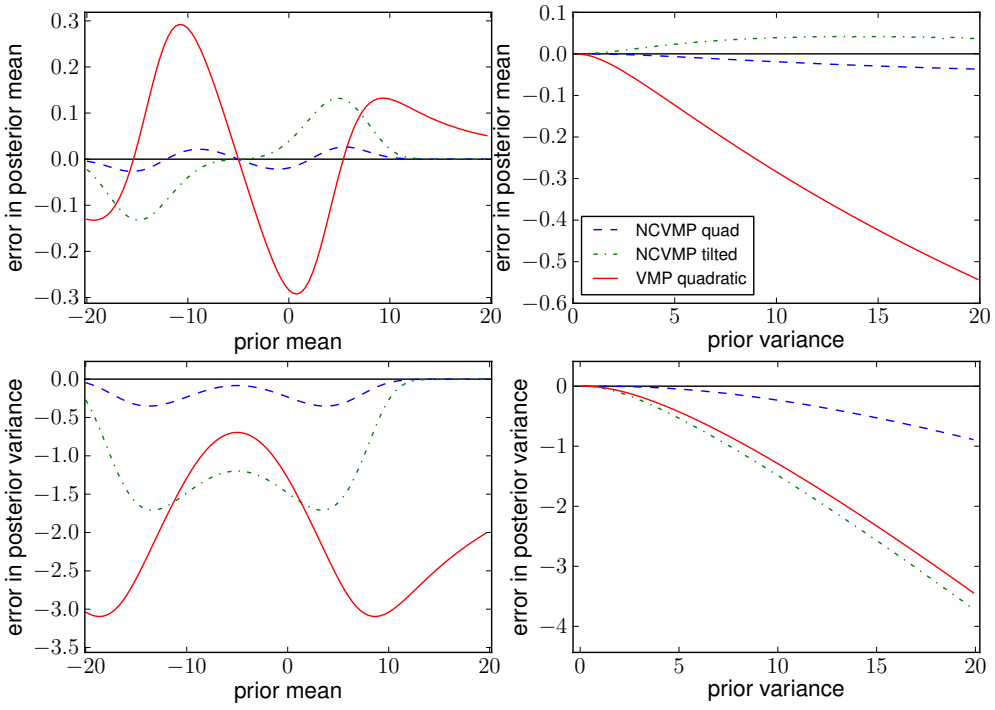
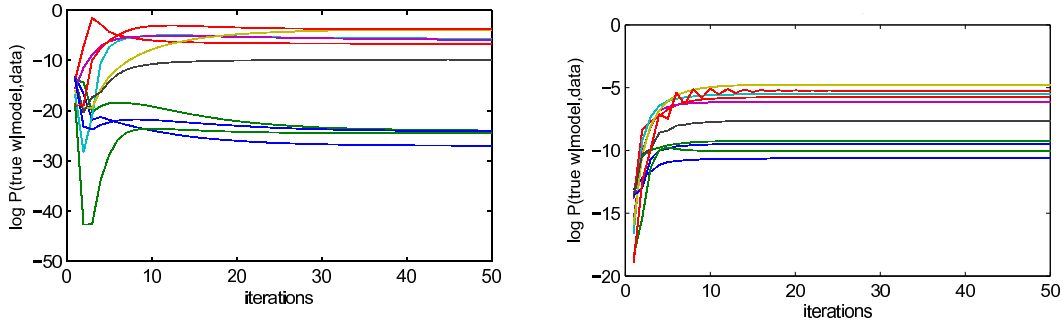


Figure 6.6: Posterior mean and variance estimates of $\sigma(x)\pi(x)$ with varying prior $\pi(x)$. Left: varying the prior mean with fixed prior variance $v = 10$. Right: varying the prior variance with fixed prior mean $m = 0$.



(a) VMP with quadratic bound (b) NCVMP with quadrature

Figure 6.7: Synthetic logistic regression experiment. Log likelihood of the true regression coefficients under the approximate posterior for 10 synthetic logistic regression datasets. Note y-axis scales.

6.6.2 Binary logistic classification

We generated ten synthetic logistic regression datasets with $N = 30$ data points and $P = 8$ covariates. We evaluated the results in terms of the log likelihood of the true regression coefficients under the approximate posterior, a measure which penalises poorly estimated posterior variances. Figure 6.7 compares the performance of non-conjugate VMP using quadrature and VMP using the quadratic bound. For four of the ten datasets the quadratic bound finds very poor solutions. Non-conjugate VMP finds a better solution in seven out of the ten datasets, and there is marginal difference in the other three. Non-conjugate VMP (with no damping) also converges faster in general, although some oscillation is seen for one of the datasets.

6.6.3 Softmax bounds

To have some idea how the various bounds for the softmax integral. $\mathbb{E}_q\left[\log \sum_{k=1}^{K} e^{x_k}\right]$ compare empirically, we calculated relative absolute error on 100 random distributions $q(x) = \prod_k N(x_k; m_k, v)$. We sample $m_k \sim N(0, u)$. When not being varied, $K = 10, u = 1, v = 1$. Ground truth was calculated using 10^5 Monte Carlo samples. We vary the number of classes, K, the distribution variance v and spread of the means u. Results are shown in Figure 6.8. As expected the tilted bound (6.69) dominates the log bound (6.55), since it is a generalisation. As K

is increased the relative error made using the quadratic bound increases, whereas both the log and the tilted bound get tighter. In agreement with Bouchard [2007] we find the strength of the quadratic bound (6.58) is in the high variance case, and Bohning's bound is very loose under all conditions. Both the log and tilted bound are extremely accurate for variances $v < 1$. In fact, the log and tilted bounds are asymptotically optimal as $v \to 0$. "Taylor" gives accurate results but is not a bound, so convergence is not guaranteed and the global bound on the marginal likelihood is lost. The spread of the means u does not have much of an effect on the tightness of these bounds. These results show that even when quadrature is not an option, much tighter bounds can be found if the constraint of requiring quadratic bounds imposed by VMP is relaxed. For the remainder of the results we consider only the quadratic, log and tilted bounds.

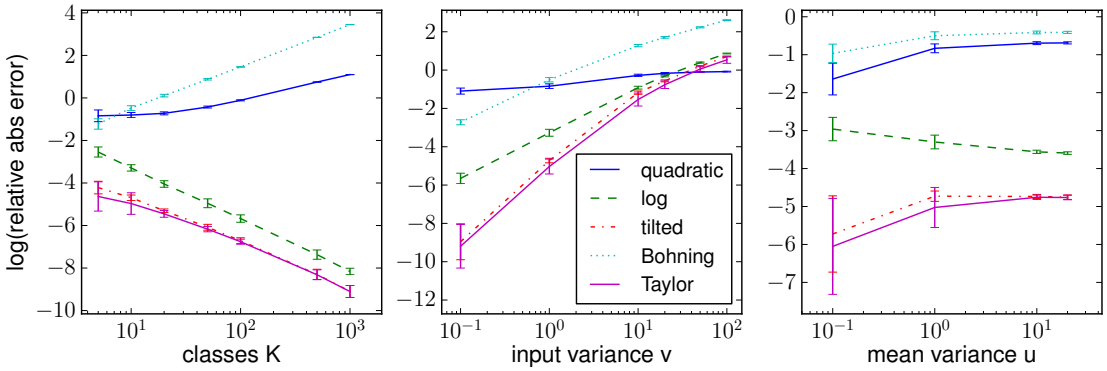


Figure 6.8: Log10 of the relative absolute error approximating $\mathbb{E} \log \sum \exp$, averaged over 100 runs.

6.6.4 Multinomial softmax classification

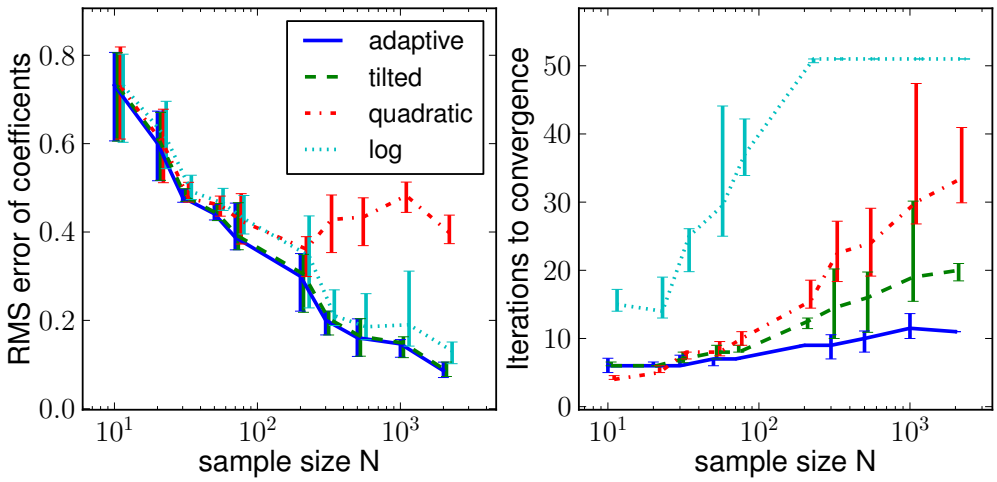
Synthetic data. For synthetic data sampled from the generative model we know the ground truth coefficients and can control characteristics of the data. We first investigate the performance with sample size N, with fixed number of features $P = 6$, classes $K = 4$, and no noise (apart from the inherent noise of the softmax function). As expected our ability to recover the ground truth regression coefficients improves with increasing N (see Figure 6.9a, left). However, we see that the methods using the tilted bound perform best, closely followed by

the log bound. Although the quadratic bound has comparable performance for small $N < 200$ it performs poorly with larger N due to its weakness at small variances. The choice of bound impacts the speed of convergence (see Figure 6.9a, right). The log bound performed almost as well as the tilted bound at recovering coefficients but takes many more iterations to converge. The extra flexibility of the tilted bound allows faster convergence, analogous to parameter expansion [Qi & Jaakkola, 2006]. For small N the tilted bound, log bound and adaptive method converge rapidly, but as N increases the quadratic bound starts to converge much more slowly, as do the tilted and adaptive methods to a lesser extent. "Adaptive" converges fastest because the quadratic bound gives good initial updates at high variance, and the tilted bound takes over once the variance decreases. We vary the level of noise in the synthetic data, fixing $N = 200$, in Figure 6.9b. For all but very large noise values the tilted and adaptive bounds perform best.

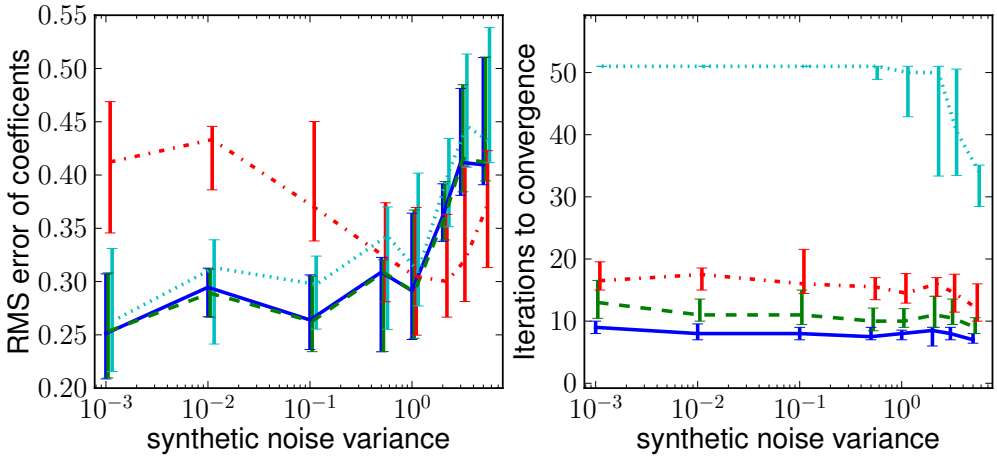
UCI datasets. We test the multinomial classification model on three standard UCI datasets: Iris ($N = 150, D = 4, K = 3$), Glass ($N = 214, D = 8, K = 6$) and Thyroid ($N = 7200, D = 21, K = 3$), see Table 6.1. Here we have also included "Probit", corresponding to a Bayesian multinomial probit regression model, estimated using VMP, and similar in setup to Girolami & Rogers [2006], except that we use EP to approximate the predictive distribution, rather than sampling. On all three datasets the marginal likelihood calculated using the tilted or adaptive bounds is optimal out of the logistic models ("Probit" has a different underlying model, so differences in marginal likelihood are confounded by the Bayes factor). In terms of predictive performance the quadratic bound seems to be slightly worse across the datasets, with the performance of the other methods varying between datasets. We did not compare to the log bound since it is dominated by the tilted bound and is considerably slower to converge.

6.7 Discussion

NCVMP is not guaranteed to converge. Indeed, for some models we have found convergence to be a problem, which can be alleviated by damping: if the NCVMP message is $m_{f\to i}(x_i)$ then send the message $m_{f\to i}(x_i)^{1-\alpha} m^{\text{old}}_{f\to i}(x_i)^{\alpha}$ where $m^{\text{old}}_{f\to i}(x_i)$



(a) Varying sample size



(b) Varying noise level

Figure 6.9: Left: root mean squared error of inferred regression coefficients. Right: iterations to convergence. Results are shown as quartiles on 16 random synthetic datasets. All the bounds except "quadratic" were fit using NCVMP.

Iris	Quadratic	Adaptive	Tilted	Probit
Marginal likelihood	-65 ± 3.5	-31.2 ± 2	-31.2 ± 2	-37.3 ± 0.79
Predictive likelihood	-0.216 ± 0.07	-0.201 ± 0.039	-0.201 ± 0.039	-0.215 ± 0.034
Predictive error	0.0892 ± 0.039	0.0642 ± 0.037	0.065 ± 0.038	0.0592 ± 0.03
Glass	Quadratic	Adaptive	Tilted	Probit
Marginal likelihood	-319 ± 5.6	-193 ± 3.9	-193 ± 5.4	-201 ± 2.6
Predictive likelihood	-0.58 ± 0.12	-0.542 ± 0.11	-0.531 ± 0.1	-0.503 ± 0.095
Predictive error	0.197 ± 0.032	0.200 ± 0.032	0.200 ± 0.032	0.195 ± 0.035
Thyroid	Quadratic	Adaptive	Tilted	Probit
Marginal likelihood	-1814 ± 43	-909 ± 30	-916 ± 31	-840 ± 18
Predictive likelihood	-0.114 ± 0.019	-0.0793 ± 0.014	-0.0753 ± 0.008	-0.0916 ± 0.010
Predictive error	0.0241 ± 0.0026	0.0225 ± 0.0024	0.0226 ± 0.0023	0.0276 ± 0.0028

Table 6.1: Average results and standard deviations on three UCI datasets, based on 16 random 50 : 50 training-test splits. Adaptive and tilted use NCVMP, quadratic and probit use VMP.

was the previous message sent to i and $0 \leq \alpha < 1$ is a damping factor. Damping does not change the fixed points of the algorithm.

We have introduced Non-conjugate Variational Message Passing, which extends variational Bayes to non-conjugate models while maintaining the convenient message passing framework of VMP and allowing freedom to choose the most accurate available method to approximate required expectations. Deterministic and stochastic factors can be combined in a modular fashion, and conjugate parts of the model can be handled with standard VMP. We have shown NCVMP to be of practical use for fitting Bayesian binary and multinomial logistic models, and shown proof of concept results on other models. We derived a new bound for the softmax integral which is tighter than other commonly used bounds, but has variational parameters that are still simple to optimise. Tightness of the bound is valuable both in terms of better approximating the posterior and giving a closer approximation to the marginal likelihood, which may be of interest for model selection.

I have implemented NCVMP for the factors described in this chapter (namely the logistic, softmax, gamma and exponential factors) in Infer.NET [Minka *et al.*, 2010], a probabilistic programming language which currently focuses on efficient message passing algorithms such as VMP and EP. If a user defines a model involving one of the factors described in this chapter and runs inference using

VMP, Infer.NET will actually use the NCVMP method presented here. Infer.NET is freely available for non-commercial use. All the experimental results in this chapter were conducted using Infer.NET.

6.A Quadrature for the gamma factor

Calculating $\frac{\partial S(\theta)}{\partial \theta}$ requires quadrature for the term

$$F(a,b) = \int G(s;a,b) \log \Gamma(s) ds. \tag{6.74}$$

Derivatives of F are given by:

$$\frac{\partial F}{\partial a} = \int G(s;a,b)(\log(sb) - \psi(a)) \log \Gamma(s) ds \tag{6.75}$$

$$\frac{\partial F}{\partial b} = -\frac{1}{b} \int s G(s;a,b) \psi(s) ds. \tag{6.76}$$

Gamma quadrature is difficult to calibrate so we apply a change of variable $x = \log s$ so the domain is the whole real line and we can use Gauss-Hermite quadrature.

$$\int_0^\infty G(s;a,b) f(s) ds = \frac{b^a}{\Gamma(a)} \int_{-\infty}^\infty \exp(ax - be^x) f(e^x) dx \tag{6.77}$$

The density $p(x) = \frac{b^a}{\Gamma(a)} \exp(ax - be^x)$ is log-concave with its mode at $\log(a/b)$ and Laplace approximation variance of $1/a$. The mean of p is given by the expectation of $\log s$ under the original Gamma distribution, which is $\psi(a) - \log b$. We will use this mean and the Laplace approximation variance for the proposal distribution when performing Gauss-Hermite quadrature. The double exponential term of p tails off very rapidly as $x \to \infty$, but the e^{ax} term tails off only slowly as $x \to -\infty$, especially for small a. We first aim to calculate the $\mathbb{E}[s\psi(s)]$ term required for Equation 6.76. Making a change of variables as in Equation 6.77 means finding the expectation of $e^x \psi(e^x)$ under p. For $x < 0$ we find that $\psi(e^x)$ is well approximated by $-e^{-x}$ (this follows from the basic identity

$\Gamma(y+1) = y\Gamma(y)$), so $e^x\psi(e^x) \approx -1$. We rewrite the expectation as

$$\mathbb{E}[e^x\psi(e^x)] = \mathbb{E}[e^x\psi(e^x) + 1] - 1 \tag{6.78}$$

The $\mathbb{E}[e^x\psi(e^x) + 1]$ term is evaluated using quadrature. This is accurate because $e^x\psi(e^x) + 1$ tends rapidly to zero as $x \to -\infty$, fortuitously squashing the heavy tail of p for $x < 0$.

To calculate the $\mathbb{E}\log\Gamma(s)$ term required for Equation 6.75 we perform a similar trick. Since $\log\Gamma(e^x)$ is well approximated by $-x$ for $x < 0$ we write the expectation as

$$\mathbb{E}\log\Gamma(e^x) = \mathbb{E}[\log\Gamma(e^x) + x] - \mathbb{E}[x] \tag{6.79}$$

The term $\mathbb{E}[\log\Gamma(e^x) + x]$ is accurately calculated using quadrature, and it is straightforward to show that the term $\mathbb{E}[x] = \psi(a) - \log b$ by transforming back to the original s domain.

The final expectation we need for Equation 6.75 is

$$\mathbb{E}_S[\log s \log\Gamma(s)] = \mathbb{E}_X[x \log\Gamma(e^x)] \tag{6.80}$$

The integrand is approximately $-x^2$ for $x < 0$ so we split the integral as

$$\mathbb{E}[x \log\Gamma(e^x)] = \mathbb{E}[x \log\Gamma(e^x) + x^2] - \mathbb{E}[x^2] \tag{6.81}$$

The term $\mathbb{E}[x^2]$ is calculated as follows:

$$\mathbb{E}_X[x^2] = \mathbb{E}_S[\log^2 s] = \psi'(a) + (\psi(a) - \log b)^2 \tag{6.82}$$

where the final expectation is found by differentiating $\int \exp[(a-1)\log x - bx]dx = \Gamma(a)/b^a$ twice wrt a.

To show the utility of these improvements to the quadrature routines, we calculate the relative error made in calculating $\frac{\partial F}{\partial a}$ and $\frac{\partial F}{\partial b}$. Table 6.2 shows the relative errors made with and without the "tricks" of this section. In all cases the relative error is decreased by several orders of magnitude, at almost no cost in computation time. An alternative is simply to increase the number of

	With tricks			Without tricks		
$\frac{\partial F}{\partial a}$	$b = 0.1$	$b = 1$	$b = 10$	$b = 0.1$	$b = 1$	$b = 10$
$a = 0.1$	-3E-15	-6.3E-15	-2.5E-15	-0.67	-0.69	-0.73
$a = 1$	3E-13	-4.2E-13	-2.4E-13	0.0021	-0.066	-0.056
$a = 10$	3.3E-13	3.3E-13	-8.1E-13	1.7E-07	9.2E-07	-6.1E-05
$\frac{\partial F}{\partial b}$	$b = 0.1$	$b = 1$	$b = 10$	$b = 0.1$	$b = 1$	$b = 10$
$a = 0.1$	1.8E-13	1.8E-15	1.5E-15	0.18	0.049	0.05
$a = 1$	-5.3E-15	2.2E-14	2.7E-16	-5E-05	0.0064	0.0013
$a = 10$	-5E-13	-5.2E-13	4.3E-13	-2.3E-09	-4.6E-08	2E-06

Table 6.2: Relative errors made in calculating derivatives of C using different quadrature methods for varying a (rows) and b (columns).

quadrature nodes used, but this is typically much less effective as well as being more computationally expensive.

6.B Optimising the variational parameters for the quadratic softmax bound

Here we discuss how to optimise the variational parameters a and $\{t_k\}$ required for the quadratic softmax bound in Equation 6.59. Taking the expectation of Equation 6.59 wrt to x we have

$$\langle \log \sum_{k=1}^{K} e^{x_k} \rangle \leq F(a,t) := \\ a + \sum_{k=1}^{K} \frac{m_k - a - t_k}{2} + \lambda(t_k)[(m_k - a)^2 + v_k - t_k^2] - \log \sigma(-t_k). \tag{6.83}$$

To minimise Equation 6.83 wrt a and $\{t_k\}$ Bouchard [2007] derives coordinate descent fixed point updates as follows:

$$a \leftarrow \frac{2\sum_{k=1}^{K} m_k \lambda(t_k) + K/2 - 1}{2\sum_{k=1}^{K} \lambda(t_k)}, \\ t_k^2 \leftarrow (m_k - a)^2 + v_k \qquad \forall k. \tag{6.84}$$

For small dimensionality and counts these fixed point iterations converge very fast. However, for large counts and dimensionality K we found that the coupling between t and a was very strong and co-ordinate-wise optimization was highly inefficient. In this regime an effective solution is to substitute the expression for t_k in Equation 6.84 into the objective function to give a univariate optimization problem in a, which can be solved efficiently using Newton's method. At the minimum we have

$$t_k(a) = \sqrt{(m_k - a)^2 + v_k} \tag{6.85}$$

Substituting this expression into Equation 6.83 we get

$$F(a) = \min_t F(a,t) = a + \sum_{k=1}^{K} \frac{m_k - a - t_k(a)}{2} - \log \sigma(-t_k(a)) \tag{6.86}$$

The derivatives of t_k wrt a are

$$\begin{aligned} t_k'(a) &= -(m_k - a)/t_k(a) \\ t_k''(a) &= 1/t_k(a) - (m_k - a)^2/t_k(a)^3 \end{aligned} \tag{6.87}$$

Using the chain rule we now find:

$$\begin{aligned} F'(a) &= 1 + \sum_k -(1 + t_k'(a))/2 + t_k'(a)\sigma(t_k(a)) \\ F''(a) &= \sum_k t_k''(a)(\sigma(t_k(a)) - .5) + t_k'(a)^2\sigma(t_k(a))\sigma(-t_k(a)) \end{aligned} \tag{6.88}$$

We can then use a Newton algorithm with Levenberg-Marquardt line search to cope with small $F''(a)$.

Chapter 7

Message Passing Algorithms for Dirichlet Diffusion Trees and Pitman Yor Diffusion Trees

Much of the work in this chapter was published in Knowles et al. *[2011a].* In this chapter we demonstrate efficient approximate inference for the Dirichlet Diffusion Tree [Neal, 2003a] and the Pitman Yor diffusion tree introduced in Chapter 4, both of which are Bayesian nonparametric priors over tree structures. We utilise the message passing algorithms introduced in Chapter 5 to approximate the Bayesian model evidence for a specific tree structure. This is used to drive sequential tree building and greedy search to find optimal tree structures, corresponding to hierarchical clusterings of the data. We demonstrate appropriate observation models for continuous and binary data. The empirical performance of our method is very close to the computationally expensive MCMC alternative on a density estimation problem, and significantly outperforms kernel density estimators.

Our algorithms use the message passing framework reviewed in Chapter 5. For many models message passing has been shown to significantly outperform sampling methods in terms of speed-accuracy trade-off. However, general α-divergence [Minka, 2005] based message passing is not guaranteed to converge, which motivates our second, guaranteed convergent, algorithm which uses mes-

sage passing within EM [Kim & Ghahramani, 2006]. Message passing methods are inherently parallelisable which is increasingly important on modern computer architectures. Running inference on the same model and same data is reproducible. Convergence is simple to assess compared to MCMC (see Section 1.2 for a more detailed discussion of the trade-offs between these methods).

Little attention seems to have been given to using message passing or deterministic methods for learning hierarchical structure, although a variational inference procedure for the nested CRP [Blei *et al.*, 2010] has recently been introduced by Wang & Blei [2009]. Efficient inference using Sequential Monte Carlo for Kingman's coalescent was demonstrated in Teh & Gorur [2009]. We leave investigating whether our framework could be adapted to the coalescent as future work.

The contributions of this chapter are as follows. We derive and demonstrate full message passing (Section 7.2) and message passing within EM algorithms (Section 7.3) to approximate the model evidence for a specific tree, including integrating over hyperparameters (Section 7.4). We show how the resulting approximate model evidence can be used to drive greedy search over tree structures (Section 7.5). We demonstrate that it is straightforward to connect different observation models to this module to model different data types, using binary vectors as an example. Finally we present experiments using the DDT and our approximate inference scheme in Section 7.7 (results for the PYDT can be found in Chapter 4).

7.1 Approximate Inference

Recall that both the DDT and PYDT define a distribution over the tree structure, $\mathcal{T}$, branching times, t and node locations, x. We assume that the likelihood of the observed data $\{y_n : n = 1, ..., N\}$ can be written as a product of conditional probabilities for each of the leaves, x_n: $\prod_n l(y_n|x_n)$. Our aim is to calculate the posterior distribution

$$P(x, t, \mathcal{T}|y) = \frac{P(y, x, t, \mathcal{T})}{\sum_{\mathcal{T}} \int P(y, x, t, \mathcal{T}) dx dt}. \tag{7.1}$$

Unfortunately, this integral is analytically intractable. Our solution is to use message passing or message passing within EM to approximate the marginal likelihood for a given tree structure: $P(y|\mathcal{T}) = \int P(y, x, t|\mathcal{T})dxdt$. We use this approximate marginal likelihood to drive tree building/search algorithm to find a weighted set of K-best trees.

7.2 Message passing algorithm

Here we describe our message passing algorithm for a fixed tree structure, $\mathcal{T}$. We employ the α-divergence framework from Minka [2005]. For each segment, $[ab] \in \mathcal{S}(\mathcal{T})$ we introduce two variables which are deterministic functions of existing variables: the branch length, $\Delta_{[ab]} = t_b - t_a$ and the variance of the Gaussian factor connecting a and b, $v_{[ab]} = \sigma^2 \Delta_{[ab]}$. We now write the unnormalized posterior as a product of factors:

$$\begin{aligned} &\prod_{n \in \text{leaves}} l(y_n|x_n) \prod_{[ab] \in \mathcal{S}(\mathcal{T})} N(x_b; x_a, v_{[ab]}) \\ &\times \delta(v_{[ab]} - \sigma^2 \Delta_{[ab]}) \delta(\Delta_{[ab]} - (t_b - t_a)) \\ &\times \mathbb{I}(0 < \Delta_{[ab]} < 1) P(t_b|\mathcal{T}), \end{aligned} \tag{7.2}$$

where $\delta(.)$ is the Dirac delta spike at 0, $\mathbb{I}(.)$ is the indicator function and $P(t_b|\mathcal{T})$ is given by Equation 4.5 for the DDT and by Equation 4.28 for the PYDT. These functions are used to break down more complex factors into simpler ones for computational and mathematical convenience. Equation 7.2 defines a factor graph over the variables, shown in Figure 7.1. Our variational approximation is fully factorized with a Gaussian q for the locations x_b, divergence times t_b, branch lengths $\Delta_{[ab]}$, variances $v_{[ab]}$ and overall variance σ^2, and a Gamma variational distribution for the divergence function parameter c.

7.2.1 Choosing α-divergences.

We choose an α for each factor f in the factor graph and then minimize the α-divergence, $D_\alpha[q^{\sim f}(W)\tilde{f}(W)||q^{\sim f}(W)f(W)]$ with respect to $\tilde{f}(W)$ where $W =$

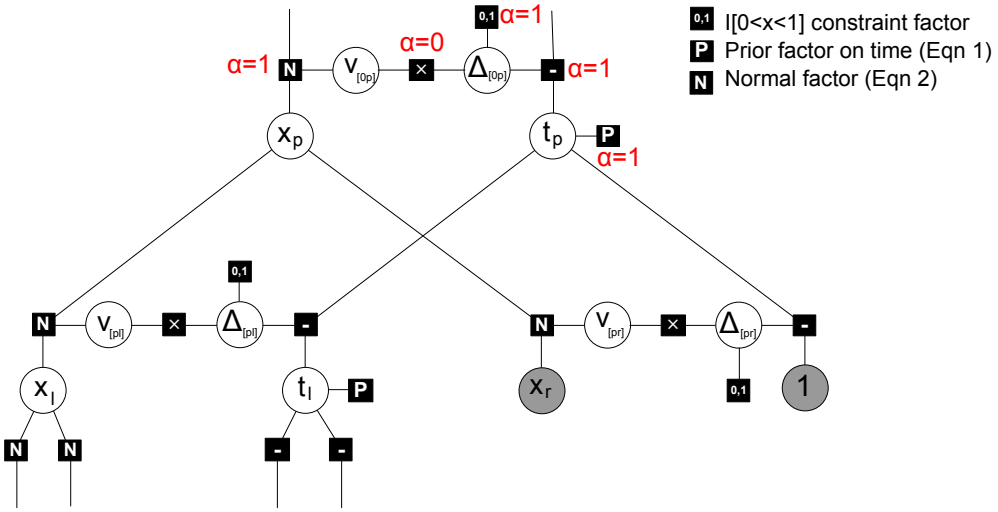


Figure 7.1: A subsection of the factor graph for a tree. The left node represents an internal node and is connected to two child nodes (not depicted). The right node is a leaf node hence its location and divergence time are observed (denoted in gray). The choice of α for the factors is shown for the parent node. The hyperparameters σ^2 and c which are connected to the $\times$ (multiplication) factor and time prior P respectively are not shown.

$\{x, t, \Delta, v\}$ is the set of latent variables for all nodes. Here

$$D_\alpha(p||q) = \frac{1}{\alpha(1-\alpha)} \int 1 - p(x)^\alpha q(x)^{(1-\alpha)} dx, \tag{7.3}$$

is the α-divergence between two (normalized) distributions p and q; and $q^{\sim f}(W) = q(W)/\tilde{f}(W)$ is the cavity distribution: the current variational posterior without the contribution of factor f. Minka [2005] describes how this optimization can be implemented as a message passing algorithm on the factor graph.

We choose which α-divergence to minimize for each factor considering performance and computational tractability. The normal factor, deterministic minus factor, divergence time prior and constraint factor use $\alpha = 1$, which corresponds to Expectation Propagation (see Section 5.5). The multiplication factor and prior on divergence function parameter c use $\alpha = 0$, which corresponds to Variational Bayes/Variational Message Passing (see Sections 5.7—5.9). For the normal factor we use $\alpha = 1$ to attempt to approximate the evidence unbiasedly ($\alpha = 0$ only lower bounds the evidence, see Minka [2005]). For the divergence time prior we

used $\alpha = 1$ (EP), although we could have used Non-conjugate VMP (Chapter 6). For the constraint factors the divergence is infinite for $\alpha = 0$ but analytic for $\alpha = 1$. For the multiplication factor we use $\alpha = 0$ due to the multimodal posterior [Stern *et al.*, 2009]. Since we only use $\alpha = 0$ and 1 our algorithm can also be viewed as a hybrid Expectation Propagation [Minka, 2001b] and Variational Message Passing [Winn & Bishop, 2006]/mean field [Beal & Ghahramani, 2006] algorithm. We use the Infer.NET [Minka *et al.*, 2010] low level library of message updates to calculate the outgoing message from each factor.

7.2.2 Further approximations

We found several approximations to the full message passing solution to be beneficial to the accuracy-speed trade-off of our algorithm.

- The message from the divergence time prior is a Beta distribution. Calculating the true EP message when $q(t)$ is Gaussian would require quadrature, which we found to be less accurate and more computationally expensive than the following: map the incoming Gaussian message to a Beta distribution with the same mean and variance; multiply in the Beta message; then map the outgoing message back to a Gaussian, again by matching moments.

- For practically sized trees (i.e. with 10 or more leaves) we found the message from the variance to the normal factor was typically quite peaked. We found no significant loss in performance in using only the mean of this message when updating the location marginals. In fact, since this removes the need to do any quadrature, we often found the accuracy was improved.

- Similarly for the divergence function parameter c, we simply use the mean of the incoming message, since this was typically quite peaked.

Approximating the model evidence is required to drive the search over tree structures (see Section 7.5). Our evidence calculations follow Minka [2005], to which we defer for details. We use the evidence calculated at each iteration to assess convergence.

7.2.3 Scheduling

Sensible scheduling of the message passing algorithm aided performance. The factor graph consists of two trees: one for the divergence locations, one for the divergences times, with the branch lengths and variances forming cross links between the trees. Belief propagation on a tree is exact: in one sweep up and then down the tree all the marginals are found. Although this is not the case when the divergence times are unknown or KL projection is required at the leaves, it implies that such sweeps will propagate information efficiently throughout the tree, since EP is closely related to BP. To propagate information efficiently throughout this factor graph, our schedule consists of one sweep up and down the tree of locations and tree of times, followed by one sweep back and forth along the cross-links.

Usually one would start with the messages coming in from the prior: for example for the divergence times. Unfortunately in our case these messages are improper, and only result in proper marginals when the constraints that $t_a < t_b$ are enforced through the constraint that the branch length $\Delta_{a \to b}$ must be positive. To alleviate this problem we initially set the message from the prior to spread the divergence times in the correct order, between 0 and 1, then run an iteration of message passing on the tree of divergence times, the constraints $0 < \Delta_{a \to b} < 1$ and $0 < t < 1$ and the prior. This results in proper variance messages into the Normal factors when we sweep over the tree of location times.

7.3 Message passing in EM algorithm

For high dimensional problems we have found that our message passing algorithm over the divergence times can have convergence problems. This can be addressed using damping, or by maximizing over the divergence times rather than trying to marginalize them. In high dimensional problems the divergence times tend to have more peaked posteriors because each dimension provides independent information on when the divergence times should be. Because of this, and because of the increasing evidence contribution from the increasing number of Gaussian factors in the model at higher dimension D, modeling the uncertainty in the divergence times becomes less important. This suggests optimizing the divergence

times in a variational EM type algorithm.

In the E-step, we use message passing to integrate over the locations and hyperparameters. In the M-step we maximize the variational lower bound on the marginal likelihood with respect to the divergence times. While directly maximizing the marginal likelihood itself is in principle possible calculating the gradient with respect to the divergence times is intractable since all the terms are all coupled through the tree of locations.

One simple approach is to optimize each divergence time in turn (e.g. using golden section search), performing a co-ordinate ascent. However, we found jointly optimizing the divergence times using LBFGS [Liu & Nocedal, 1989] to be more computationally efficient. Since the divergence times must lie within $[0, 1]$ we use the reparameterization $s_i = \log[t_i/(1-t_i)]$ to extend the domain to the whole space, which we find improves empirical performance. From Equations 2.3 and 4.5 the lower bound on the log evidence with respect to an individual divergence time t_i is

$$(\langle c \rangle J^{\theta,\alpha}_{\mathbf{n}^i} - 1)\log(1-t_i) - \frac{D}{2}\log(t_i - t_p) - \langle \frac{1}{\sigma^2} \rangle \frac{b_{[pi]}}{t_i - t_p}$$
$$a = \frac{D}{2}, \qquad b_{[pi]} = \frac{1}{2}\sum_{d=1}^{D} \mathbb{E}[(x_{di} - x_{dp})^2] \tag{7.4}$$

where x_{di} is the location of node i in dimension d, and p is the parent of node i. The full lower bound is the sum of such terms over all nodes. The expectation required for $b_{[pi]}$ is readily calculated from the marginals of the locations after message passing. Differentiating to obtain the gradient with respect to t_i is straightforward so we omit the details. The chain rule is used to obtain the gradient with respect to the reparameterisation s_i, i.e. $\frac{\partial \cdot}{\partial s_i} = \frac{\partial t_i}{\partial s_i}\frac{\partial \cdot}{\partial t_i} = t_i(1-t_i)\frac{\partial \cdot}{\partial t_i}$ Although this is a constrained optimization problem (branch lengths cannot be negative) it is not necessary to use the log barrier method because the $1/(t_i - t_p)$ terms in the objective implicitly enforce the constraints.

7.4 Hyperparameter learning.

The DDT has two hyperparameters: the variance of the underlying Brownian motion σ^2 and the divergence function parameter c, which controls the smoothness of the data. For the full message passing framework, the overall variance σ^2 is given a Gaussian prior and variational posterior and learnt using the multiplication factor with $\alpha = 0$, corresponding to the mean field divergence measure. For the EM algorithm we use a Gamma prior and variational posterior for $1/\sigma^2$. The message from each segment $[ab]$ to $1/\sigma^2$ is then

$$m_{[ab]\to 1/\sigma^2} = \mathrm{G}\left(\frac{D}{2}+1, \frac{b_{[pi]}}{2(t_b - t_a)}\right), \tag{7.5}$$

where $G(\alpha, \beta)$ is a Gamma distribution with shape α and rate β, and $b_{[pi]}$ is the same as for Equation 7.4. The smoothness c is given a Gamma prior, and sent the following VMP message from every internal node i:

$$\begin{aligned} \langle \log p(t_i, c)\rangle &= \log c + (cJ_{\mathbf{n}^i} - 1)\langle \log(1-t_i)\rangle \\ \Rightarrow m_{i\to c} &= \mathrm{G}\left(c; 2, -J_{\mathbf{n}^i}\langle \log\left[1-t_i\right]\rangle\right) \end{aligned} \tag{7.6}$$

The term $\langle \log(1-t_i)\rangle$ is deterministic for the EM algorithm and is easily approximated under the full message passing algorithm by mapping the Gaussian $q(t_i)$ to a $\mathrm{Beta}(t_i; \alpha, \beta)$ distribution with the same mean and variance, and noting that $\langle \log(1-t_i)\rangle = \phi(\beta) - \phi(\alpha+\beta)$ where $\phi(.)$ is the digamma function.

The PYDT has two additional hyperparameters, the concentration parameter θ and discount parameter α. We optimise the variational lower bound with respect to these parameters using golden section search on the terms in Equations 4.23 and 4.25.

7.5 Search over tree structures

Our resulting message passing algorithm approximates the marginal likelihood for a fixed tree structure, $p(y|\mathcal{T})p(\mathcal{T})$ (we include the factor for the probability of the tree structure itself). Ideally we would now sum the marginal likelihood

over all possible tree structures $\mathcal{T}$ over N leaf nodes. Unfortunately, there are exponentially many such tree structures so that enumeration of all tree structures for even a modest number of leaves is not feasible. Instead we maintain a list of K-best trees (typically $K = 10$) which we find gives good empirical performance on a density estimation problem.

We search the space of tree structures by detaching and re-attaching subtrees, which may in fact be single leaf nodes. Central to the efficiency of our method is keeping the messages (and divergence times) for both the main tree and detached subtree so that small changes to the structure only require a few iterations of inference to reconverge.

We experimented with several heuristics for choosing which subtree to detach but none significantly outperformed choosing a subtree at random. However, we greatly improve upon attaching at random. We calculate the local contribution to the evidence that would be made by attaching the root of the subtree to the midpoint of each possible branch (and to every possible branch point for the PYDT). We then run inference on the L-best attachments ($L = 3$ worked well, see Figure 7.4). What we do next depends on whether we are in the initialisation tree building phase or the tree search phase, as described in the following.

7.5.1 Sequential tree building

To build an initial tree structure we sequentially process the N leaves. We start with a single internal node with the first two leaves as children. We run inference to convergence on this tree. Given a current tree incorporating the first $n - 1$ leaves, we use the local evidence calculation described above to propose L possible branches (or branch points for the PYDT) at which we could attach leaf n. We run inference to convergence on the L resulting trees and choose the one with the best evidence for the next iteration.

7.5.2 Tree search

Starting from a random tree or a tree built using the sequential tree building algorithm, we can use tree search to improve the list of K-best trees. We detach a subtree at random from the current best tree, and use the local evidence calcu-

lation to propose L positions at which to re-attach the detached subtree. We run message passing/EM to convergence in the resulting trees, add these to the list of trees and keep only the K best trees in terms of model evidence for the next iteration.

7.6 Predictive distribution

To calculate the predictive distribution for a specific tree we compute the distribution for a new data point conditioned on the posterior location and divergence time marginals. Firstly, we calculate the probability of diverging from each branch (and branch point in the PYDT case) according to the data generating process. Secondly we draw several (typically three) samples of when divergence from each branch occurs. Finally we calculate the Gaussian at the leaves resulting from Brownian motion starting at the sampled divergence time and location up to $t = 1$. This results in a predictive distribution represented as a weighted mixture of Gaussians. Finally we average the density from the K-best trees found by the algorithm.

7.6.1 Likelihood models

Connecting our DDT module to different likelihood models is straightforward. We demonstrate a Gaussian observation model for multivariate continuous data and a probit model for binary vectors. Both factors use $\alpha = 1$, corresponding to EP [Minka, 2001b].

7.6.2 Computational cost

One iteration of message passing costs $O(ND)$. Message passing therefore has complexity $O(mND)$ where m is the number of iterations. m is kept small by maintaining messages when changing the structure. The E-step of EM costs $O(nND)$, where n is the number of iterations. With fixed σ and Gaussian observations the E-step is simply belief propagation so $n = 1$. Usually $n < m$ due to not updating divergence times. The order k LBFGS in the M-step costs $O(ksN)$,

where s, the number of iterations, is typically small due to good initialisation. So an iteration costs $O(mND + skN)$.

In contrast to our message passing solution, MCMC requires sampling the divergence times using slice sampling: there are $N-1$ divergence times to sample, each of which requires an $O(ND)$ belief propagation sweep, giving total cost $O(SN^2D)$ where S is the number of slice sampling iterations used before sampling the structure itself. The different scaling with N, combined with our more directed, greedy search confers our improvement in performance.

7.7 Experiments

We tested our algorithms on both synthetic and real world data to assess computational and statistical performance both of variants of our algorithms and competing methods. Where computation times are given these were on a system running Windows 7 Professional with a Intel Core i7 2.67GHz quadcore processor and 4GB RAM. Note that the results presented in this chapter are for the DDT, results for the PYDT are shown in Chapter 4.

7.7.1 Toy 2D fractal dataset.

Our first experiment is on a simple two dimensional toy example with clear hierarchical (fractal) structure shown in Figure 7.2, with $N = 63$ datapoints. Using the message passing in EM algorithm with sequential tree building followed by 100 iterations of tree search we obtain the tree shown in Figure 7.2 in 7 seconds. The algorithm has recovered the underlying hierarchical structure of data apart from the occasional mistake close to the leaves where it is not clear what the optimal solution should be anyway.

7.7.2 Data from the prior ($D = 5, N = 200$)

We use a dataset sampled from the prior with $\sigma^2 = 1, c = 1$, shown in Figure 7.3, to assess the different approaches to tree building and search discussed in Section 7.5. The results are shown in Figure 7.4. Eight repeats of each method were performed using different random seeds. The slowest method starts with a

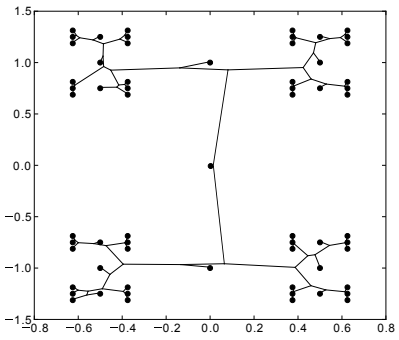


Figure 7.2: Toy 2D fractal dataset (N=63) showing learnt tree structure.

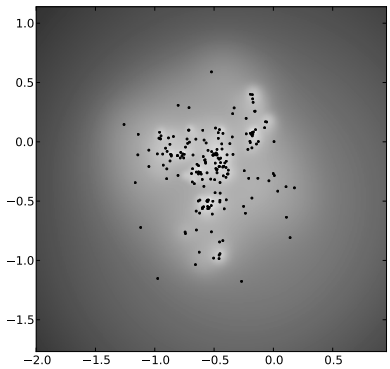


Figure 7.3: First two dimensions of the synthetic dataset from the prior with $D = 5, N = 200, \sigma^2 = 1, c = 1$. Lighter background denotes higher probability density.

random tree and tries randomly re-attaching subtrees ("*search random*"). Preferentially proposing re-attaching subtrees at the best three positions significantly improves performance ("*search greedy*"). Sequential tree building is very fast (5-7 seconds), and can be followed by search where we only move leaves ("*build+search leaves*") or better, subtrees ("*build+search subtrees*"). The spread in initial log evidences for the sequential tree build methods is due to different permutations of the data used for the sequential processing. This variation suggests tree building using several random permutations of the data (potentially in parallel) and then choosing the best resulting tree.

7.7.3 Macaque skull measurements ($N = 200, D = 10$)

We use the macaque skull measurement data of Adams *et al.* [2008] to assess our algorithm's performance as a density model. Following Adams *et al.* [2008] we split the 10 dimensional data into 200 training points and 28 test points. The data consists of three technical repeats which we simply model as three separate datasets. We compare to the infinite mixture of Gaussians (iMOG MCMC) and DDT MCMC methods implemented in Radford Neal's Flexible Bayesian Mod-

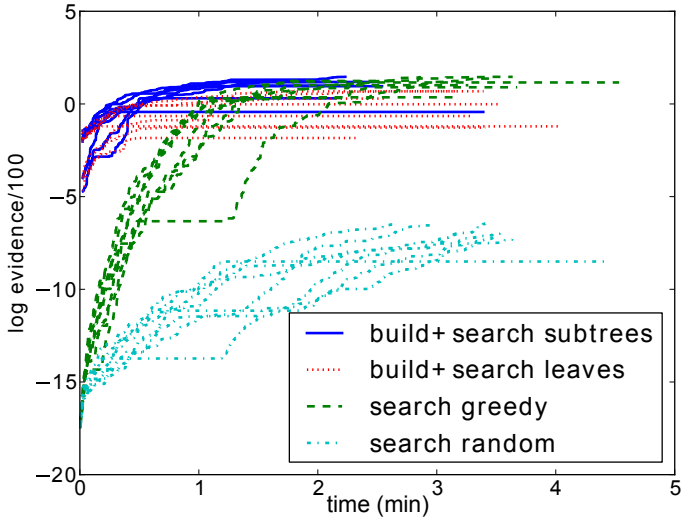


Figure 7.4: Performance of different tree building/search methods on the synthetic dataset.

eling software[1]. As a baseline we use a kernel density estimate with bandwidth selected using the `npudens` R package. The results are shown in Figure 7.5. The EM version of our algorithm is able to find a good solution in just a few tens of seconds, but is eventually beaten on predictive performance by the MCMC solution. The full message passing solution lies between the MCMC and EM solutions in terms of speed, and only outperforms the EM solution on the first of the three repeats. The DDT based algorithms typically outperform the infinite mixture of Gaussians, with the exception of the second dataset.

7.7.4 Gene expression dataset ($N = 2000, D = 171$)

We apply the EM algorithm with sequential tree building and 200 iterations of tree search to hierarchical clustering of the 2000 most variable genes from Yu & et al. Landsittel [2004]. We calculate predictive log likelihoods on four splits into 1800 training and 200 test genes. The results are shown in Table 7.1. The EM algorithm for the DDT has comparable statistical performance to the MCMC solution whilst being an order of magnitude faster. Both implementations signif-

[1] http://www.cs.toronto.edu/~radford/

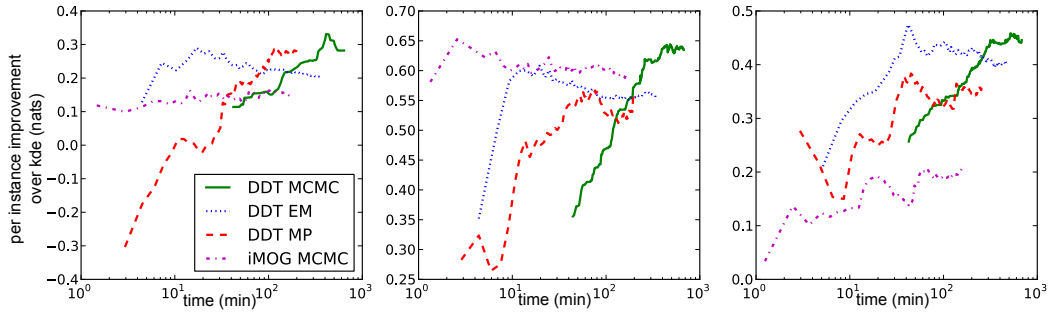


Figure 7.5: Per instance test set performance on the macaque skull measurement data Adams *et al.* [2008]. The three plots arise from using the three technical replicates as separate datasets.

	iMOG	DDT EM	DDT MCMC
Score	-1.00 ± 0.04	-0.91 ± 0.02	-0.88 ± 0.03
Time	37min	48min	18hours

Table 7.1: Results on a gene expression dataset [Yu & et al. Landsittel, 2004]. *Score* is the per test point, per dimension log predictive likelihood. *Time* is the average computation time on the system described in Section 7.7.

icantly outperform iMOG in terms of predictive performance. DDT MCMC was run for 100 iterations, where one iteration involves sampling the position of every subtree, and the score computed averaging over the last 50 samples. Running DDT MCMC for 5 iterations takes 54min (comparable to the time for EM) and gives a score of -0.98 ± 0.04, worse than DDT EM.

7.7.5 Animal species

To demonstrate the use of an alternative observation model we use a probit observation model in each dimension to model 102-dimensional binary feature vectors relating to attributes (e.g. being warm-blooded, having two legs) of 33 animal species [Tenenbaum & Kemp, 2008]. The tree structure we find, shown in Figure 7.6, is intuitive, with subtrees corresponding to land mammals, aquatic mammals, reptiles, birds, and insects (shown by colour coding). The tree obtained is broadly consistent with that found by Bayesian hierarchical clustering [Heller & Ghahramani, 2005]. BHC is faster than our method, taking around 3s vs 30s

for DDT EM (the difference would be less for continuous data where we do not require EP). However, BHC is not a generative model and so cannot be coherently incorporated into larger models.

7.8 Conclusion

Our approximate inference scheme, combining message passing and greedy tree search, is a computationally attractive alternative to MCMC for DDT and PYDT models. We have demonstrated the strength of our method for modeling observed continuous and binary data at the leaves, and hope that by demonstrating that efficient inference is possible we will encourage the community to use this elegant prior over hierarchies.

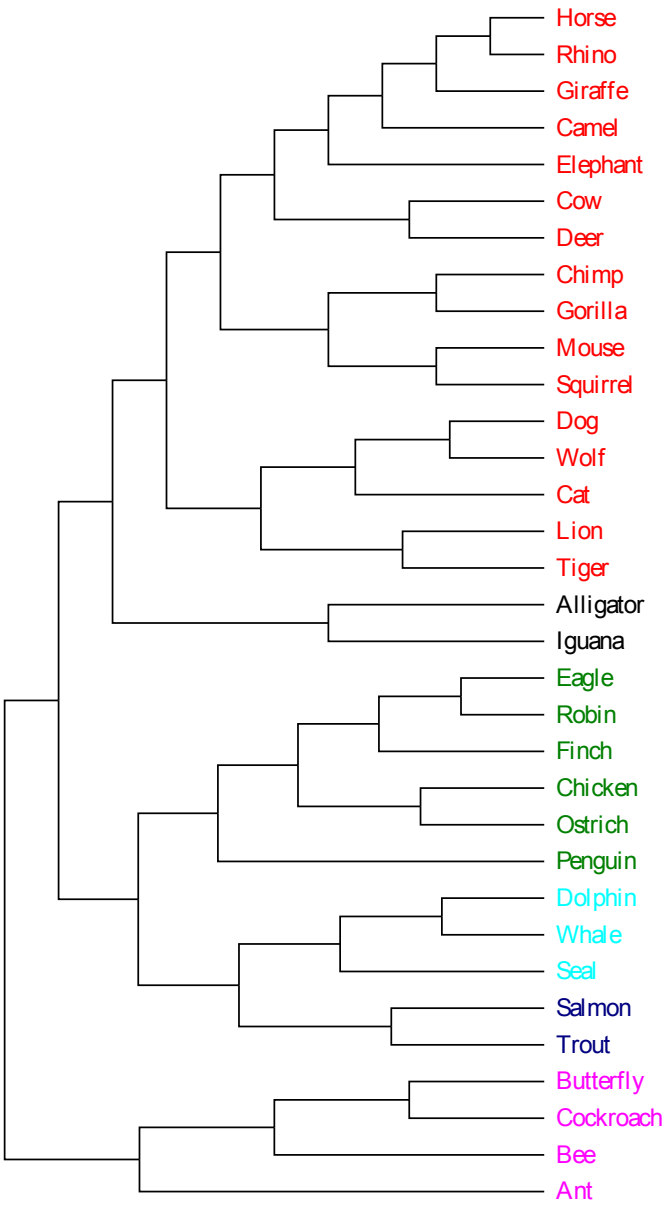


Figure 7.6: DDT structure learnt over animals using 102 binary features with the probit observation model. The hierarchy is intuitively reasonable, which subtrees for the different animal kingdoms. Contrast this to Figure 4.11 which shows the solution under the PYDT.

Chapter 8

Conclusion and Future Work

This thesis aims to contribute towards allowing more expressive probabilistic models to be defined and handled computationally. To this end two new Bayesian non-parametric models were introduced: non-parametric sparse factor analysis (Chapter 3) and the Pitman Yor diffusion tree (Chapter 4). Both these models uncover latent structure in observed data and provide state of the art predictive performance. Chapter 5 reviewed efficient approximate message passing algorithms for Bayesian inference, and a novel extension of variational message passing in particular was presented in Chapter 6. Finally in Chapter 7 message passing algorithms were used to enable faster inference in the Dirichlet diffusion tree and Pitman Yor diffusion tree.

8.1 Future work

Here we discuss some potential future research directions.

8.1.1 Non-parametric sparse factor analysis

Prior knowledge. In a domain like biology there is wealth of semantic prior knowledge: known transcription factors, protein interactions, signalling pathways and so on. Incorporating all this information is a daunting challenge both from a statistical and computational perspective. The non-parametric sparse factor analysis could represent a component of a system designed to integrate and ex-

trapolate from these diverse sources of data. One can imagine incorporating prior knowledge in a soft fashion: allowing observed data to suggest corrections to "known" signalling pathways, or hypothesise previously unknown molecular interactions which could then be tested experimentally.

Efficient inference. While considerable effort has been invested in both more efficient MCMC [Doshi-Velez & Ghahramani, 2009a; Doshi-Velez *et al.*, 2009a] and variational methods [Ding *et al.*, 2010; Doshi-Velez *et al.*, 2009b] for the most simple IBP based models, little attempt has been made to extend these methods to more complex models such as NSFA, although this should be relatively straightforward.

8.1.2 Pitman Yor diffusion trees

Different diffusion processes. We focused on the specific case of Brownian diffusion on the tree structure, which is particularly attractive since belief propagation can be used to very efficiently evaluate the marginal likelihood. Combined with different link functions at the leaves this need not be particularly restrictive: Chapter 7 demonstrated using a probit observation model to allow binary data to be modelled, and appropriate link functions could be used analogously to generalised linear models [McCullagh & Nelder, 1989]. Alternatively the diffusion process itself can be changed, for example random mutation processes are commonly used in modeling genetic variation [Huelsenbeck *et al.*, 2001], or a hierarchical Dirichlet process [Teh *et al.*, 2006] as proposed in Adams *et al.* [2010]. Care must be taken to understand the properties of these different diffusion processes [Steinhardt & Ghahramani, 2012]. Another option would be the auxiliary variable method of [Pitt & Walker, 2005]. For the subset of these models with low rank continuous spaces, exact inference is tractable [Smith *et al.*, 2012].

Latent variable models. It would be particularly interesting to investigate the use of the PYDT to learn hierarchical structure over latent variables such as Hidden Markov Models, specifically in part of speech tagging [Kupiec, 1992] where a hierarchy over the latent states aids interpretability, and Latent Dirichlet

Allocation, where it is intuitive that topics might be hierarchically clustered [Blei *et al.*, 2004].

Analogously to the Hierarchical DP construction of the infinite HMM, it should be possible to construct a hierarchical diffusion process, to allow for example hierarchical states in an HMM.

Dependent tree structures. In Teh *et al.* [2011] a time dependent partition valued stochastic process is constructed using coagulation and fragmentation process and used to model genetic variation. While providing excellent empirical performance, this construction assumes a flat partitioning of the data where we know the true generating process to be hierarchical. An open question is how to construct covariate dependent distributions over tree structures.

Improved MCMC methods. Much of the computational benefit of the greedy message passing algorithm for the DDT and PYDT presented in Chapter 7 comes from being able to propose sensible reattachment positions for the detached subtrees. In principle such moves should be possible as part of a Metropolis Hastings scheme, but care must of course be taken to maintain detailed balance.

Variational methods for the tree structure. While we used message passing methods to approximate the marginal likelihood for a given tree structure, it would be interesting to investigate learning the tree structure itself using variational methods. Mean field and structured variational inference over the latent variables and tree structure for the model of Williams [2000] was described in [Adams *et al.*, 2000] and [Storkey, 2000] respectively. However, these methods retain the disadvantage of requiring the number of levels and number of nodes per level to be pre-specified. An adaptive variational method able to add nodes/levels as required could be beneficial, but escaping local modes might be challenging.

8.1.3 Message passing

While we focused on deterministic methods such as quadrature and bounds to find the expectations required for Non-conjugate VMP, there has been recent working using Monte Carlo instead [Graves, 2011]. Since the approximate posterior q

typically has a simple exponential family form it is easy to sample from. An interesting question for future research is what convergence guarantees can be given for such an algorithm and how many samples are needed: i.e. how accurate do the gradient approximations need to be?

While NCVMP and EP both allow message passing algorithms to be applied to a wide range of models, neither has clear convergence guarantees, and indeed in general little is known about the convergence properties of such algorithms. As mentioned in Section 5.12 some limited results are available for mixture models [Titterington, 2011; Wang & Titterington, 2006]. Better understanding of when these algorithms converge is clearly important if they are to see widespread use, especially in a package such as Infer.NET which aims to provide "automatic" inference where a user is able to input any generative model and get back a custom message passing algorithm.

References

Adams, N., Storkey, A., Ghahramani, Z. & Williams, C. (2000). MFDTs: Mean field dynamic trees. In *15th International Conference on Pattern Recognition*, IEEE. 153

Adams, R.P. & Stegle, O. (2008). Gaussian process product models for nonparametric nonstationarity. In *28th International Conference on Machine Learning (ICML)*, ACM. 117

Adams, R.P., Murray, I. & MacKay, D. (2008). The Gaussian process density sampler. In *Advances in Neural Information Processing Systems*, vol. 21, MIT Press. 41, 71, 72, 146, 148

Adams, R.P., Ghahramani, Z. & Jordan, M.I. (2010). Tree-structured stick breaking for hierarchical data. In *Advances in Neural Information Processing (NIPS) 23*. 41, 43, 61, 152

Aldous, D.J. (1983). Exchangeability and related topics. In *Ecole d'Ete de Probabilities de Saint-Flour*, vol. XIII, 1–198, Springer. 45, 48

Antoniak, C.E. (1974). Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, **2**, 1152–1174. 3, 41

Archambeau, C. & Bach, F. (2009). Sparse probabilistic projections. In *Advances in Neural Information Processing Systems (NIPS)*, 73–80, MIT Press, Vancouver, Canada. 28

Armagan, A., Dunson, D.B. & Clyde, M. (2011). Generalized beta mixtures of Gaussians. In *Advances in Neural Information Processing Systems (NIPS)*. 13

Attias, H. (2000). A variational Bayesian framework for graphical models. In *Advances in Neural Information Processing Systems (NIPS) 12*, 209–215. 5, 88, 89

Beal, M.J. & Ghahramani, Z. (2006). Variational Bayesian learning of directed graphical models with hidden variables. *Bayesian Analysis*, **1**, 793–832. 5, 88, 139

Beal, M.J., Ghahramani, Z. & Rasmussen, C.E. (2002). The infinite hidden Markov model. *Advances in Neural Information Processing Systems*, **1**, 577–584. 4

Berrou, C., Glavieux, A. & Thitimajshima, P. (1993). Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *IEEE International Conference on Communications*, vol. 2, 1064–1070, IEEE. 5, 79

Bishop, C.M. (1999). Bayesian PCA. In *Advances in Neural Information Processing Systems (NIPS)*, 382–388, MIT Press, Cambridge, MA, USA. 10

Blei, D.M. & Jordan, M.I. (2006). Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 121–144. 6

Blei, D.M. & Lafferty, J.D. (2007). A correlated topic model of science. *Annals of Applied Statistics*. 100, 120

Blei, D.M., Griffiths, T.L., Jordan, M.I. & Tenenbaum, J.B. (2004). Hierarchical topic models and the nested Chinese restaurant process. *Advances in Neural Information Processing Systems (NIPS)*, **16**, 106. 60, 61, 153

Blei, D.M., Griffiths, T.L. & Jordan, M.I. (2010). The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, **57**, 7:1—-7:30. 41, 43, 61, 136

Blundell, C., Teh, Y.W. & Heller, K.A. (2010). Bayesian rose trees. In *Conference on Uncertainty in Artificial Intelligence (UAI)*. 41, 42

Bouchard, G. (2007). Efficient bounds for the softmax function, applications to inference in hybrid models. In *NIPS workshop on approximate inference in hybrid models*, 1–9. 100, 121, 127, 133

Boyd, S. & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. 118

Boyen, X. & Koller, D. (1998). Tractable inference for complex stochastic processes. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, vol. 98. 79

Carvalho, C.M., Polson, N.G. & Scott, J.G. (2009). Handling sparsity via the horseshoe. *Journal of Machine Learning Research*, **136**, 2144–2162. 11, 13

Cemgil, A.T., Fevotte, C. & Godsill, S.J. (2005). Blind separation of sparse sources using variational EM. In *Proc. 13th European Signal Processing Conference (EUSIPCO05)*. 14

Colless, D. (1982). Phylogenetics: the theory and practice of phylogenetic systematics. *Systematic Zoology*, **31**, 100–104. 53

Courville, A.C., Eck, D. & Bengio, Y. (2009). An infinite factor model hierarchy via a noisy-or mechanism. In *Advances in Neural Information Processing Systems (NIPS) 21*, The MIT Press. 39

Cowles, M.K. & Carlin, B.P. (1996). Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 883–904. 6

De Finetti, B. (1931). Funzione caratteristica di un fenomeno aleatorio. 4

Dempster, A.P., Laird, N.M., Rubin, D.B. & Others (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, **39**, 1–38. 97

Ding, N., Qi, Y.A., Xiang, R., Molloy, I. & Li, N. (2010). Nonparametric Bayesian matrix factorization by power-EP. *Journal of Machine Learning Research*, **9**. 88, 96, 152

Doshi-Velez, F. & Ghahramani, Z. (2009a). Accelerated inference for the Indian buffet process. In *Proceedings of the International Conference on Machine Learning*. 21, 152

Doshi-Velez, F. & Ghahramani, Z. (2009b). Correlated non-parametric latent feature models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 143–150. 39

Doshi-Velez, F., Knowles, D.A., Mohamed, S. & Ghahramani, Z. (2009a). Large scale nonparametric Bayesian inference: Data parallelisation in the Indian buffet process. In *Advances in Neural Information Processing Systems (NIPS)*. 38, 152

Doshi-Velez, F., Miller, K.T., Van Gael, J. & Teh, Y.W. (2009b). Variational inference for the Indian buffet process. In *Proceedings of the Intl. Conf. on Artificial Intelligence and Statistics*, vol. 12, 137–144. 152

Duda, R.O., Hart, P.E. & Stork, D.G. (2001). *Pattern Classification*. Wiley-Interscience, 2nd edn. 41, 42

Elidan, G., McGraw, I. & Koller, D. (2006). Residual belief propagation: Informed scheduling for asynchronous message passing. In *22nd Conference on Uncertainty in Artificial Intelligence (UAI)*, vol. 6, 4–6. 97

Escobar, M.D. & West, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, **90**, 577–588. 3

Fevotte, C. & Godsill, S.J. (2006). A Bayesian approach for blind separation of sparse sources. *Audio, Speech, and Language Processing, IEEE Transactions on*, **14**, 2174–2188. 14, 28

Fokoue, E. (2004). Stochastic determination of the intrinsic structure in Bayesian factor analysis. Tech. rep., Statistical and Applied Mathematical Sciences Institute. 13, 14, 28, 29

Frey, B.J. & MacKay, D.J.C. (1998). A revolution: Belief propagation in graphs with cycles. In *Advances in Neural Information Processing Systems (NIPS)*, 479–485. 5, 79

Gallager, R.G. (1963). *Low Density Parity Check Codes*. M.I.T. Press. 5

Geweke, J. (1993). Bayesian treatment of the independent student-t linear model. *Journal of Applied Econometrics*, **8**, 19–40. 11

Geweke, J. (2004). Getting it right. *Journal of the American Statistical Association*, **99**, 799–804. 26

Ghahramani, Z. (2006). Deterministic alternatives to MCMC. In *Stochastic Computation in the Biological Sciences*. 6

Ghahramani, Z. & Beal, M.J. (2001). Propagation algorithms for variational Bayesian learning. In *Advances in Neural Information Processing Systems (NIPS) 13*, 507–513, MIT Press. 14

Ghahramani, Z., Griffiths, T.L. & Sollich, P. (2007). Bayesian nonparametric latent feature models. In *Bayesian Statistics 8*, vol. 8, 1–25, Oxford University Press. 15

Girolami, M. & Rogers, S. (2006). Variational Bayesian multinomial probit regression with Gaussian process priors. *Neural Computation*, **18**, 1790–1817. 128

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 153

Griffiths, T. & Ghahramani, Z. (2005). Infinite latent feature models and the Indian buffet process. Tech. Rep. 1, Gatsby Computational Neuroscience Unit. 15, 16, 18

Griffiths, T.L. & Ghahramani, Z. (2011). The Indian buffet process: An introduction and review. *Journal of Machine Learning Research*, **12**, 1185–1224. 15, 21

Guiver, J. & Snelson, E. (2009). Bayesian inference for Plackett-Luce ranking models. In *26th International Conference on Machine Learning (ICML)*. 88, 96

Guyon, I., Gunn, S., Ben-Hur, A. & Dror, G. (2005). Result analysis of the NIPS 2003 feature selection challenge. *Advances in Neural Information Processing Systems (NIPS)*, **17**, 545–552. 41

Heller, K.A. & Ghahramani, Z. (2005). Bayesian hierarchical clustering. In *22nd International Conference on Machine Learning (ICML)*, 304. 41, 42, 148

Heskes, T. & Zoeter, O. (2002). Expectation propagation for approximate inference in dynamic Bayesian networks. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 216–233. 80, 97

Heskes, T., Opper, M., Wiegerinck, W., Winther, O. & Zoeter, O. (2005). Approximate inference techniques with expectation constraints. *Journal of Statistical Mechanics: Theory and Experiment*, **2005**, P11015. 97

Hewitt, E. & Savage, L.J. (1955). Symmetric measures on Cartesian products. *Transactions of the American Mathematical Society*, **80**, 470–501. 59

Hjort, N.L. (1990). Nonparametric bayes estimators based on beta processes in models for life history data. *The Annals of Statistics*, **18**, 1259–1294. 15

Honkela, A., Tornio, M., Raiko, T. & Karhunen, J. (2007). Natural conjugate gradient in variational inference. In *ICONIP (2)*, vol. 4985 of *Lecture Notes in Computer Science*, 305–314. 100

Honkela, A., Raiko, T., Kuusela, M., Tornio, M. & Karhunen, J. (2010). Approximate Riemannian conjugate gradient learning for fixed-form variational Bayes. *Journal of Machine Learning Research*, **11**, 3235–3268. 100

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, **24**, 417–441. 9

Huelsenbeck, J.P., Ronquist, F. & Others (2001). MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics*, **17**, 754–755. 152

Ishwaran & Rao, J.S. (2003). Spike and slab variable selection: frequentist and Bayesian strategies. *Annals of Statistics*. 11

Ishwaran, H. & Rao, J.S. (2005). Spike and slab gene selection for multigroup microarray data. *Journal of the American Statistical Association*, **100**, 764–780. 11

Jaakkola, T.S. & Jordan, M.I. (1996). A variational approach to Bayesian logistic regression models and their extensions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 100, 118

Jensen, F., Jensen, F.V., Dittmer, S.L. & Others (1994). From influence diagrams to junction trees. In *10th Conference on Uncertainty in Artificial Intelligence (UAI)*. 5, 79

Kao, K.C., Yang, Y.L., Boscolo, R., Sabatti, C., Roychowdhury, V. & Liao, J.C. (2004). Transcriptome-based determination of multiple transcription regulator activities in Escherichia coli by using network component analysis. *Proceedings of the National Academy of Science*, **101**, 641–646. v, viii, 29, 30, 33, 34

Kaufman, G.M. & Press, S.J. (1973). Bayesian factor analysis. Tech. Rep. 662-73, Sloan School of Management, University of Chicago. 28

Khan, M.E., Marlin, B.M., Bouchard, G. & Murphy, K.P. (2010). Variational bounds for mixed-data factor analysis. In *Advances in Neural Information Processing (NIPS) 23*. 100

Kim, H.C.C. & Ghahramani, Z. (2006). Bayesian Gaussian process classification with the EM-EP algorithm. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, 1948–1959, IEEE. 97, 136

Kingman, J.F.C. (1982). The coalescent. *Stochastic processes and their applications*, **13**, 235–248. 43

Kingman, J.F.C. (1993). *Poisson processes*. Oxford University Press, USA. 64

Knowles, D.A. & Ghahramani, Z. (2007). Infinite sparse factor analysis and infinite independent components analysis. In *7th International Conference on Independent Component Analysis and Signal Separation*, vol. 4666, 381–388, Springer. 18, 21

Knowles, D.A. & Ghahramani, Z. (2011a). Nonparametric Bayesian sparse factor models with application to gene expression modeling. *The Annals of Applied Statistics*, **5**, 1534–1552. 18

Knowles, D.A. & Ghahramani, Z. (2011b). Pitman-Yor diffusion trees. *Conference on Uncertainty in Artificial Intelligence (UAI)*. 40

Knowles, D.A. & Minka, T.P. (2011). Non-conjugate variational message passing for multinomial and binary regression. In *Advances in Neural Information Processing Systems (NIPS)*. 99

Knowles, D.A., Parts, L., Glass, D. & Winn, J.M. (2010). Modeling skin and ageing phenotypes using latent variable models in Infer.NET. In *NIPS Workshop: Predictive Models in Personalized Medicine Workshop*. 2

Knowles, D.A., Gael, J.V. & Ghahramani, Z. (2011a). Message passing algorithms for Dirichlet diffusion trees. In *28th International Conference on Machine Learning (ICML)*. 42, 135

Knowles, D.A., Parts, L., Glass, D. & Winn, J.M. (2011b). Inferring a measure of physiological age from multiple ageing related phenotypes. In *NIPS Workshop: From Statistical Genetics to Predictive Models in Personalized Medicine*. 2

Kupiec, J. (1992). Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language*, **6**, 225–242. 152

Lauritzen, S.L. (1992). Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 1098–1108. 79

Lauritzen, S.L. & Spiegelhalter, D.J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 157–224. 5

Liu, D.C. & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, **45**, 503–528. 120, 141

MacKay, D. & Broderick, T. (2007). Probabilities over trees: generalizations of the Dirichlet diffusion tree and Kingman's coalescent. Website. 52

MacKay, D.J.C. (1994). Bayesian nonlinear modeling for the prediction competition. *ASHRAE Transactions*, **100**, 1053–1062. 10

MacKay, D.J.C. (2003). *Information theory, inference, and learning algorithms*. Cambridge Univ Press. 122

Marlin, B.M., Khan, M.E. & Murphy, K.P. (2011). Piecewise bounds for estimating Bernoulli-logistic latent Gaussian models. In *Proceedings of the 28th Annual International Conference on Machine Learning*. 100

Maybeck, P.S. (1979). *Stochastic models, estimation and control*, vol. 1. Academic Press. 79

McCullagh, P. & Nelder, J.A. (1989). Generalized linear models. *Monographs on Statistics and Applied Probability*. 152

McCullagh, P., Pitman, J. & Winkel, M. (2008). Gibbs fragmentation trees. *Bernoulli*, **14**, 988–1002. 42

Meeds, E., Ghahramani, Z., Neal, R. & Roweis, S. (2006). Modeling dyadic data with binary latent factors. In *Advances in Neural Information Processing Systems (NIPS)*, vol. 19. 22, 23

Miller, K.T., Griffiths, T.L. & Jordan, M.I. (2008). The phylogenetic Indian buffet process: A non-exchangeable nonparametric prior for latent features. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, Helsinki. 4

Minka, T. (2005). Divergence measures and message passing. Tech. Rep. MSR-TR-2005-173, Microsoft Research. 5, 87, 101, 135, 137, 138, 139

Minka, T.P. (2000). Automatic choice of dimensionality for PCA. Tech. rep., MIT Media Lab. 10

Minka, T.P. (2001a). The EP energy function and minimization schemes. Website. 97

Minka, T.P. (2001b). Expectation propagation for approximate Bayesian inference. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, vol. 17. 5, 70, 139, 144

Minka, T.P. (2001c). *A family of algorithms for approximate Bayesian inference*. Ph.D. thesis, MIT. 79

Minka, T.P. (2002). Estimating a Gamma distribution. Tech. rep., Microsoft Research, Cambridge UK. 115

Minka, T.P. (2004). Power EP. Tech. rep., Department of Statistics, Carnegie Mellon University, Pittsburgh, PA. 87

Minka, T.P. & Lafferty, J. (2002). Expectation-propagation for the generative aspect model. In *18th Conference on Uncertainty in Artificial Intelligence (UAI)*, 352–359. 97

Minka, T.P., Winn, J.M., Guiver, J.P. & Knowles, D.A. (2010). Infer.NET 2.4. 2, 7, 70, 98, 100, 130, 139

Mohamed, S., Heller, K. & Ghahramani, Z. (2011). Bayesian and L1 approaches to sparse unsupervised learning. *ArXiv e-prints*. 6, 15, 38, 39

Neal, R.M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. *Artificial Intelligence*, **62**, 144. 4

Neal, R.M. (2001). Defining priors for distributions using Dirichlet diffusion trees. Tech. Rep. 0104, Deptart of Statistics, University of Toronto. 40, 45, 59

Neal, R.M. (2003a). Density modeling and clustering using Dirichlet diffusion trees. *Bayesian Statistics*, **7**, 619–629. 41, 42, 44, 46, 62, 69, 135

Neal, R.M. (2003b). Slice sampling. *The Annals of Statistics*, **31**, 705–741. 70

Nickisch, H. & Rasmussen, C.E. (2008). Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, **9**, 2035–2078. 6, 100

Opper, M. & Archambeau, C. (2009). The variational Gaussian approximation revisited. *Neural Computation*, **21**, 786–792. 100

Orbanz, P. & Teh, Y.W. (2010). *Bayesian nonparametric models*. Springer. 1

Parker, R.L. & Rice, J.A. (1985). Discussion of "Some aspects of the spline smoothing approach to nonparametric curve fitting" by B. W. Silverman. *Journal of the Royal Statistical Society, Series B*, **47**, 40–42. 117

Pearl, J. (1982). Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the Second National Conference on Artificial Intelligence*, 133–136, AAAI Press. 5

Pearl, J. & Shafer, G. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann San Mateo, CA. 76

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, **2**, 559–572. 9

Pitman, J. (1999). Coalescents with multiple collisions. *Annals of Probability*, 1870–1902. 44

Pitman, J. & Yor, M. (1997). The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, **25**, 855–900. 48, 65

Pitt, M.K. & Walker, S.G. (2005). Constructing stationary time series models using auxiliary variables with applications. *Journal of the American Statistical Association*, **100**, 554–564. 152

Qi, Y. & Minka, T.P. (2004). Tree-structured approximations by expectation propagation. In *Advances in Neural Information Processing Systems (NIPS) 16*, 193, The MIT Press. 97

Qi, Y., Szummer, M. & Minka, T.P. (2005). Diagram structure recognition by Bayesian conditional random fields. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, 191 – 196 vol. 2. 96

Qi, Y.A. & Jaakkola, T. (2006). Parameter expanded variational Bayesian methods. In *Advances in Neural Information Processing (NIPS) 19*, 1097–1104, MIT Press. 128

Rai, P. & Daumé III, H. (2008). The infinite hierarchical factor regression model. In *Advances in Neural Information Processing Systems (NIPS)*. 4, 19, 31, 41

Raiko, T., Valpola, H., Harva, M. & Karhunen, J. (2007). Building blocks for variational Bayesian learning of latent variable models. *Journal of Machine Learning Research*, **8**, 155–201. 100

Rasmussen, C.E. (2000). The infinite Gaussian mixture model. *Advances in Neural Information Processing Systems*, **12**, 554–560. 3, 41

Rasmussen, C.E. & Williams, C.K.I. (2006). *Gaussian processes for machine learning*. MIT Press. 76, 116

Rattray, M., Stegle, O., Sharp, K. & Winn, J. (2009). Inference algorithms and learning theory for Bayesian sparse factor analysis. In *Journal of Physics: Conference Series*, vol. 197, 12002, IOP Publishing. 6

Rogers, J.S. (1996). Central moments and probability distributions of three measures of phylogenetic tree imbalance. *Systematic biology*, **45**, 99–110. 53

Rowe, D.B. & Press, S.J. (1998). Gibbs sampling and hill climbing in Bayesian factor analysis. Tech. Rep. 255, Department of Statistics, University of California Riverside. 28

Roweis, S. (1998). EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems (NIPS)*, 626–632, MIT Press. 9

Sagitov, S. (1999). The general coalescent with asynchronous mergers of ancestral lines. *Journal of Applied Probability*, **36**, 1116–1125. 44

Saul, L.K. & Jordan, M.I. (1996). Exploiting tractable substructures in intractable networks. *Advances in Neural Information Processing Systems (NIPS)*, 486–492. 97

Saul, L.K. & Jordan, M.I. (1999). A mean field learning algorithm for unsupervised neural networks. *Learning in graphical models*. 100, 119

Smith, C., Wood, F. & Paninski, L. (2012). Low rank continuous-space graphical models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 152

Steinhardt, J. & Ghahramani, Z. (2012). Flexible martingale priors for deep hierarchies. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 43, 61, 62, 65, 152

Stern, D.H., Herbrich, R. & Graepel, T. (2009). Matchbox: large scale online Bayesian recommendations. In *18th International World Wide Web Conference (WWW2009)*, 111–120. 88, 96, 139

Storkey, A.J. (2000). Dynamic trees: A structured variational method giving efficient propagation rules. In *Conference on Uncertainty in Artificial Intelligence (UAI)*. 153

Sudderth, E.B., Ihler, A.T., Freeman, W.T. & Willsky, A.S. (2003). Nonparametric belief propagation. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, **1**, 605–612. 78

Sutton, C. & McCallum, A. (2007). Improved dynamic schedules for belief propagation. In *23rd Conference on Uncertainty in Artificial Intelligence (UAI)*. 97

Teh, Y.W. (2006). A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics*, 992, Association for Computational Linguistics. 48

Teh, Y.W. & Gorur, D. (2009). An efficient sequential Monte Carlo algorithm for coalescent clustering. *Advances in Neural Information Processing Systems (NIPS)*. 136

Teh, Y.W. & Görür, D. (2009). Indian buffet processes with power-law behavior. In *Advances in Neural Information Processing Systems*. 15

Teh, Y.W. & Jordan, M.I. (2009). Hierarchical Bayesian nonparametric models with applications. *Bayesian Nonparameters: Principles and Practice*. 1

Teh, Y.W., Jordan, M.I., Beal, M.J. & Blei, D.M. (2006). Hierarchical dirichlet processes. *Journal of the American Statistical Association*, **101**, 1566–1581. 152

Teh, Y.W., Görür, D. & Ghahramani, Z. (2007). Stick-breaking construction for the Indian buffet process. In *Proceedings of the Intl. Conf. on Artificial Intelligence and Statistics*, vol. 11, 556–563. 15

Teh, Y.W., Daumé III, H. & Roy, D.M. (2008). Bayesian agglomerative clustering with coalescents. *Advances in Neural Information Processing Systems*, **20**. 41, 42, 43

Teh, Y.W., Blundell, C. & Elliott, L.T. (2011). Modelling genetic variations with fragmentation-coagulation processes. In *Advances in Neural Information Processing Systems (NIPS)*. 44, 153

Tenenbaum, J.B. & Kemp, C. (2008). The discovery of structural form. In *Proceedings of the National Academy of Sciences*, vol. 105. 72, 74, 148

Thibaux, R. & Jordan, M.I. (2007). Hierarchical beta processes and the indian buffet process. In *International Conference on Artificial Intelligence and Statistics*, Citeseer. 15

Tibshirani, R. (1994). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, **58**, 267–288. 11

Tipping, M.E. & Bishop, C.M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, **61**, 611–622. 9, 10

Titterington, D.M. (2011). The EM algorithm, variational approximations and expectation propagation for mixtures. *Mixtures*, 1–29. 98, 154

Trefethen, L.N. (2008). Is Gauss quadrature better than Clenshaw-Curtis? *SIAM Review*, **50**, 67–87. 119

Turner, R.E., Berkes, P. & Sahani, M. (2008). Two problems with variational expectation maximisation for time-series models. *Inference and Estimation in Probabilistic Time-Series Models*. 6, 96

Vila Casado, A.I., Griot, M. & Wesel, R.D. (2007). Informed dynamic scheduling for belief-propagation decoding of LDPC codes. In *Communications, 2007. ICC'07. IEEE International Conference on*, 932–937. 97

Wainwright, M.J. & Jordan, M.I. (2003). *Graphical models, exponential families, and variational inference*, vol. 649. Now Publishers Inc. Hanover, MA, USA. 14

Wainwright, M.J., Jaakkola, T.S. & Willsky, A.S. (2003). Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudomoment matching. In *Workshop on Artificial Intelligence and Statistics*, vol. 21. 97

Wald, A. (1949). Note on the consistency of the maximum likelihood estimate. *The Annals of Mathematical Statistics*, **20**, 595–601. 98

Walker, S. & Hjort, N.L. (2001). On Bayesian consistency. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**, 811–821. 98

Wand, M.P., Ormerod, J.T., Padoan, S.A. & Fruhwirth, R. (2010). Variational Bayes for elaborate distributions. In *Workshop on Recent Advances in Bayesian Computation*. 100

Wang, B. & Titterington, D.M. (2006). Convergence properties of a general algorithm for calculating variational Bayesian estimates for a normal mixture model. *Bayesian Analysis*, **1**, 625–650. 84, 98, 154

Wang, C. & Blei, D.M. (2009). Variational inference for the nested Chinese restaurant process. *Advances in Neural Information Processing Systems (NIPS)*. 136

Welling, M., Minka, T.P. & Teh, Y.W. (2005). Structured region graphs: Morphing EP into GBP. In *21st Conference on Uncertainty in Artificial Intelligence (UAI)*, 609–616. 97

West, M., Chang, J., Lucas, J., Nevins, J.R., Wang, Q. & Carvalho, C. (2007). High-dimensional sparse factor modelling: Applications in gene expression genomics. Tech. rep., ISDS, Duke University. v, 14, 18, 28, 34

Wiegerinck, W. & Heskes, T. (2002). Fractional belief propagation. In *Advances in Neural Information Processing Systems 15*, 438–445, MIT Press. 87

Williams, C. (2000). A MCMC approach to hierarchical mixture modelling. *Advances in Neural Information Processing Systems*, **13**. 41, 43, 153

Winn, J. & Bishop, C.M. (2006). Variational message passing. *Journal of Machine Learning Research*, **6**, 661. 5, 89, 139

Witten, D.M., Tibshirani, R. & Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, **10**, 515–534. 14

Yedidia, J.S., Freeman, W.T. & Weiss, Y. (2003). Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, **8**, 236–239. 5, 79

Yedidia, J.S., Freeman, W.T. & Weiss, Y. (2004). Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, **51**, 2282–2312. 97

Young, G. (1941). Maximum likelihood estimation and factor analysis. *Psychometrika*, **6**, 49–53. 9

Yu, Y.P. & et al. Landsittel, D. (2004). Gene expression alterations in prostate cancer predicting tumor aggression and preceding development of malignancy. *Journal of Clinical Oncology*, **22**, 2790–2799. v, 37, 147, 148

Yuille, A.L. (2002). CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural computation*, **14**, 1691–1722. 79

Zou, H., Hastie, T. & Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, **15**, 265–286. 14, 28, 31