Serial and Parallel Inference in Sparse Nonparametric Latent Factor Models applied to Gene Expression Modeling

First Year Report

David Knowles
Supervisor: Professor Zoubin Ghahramani
Cambridge University Engineering Department

October 27, 2010

Abstract

This report presents the core research I have conducted this year. Chapter 1 describes the background literature to the research. The probabilistic formulation Principal Components Analysis and Factor Analysis are introduced. Previous work on introducing sparsity in these models and inferring the latent dimensionality is explored. Chapter 2 presents my main individual project, extending work from my Part IIB project, developing nonparametric sparse Factor Analysis and comparing its performance modeling gene expression data to related models. Chapter 3 presents a collaborative project, undertaken with Finale Doshi and Shakir Mohamed, where we explore techniques for parallelising Bayesian inference in Indian Buffet Process models. Finally, Chapter 4 presents my plan for the remaining duration of the PhD.

Chapter 1

Literature Review

The central dogma of biology is that DNA is transcribed into "messenger" RNA, and RNA is translated into protein which performs the work of the cell. Of course, the reality is a little more complicated than that (RNA itself can have a functional role for example) but its a good approximation. Gene expression microarrays measure how much messenger RNA is present for each gene, which is a reasonable proxy for how much a gene is being transcribed and translated. This provides a rich source of information on the state of cells within a tissue. A gene expression microarray has thousands of probes - sequences of RNA complementary to the mRNA (messanger RNA, a intermediate product of gene transcription) of particular genes.

A microarray typically might have 10000 probes, making analysing this data a very high dimensional problem. We will use a family of linear models. We hypothesise that each sample is a linear combination of "gene signatures", perhaps representing underlying biological processes (such as the activity of a particular transcription factor or the phase of the cell cycle) or confounding experimental factors (such as the age of the patient). We will show that we can infer both these gene signatures and their activation level for each sample from data.

Principal Components Analysis (PCA), Factor Analysis (FA) and Independent Components Analysis (ICA) are models which explain observed data, $\mathbf{y}_n \in \mathbb{R}^D$, in terms of a linear superposition of independent hidden factors, $\mathbf{x}_n \in \mathbb{R}^K$. In our setting of gene expression modelling the hidden factors correspond to underlying biological processes or confounding experimental factors. The number of probes is D, the number of samples is N and the number of underlying gene signatures (latent factors) is K. The model

is:

$$\mathbf{y}_n = \mathbf{G}\mathbf{x}_n + \boldsymbol{\epsilon}_n \tag{1.1}$$

where $\mathbf{G}$ is the factor loading matrix and $\boldsymbol{\epsilon}_n$ is a noise vector, usually assumed to be Gaussian. Factor Analysis, and the closely related Principal Components Analysis (PCA) are become fundamental data analysis tools, used in data compression, image processing, ecology, genetics, portfolio management, and even time series data.

1.1 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 $\hat{\mathbf{y}}_n = \mathbf{G}\mathbf{x}_n + \bar{\mathbf{y}}$.

Bishop and Tipping, and Roweis simultaneously noted the interpretation of Principal Components Analysis as Maximum Likelihood estimation in an appropriate probabilistic model [Tipping and Bishop, 1999; Roweis, 1998]. 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.

1.2 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, usually standard normal in each element, and marginalised out. The latent factors are usually considered as random variables, and the mixing matrix as a parameter to estimate.

1.3 Inferring the latent dimensionality

Bishop [1999] extends the probabilistic PCA formulation of Tipping and 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 of $N(0, \alpha_k^{-1}\mathbf{I})$. 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-style local Gaussian 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 of the model evidence. Laplace approximation proceeds by fitting a Gaussian distribution to the posterior mode by matching the second derivatives of the 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.

1.4 Shrinkage

Much recent work in regression models has concentrated on the concept of shrinkage. The idea is that by shrinking the estimates of the regression coefficients towards zero, one can significantly reduce the variance of an estimator at only a small cost in slightly increased bias. The classic example is the Stein estimator for the mean of multivariate Gaussian, which has lower expected loss (measured as the Euclidean distance between the true and estimated values of the mean), than the MLE for any value of θ [James and Stein, 1961]. From a frequentist perspective this appears completely unintuitive, but from a Bayesian perspective it appears much more reasonable. Assume we observe vector X drawn from a multivariate normal of dimension p, with mean θ and identity covariance matrix. The MLE of θ is then just X, but the Stein estimator

$$\theta_s = \left(1 - \frac{p-2}{||X||^2}\right) X$$

The fact that this estimator performs better than the ML is termed shrinkage, because the estimator is shrunk towards 0. The Bayesian approach would be to put a Gaussian prior on θ, so

$$\theta \sim N_p(\theta; 0, \lambda^{-1} I)$$

where λ is a precision (inverse variance). In a fully Bayesian framework a Gamma prior would be put on λ, resulting in a student-t marginal prior on θ. Unfortunately one would then have to resort to sampling to infer the posterior mean of θ since the product of a Gaussian and a student-t distribution cannot be integrated analytically. However, we can optimise λ. Assuming λ is known, then the posterior of θ is

$$P(\theta|X, \lambda) \propto P(X|\theta)P(\theta|\lambda) = N_p(\theta; (1+\lambda)^{-1}X, (1+\lambda)^{-1})$$

Thus the expected value of θ is

$$E(\theta|X) = (1+\lambda)^{-1}X$$

To find the MLE of λ first integrate out θ:

$$P(X|\lambda) = \int P(X|\theta)P(\theta|\lambda)d\theta$$

$$= N_p(X; 0, \lambda_x^{-1}I)$$

where

$$\lambda_x = \frac{\lambda}{1+\lambda}$$

An unbiased estimate of λ_x is given by

$$\lambda_x^{ML} = \frac{||X||^2}{p-1}$$

Substituting for λ and rearranging gives

$$\lambda^{ML} = \left(\frac{||X||^2}{p-1} - 1\right)^{-1}$$

Substituting into the expression for $E(\theta|X)$ above and rearranging gives

$$E(\theta|X) = \left(1 - \frac{p-1}{||X||^2}\right) X$$

which is very close to the Stein estimate. Some different choice of prior on λ would result in a MAP estimate which would give the $p-2$ term of the Stein estimator. The Stein estimator, which has unintuitively desirable properties in a frequentist framework, is intuitively a sensible estimator in a Bayesian framework. This motivates using sparsity in a Bayesian context.

1.5 Sparsity

Shrinkage is closely related to the concept of sparsity, the idea that only some small proportion of coefficients should be non-zero. There are three main advantages to "sparse" models:

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

2. *Intuitive.* 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 incorporating "hard" or "soft" sparsity. Hard sparsity means that coefficients have finite mass on zero, the main example being so-called "slab-and-spike" models where the coefficient prior is a mixture between a continuous distribution and a delta-spike at 0 [Ishwaran and Rao, 2003, 2005]. Soft sparsity means that the coefficient priors have heavy tails, and so a prior are likely to have either very small (but non-zero) or large values. Examples include the Laplace distributon corresponding to LASSO regression [Tibshirani, 1994], the Student-t distribution [Geweke, 1993], or the Horseshoe prior [Carvalho et al., 2009]. The Student-t distributed can be efficiently modeled as a scale mixture of Gaussians. The inverse variance (precision) parameter of the Gaussian is given a Gamma prior. Figure 1.1 shows the density of two independent student-t variables with degrees of freedom $a = 0.05$. Note how 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 only 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

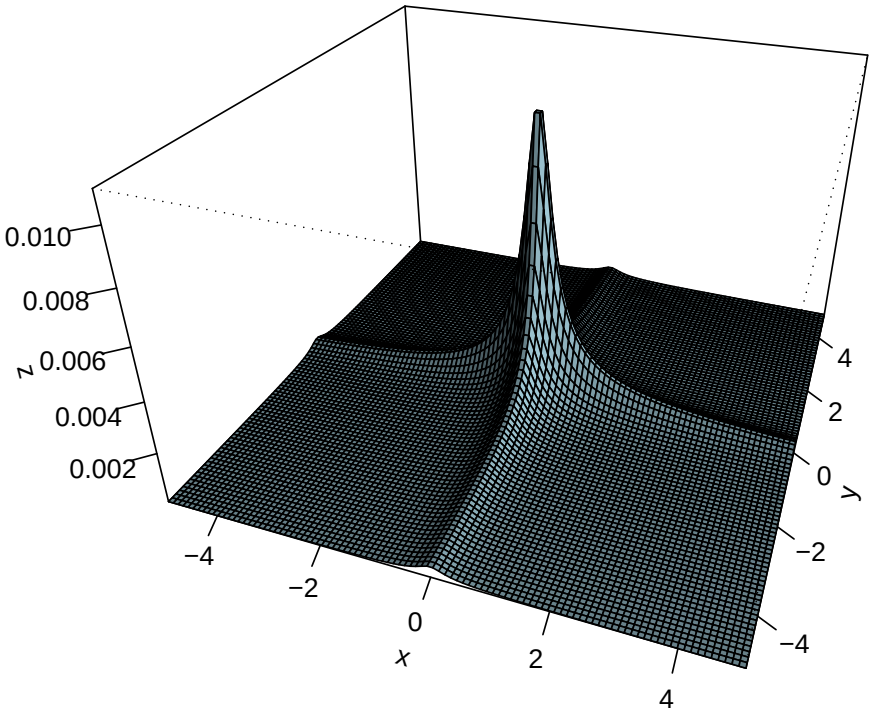


Figure 1.1: The density of two independent Student-t variables with 0.05 degrees of freedom.

$$T(x;a,b) = \frac{\Gamma(\frac{a+1}{2})}{b\sqrt{a\pi}\Gamma(a/2)}\left[1+\frac{1}{a}(x/b)^2\right]^{-\frac{a+1}{2}} \tag{1.2}$$

This is achieved as a scale mixture by setting:

$$x \sim N(x;0,\lambda^{-1}) \tag{1.3}$$

$$\lambda \sim G(\lambda;\frac{a}{2},\frac{2}{ab^2}) \tag{1.4}$$

The horseshoe prior is another scale mixture of normal prior which has recently been proposed [Carvalho et al., 2009]. The horseshoe prior is defined by:

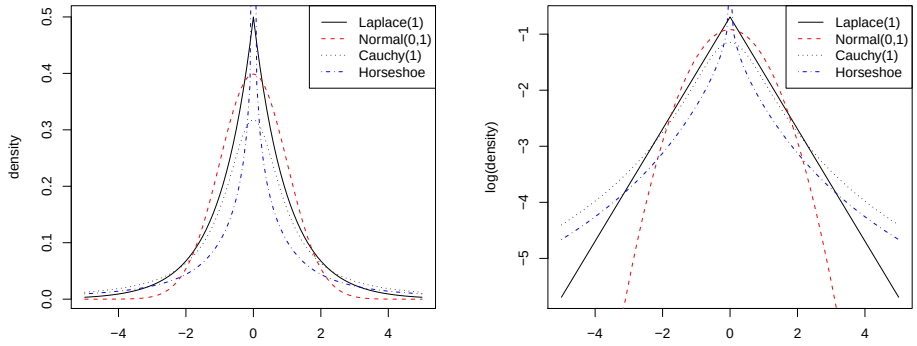
$$x|s \sim N(0,s^2) \tag{1.5}$$

$$s \sim C^+(0,1) \tag{1.6}$$

where the standard deviation s has a standard half-Cauchy distribution, $C^+(0,1)$. There is no closed form for the marginal density, but is approximately $\log(1+2/x^2)$. The horseshoe prior has infinite density at $x=0$, but its tails behave like the Cauchy distribution (the Cauchy distribution is the special case of the Student-t distribution with degrees of freedom $a=1$. The normal, Laplace, Cauchy, and horseshoe priors are shown in Figure 1.2. 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.

1.6 Sparse regression models

Ridge regression, which involves giving an L2-norm penalty (i.e. sum of squares) to the size of the regression coefficients, corresponds to maximum a posterior (MAP) estimation where the coefficients are given a zero mean Gaussian prior. The ratio between the prior variance of the coefficients and the noise variance corresponds to the regularisation constant in ridge regression. Ridge regression however only shrinks the coefficients, it does not



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

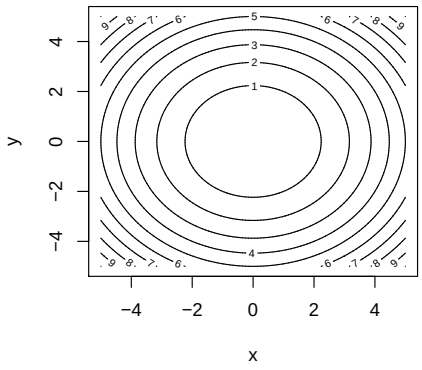
Figure 1.2: Comparing the normal, Laplace, Cauchy and horseshoe priors.

result in "zero-forcing", which may also be desirable. This is the advantage of LASSO regression [Tibshirani, 1994], where an L1 penalty (i.e. sum of absolute values) is used on the coefficients. This results in zero forcing because contours of the penalty function have "corners" on the coordinate axes, as shown in Figure 1.3.

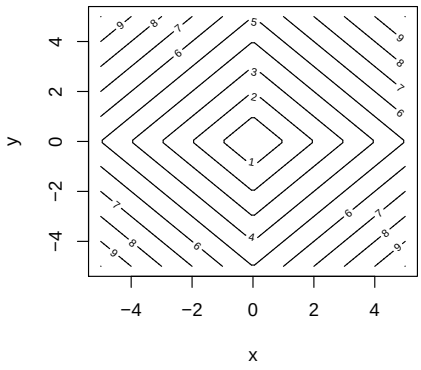
Of course there is a whole range of Lp-norms that could be used. However, the optimization problem is only convex for $p \geq 1$, and the penalty function contours only have corners, and therefore perform zero forcing, for $p \leq 1$, making $p = 1$ a very natural choice in this framework. LASSO regression corresponds to MAP estimation where the coefficient prior is Laplacian. LASSO models can be fit efficiently using Least Angle Regression [Efron et al., 2004] when the likelihood function is Gaussian. The covariates most correlated with the response are added in turn, using an analytic expression to calculate the optimal coefficient.

1.7 Sparse Factor Analysis

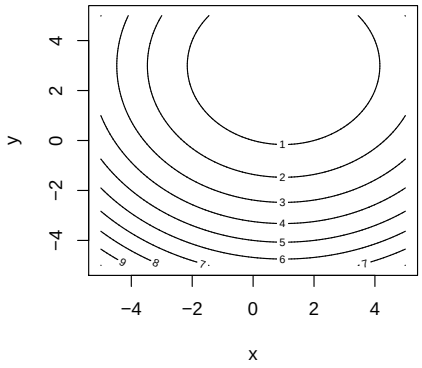
The idea of using a Student-t prior, decomposed as a scale mixture of normal, in factor analysis, seems to have been simultaneously proposed in Fokoue [2004] and Fevotte and Godsill [2006]. The elements of the mixing matrix $\mathbf{G}$ are given a Student-t distribution, and efficient inference is performed by



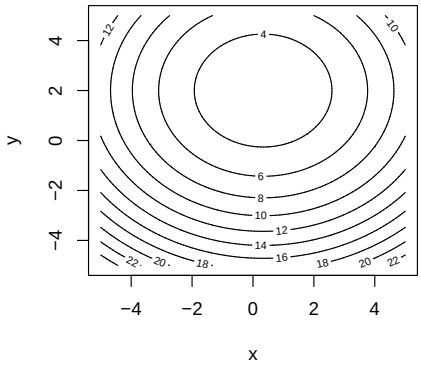
(a) Contours of the L2 norm



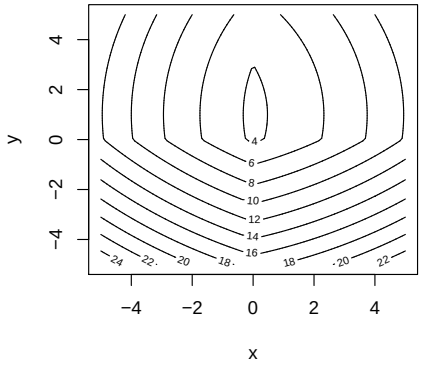
(b) Contours of the L1 norm



(c) Log likelihood function



(d) Objective + L2 norm



(e) Objective + L1 norm

Figure 1.3: Illustration of the difference between L1 (LASSO) and L2 (ridge regression) norms. Each plot shows the contours of a function.

introducing a per element precision as in Equation 1.3. Fokoue [2004] and Fevotte and Godsill [2006] perform Gibbs sampling, whereas Cemgil et al. [2005] use variational EM [Wainwright and Jordan, 2003; Ghahramani and Beal, 2001].

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

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

In BFRM a hierarchical prior is used:

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

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

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

1.8 The Indian Buffet Process

Our model 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. The Indian Process Process (IBP) was originally introduced by Griffiths and Ghahramani [2005]. A two parameter generalisation is developed in [Ghahramani et al., 2007], and a stick-breaking construction which motivates a slice sampling algorithm in [Teh et al., 2007].

1.8.1 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

$$P(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{k=1}^{K}\prod_{d=1}^{D} P(z_{dk}|\pi_k) = \prod_{k=1}^{K} \pi_k^{m_k}(1-\pi_k)^{D-m_k} \tag{1.7}$$

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

$$\pi_k|\alpha \sim \text{Beta}\left(\frac{\alpha}{K}, 1\right) \tag{1.8}$$

$$z_{dk}|\pi_k \sim \text{Bernoulli}(\pi_k) \tag{1.9}$$

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

$$P(\mathbf{Z}) = \prod_{k=1}^{K} \frac{\frac{\alpha}{K}\Gamma(m_k + \frac{\alpha}{K})\Gamma(D - m_k + 1)}{\Gamma(D + 1 + \frac{\alpha}{K})} \tag{1.10}$$

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

1.8.2 Take the infinite limit.

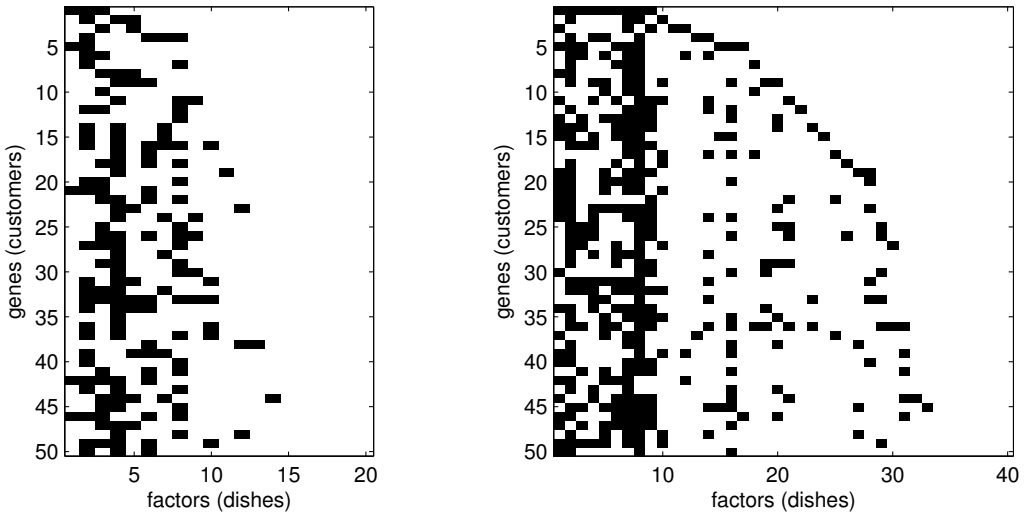
Griffiths and Ghahramani [2005] define a scheme to order the non-zero rows of $\mathbf{Z}$ which allows us to take the limit $K \to \infty$ and find

$$P(\mathbf{Z}) = \frac{\alpha^{K_+}}{\prod_{h>0} K_h!} \exp\left(-\alpha H_D\right) \prod_{k=1}^{K_+} \frac{(D - m_k)!(m_k - 1)!}{N!} \tag{1.11}$$

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.

1.8.3 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 to have previously sampled dish k. Having reached the end of the previously sampled dishes, he tries Poisson($\frac{\alpha}{i}$) new dishes. Figure 1.4 shows two draws from the IBP for two different values of α.



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

Figure 1.4: 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 exchangeable distribution. Since the distribution is exchangeable with respect to the customers we find by considering the last customer that

$$P(z_{kt} = 1|\mathbf{z}_{-kn}) = \frac{m_{k,-t}}{D} \tag{1.12}$$

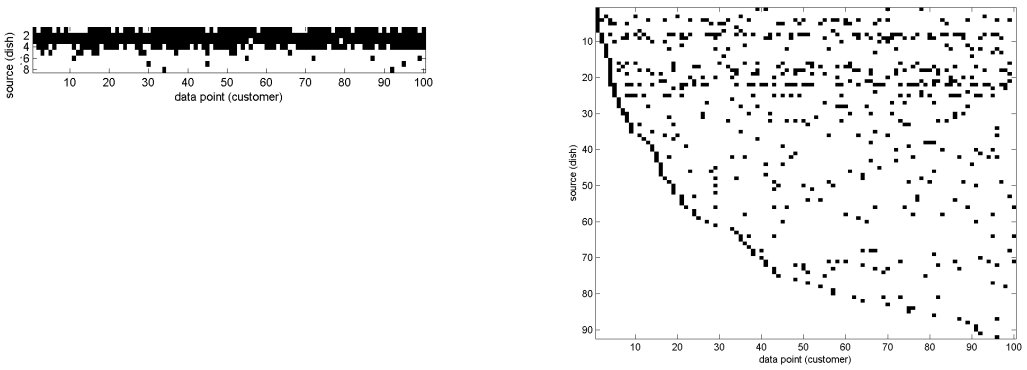
where $m_{k,-t} = \sum_{s \neq t} z_{ks}$, which is used in sampling $\mathbf{Z}$. By exchangeability and considering the first customer, the number of active sources for dimension follows a Poisson(α) distribution, and the expected number of entries in $\mathbf{Z}$ is $D\alpha$.

We also see that the number of active features, $K_+ = \sum_{d=1}^{D} \text{Poisson}(\frac{\alpha}{d}) = \text{Poisson}(\alpha H_D)$.

1.8.4 Two parameter generalisation.

A problem with the one parameter IBP is that the number of features per object, α, and the total number of features, $N\alpha$, are both controlled by α and cannot vary independently. Under this model, we cannot tune how likely it is for features to be shared across objects. To overcome this restriction we follow Ghahramani et al. [2007], introducing β, a measure of the feature *repulsion*. The ith customer now samples dish k with probability $\frac{m_k}{\beta+i-1}$ and samples Poisson($\frac{\alpha\beta}{\beta+i-1}$) new dishes.

Figure 1.5 shows draws from the two parameter IBP for two different values of β. For $\beta < 1$ we get increased sharing of sources amongst data points, as in Figure 1.5(a), and for $\beta > 1$ we get reduced sharing, as in Figure 1.5(b).



(a) $\alpha = 8, \beta = 0.1$ (b) $\alpha = 8, \beta = 4$

Figure 1.5: Draws from the one parameter IBP for two different values of α.

Following the same steps as for the one parameter IBP, we find

$$P(z_{kt} = 1 | \mathbf{z}_{-kn}, \beta) = \frac{m_{k,-t}}{\beta + D - 1} \tag{1.13}$$

The marginal probability of $\mathbf{Z}$ becomes

$$P(\mathbf{Z}|\alpha, \beta) = \frac{(\alpha\beta)^{K_+}}{\prod_{h>0} K_h!} \exp\left(-\alpha H_D(\beta)\right) \prod_{k=1}^{K_+} B(m_k, D - m_k + \beta) \tag{1.14}$$

where C is a constant with respect to α and β, and $H_D(\beta) = \sum_{j=1}^{D} \frac{\beta}{\beta+j-1}$. The expected overall number of active features is now $\bar{K}_+ = \alpha H_D(\beta)$. We will derive all our results for the two parameter case because it is straightforward to recover the one parameter case by setting $\beta = 1$.

1.8.5 Stick Breaking Construction

An alternative representation of the IBP has recently been proposed for the one-parameter IBP in Teh et al. [2007], which allows a slice sampling method to be derived allowing potentially faster mixing in the non-conjugate source distribution case. Again we start with the finite case, but now construct a decreasing ordering of the π_k of Equation (1.8): $\pi_{(1)} > \pi_{(2)} > \cdots > \pi_{(K)}$. In Teh et al. [2007] it is shown that $\mu_{(k)}$ obey the following equation:

$$\nu_{(k)} \sim \text{Beta}\,(\alpha, 1) \tag{1.15}$$

$$\pi_{(k)} = \nu_{(k)}\mu_{(k-1)} = \prod_{l=1}^{k} \nu_{(l)} \tag{1.16}$$

The analogy we use is as follows. We start with a stick of length one, and break off a length $\nu_{(1)}$, and record its length as $\pi_{(1)}$. At iteration k, we break off a length $\nu_{(k)}$ relative to the remaining length, and record its length as $\pi_{(k)}$.

1.8.6 Accelerated sampling

Many serial procedures have been developed for inference in the IBP, including variants of Gibbs sampling [Griffiths and Ghahramani, 2005; Doshi-Velez and Ghahramani, 2009], which may be augmented with Metropolis split-merge proposals [Meeds et al., 2006], slice sampling [Teh et al., 2007], particle filtering [Wood and Griffiths, 2007], and variational inference [Doshi-Velez et al., 2009]. With the exception of the accelerated Gibbs sampler of Doshi-Velez and Ghahramani [2009], these methods have been applied to datasets with less than 1,000 observations.

The motivation for the accelerated sampler is the following, in the context of the bilinear model $X = ZA + \epsilon$. Here X is the $N \times D$ data matrix, A is a matrix with a priori standard Gaussian independent variables, and ϵ is isotropic Gaussian noise. The uncollapsed sampler (where we sample

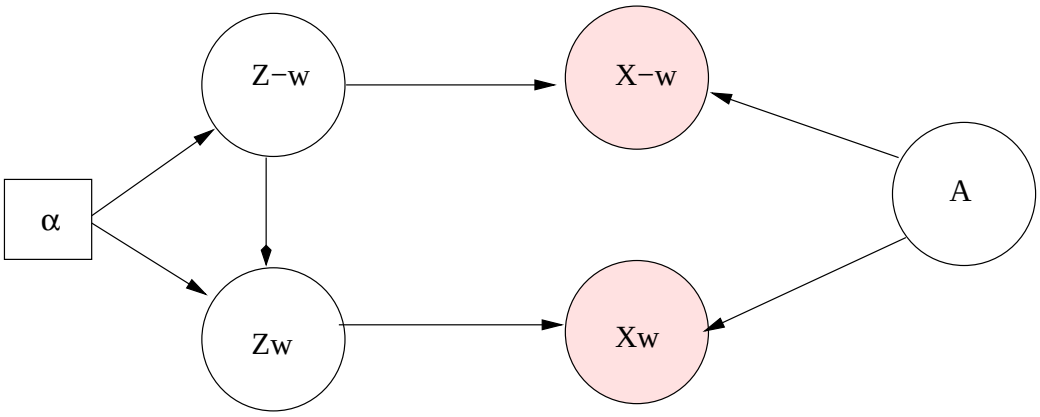


Figure 1.6: Graphical model for an IBP latent factor model, showing the observations and the feature-assignment matrix Z split in two. In this drawing, the observations corresponding to Z_W are imagined to have occured "after" Z_{-W} in the generative sampling process, and thus depend on the counts in Z_{-W} [From Doshi-Velez and Ghahramani [2009]].

both Z and A) has very poor mixing properties, because the product ZA is constrained equal to the data matrix X (apart from some small noise term). This makes it very difficult for the chain to explore the full parameter space because Z and A are highly correlated in the posterior distribution. The collapsed sampler has much better mixing properties because it avoid this problem, but at great computational cost: for each element of Z that is sampled the calculation involves the entire X matrix, rather than just the row associated with that particular element. The accelerated sampler aims to achieve the best of both worlds: the flexibility and therefore the mixing properties of the collapsed sampler with only a slightly increase in computational load compared to the uncollapsed sampler.

The accelerated sampler proceeds as follows. Following Doshi-Velez and Ghahramani [2009]: let X_W denote some set of W observations that contains the observation n. The exchangeability of the IBP allows us to assume without loss of generality that X_W were the final W observations and observation n is the last observation sampled in the generative process. To sample Z we

need $p(Z_{nk} = 1|Z_{-nk}, X)$, given by Bayes rule:

$$\begin{aligned} p(Z_{nk} = 1|Z_{-nk}, X) &\propto p(Z_{nk}|Z_{-nk})p(X|Z) \\ &= \frac{m_k}{n} \int_A p(X|Z, A)p(A)dA. \end{aligned}$$

We split the data into sets X_W and X_{-W} and apply the conditional independencies implied in figure 1.6 to get

$$\begin{aligned} & p(Z_{nk} = 1|Z_{-nk}, X) && (1.17) \\ &= \frac{m_k}{n} \int_A p(X_W, X_{-W}|Z_W, Z_{-W}, A)p(A)dA && (1.18) \\ &= \frac{m_k}{n} \int_A p(X_W|Z_W, A)p(X_{-W}|Z_{-W}, A)p(A)dA && (1.19) \end{aligned}$$

Finally, we apply Bayes rule again to $p(X_{-W}|Z_{-W}, A)$:

$$\begin{aligned} & p(Z_{nk} = 1|Z_{-nk}, X) && (1.20) \\ &= \frac{m_k}{n} \int_A p(X_W|Z_W, A)p(A|Z_{-W}, X_{-W})p(X_{-W}|Z_{-W})dA \\ &\propto \frac{m_k}{n} \int_A p(X_W|Z_W, A)p(A|X_{-W}, Z_{-W})dA \end{aligned}$$

Thus, given the posterior distribution $p(A|X_{-W}, Z_{-W})$, it is possible to compute $p(Z_{nk} = 1|Z_{-nk}, X)$ without involving the remaining data, X_{-W}.

The accelerated sampler proceeds by maintaining the distribution $P(A|X, Z)$. For conjugate exponential models the contribution of the window Z_W is easily removed to give $p(A|X_{-W}, Z_{-W})$. This is used for sampling according to Equation 1.20, and $P(A|X, Z)$ is again calculated using only a rank-one update depending on the new rows Z_W.

Chapter 2

Non-parametric sparse factor models

In our previous work [Knowles and 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 hierachical sparsity prior they use is somewhat different, see Section 1.7. 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 1.5 for more details. We compare these three sparsity schemes empirically in the context of gene expression modelling.

We use the Indian Buffet Process [Griffiths and Ghahramani, 2005], which defines a distribution over infinite binary matrices, to provide sparsity and a framework for infering the appropriate latent dimension of 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 and Daumé III [2008], and is a simultaneous development.

2.1 The Model

We will define our model in terms of Equation 1.1. 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 mixing matrix by

$$p(G_{dk}|Z_{dk}, \lambda_k) = Z_{dk}\mathcal{N}\left(G_{dk}; 0, \lambda_k^{-1}\right) + (1 - Z_{dk})\delta_0(G_{dk}) \tag{2.1}$$

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 Φ. 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.

2.2 Inference

Given the observed data $\mathbf{Y}$, we wish to infer the hidden sources $\mathbf{X}$, which sources are active $\mathbf{Z}$, the mixing 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 marginal 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

$$P(\mathbf{Y}|\mathbf{G}, \mathbf{X}, \boldsymbol{\psi}) = \prod_{t=1}^{N} \frac{1}{(2\pi)^{\frac{D}{2}}|\boldsymbol{\psi}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{y}_n - \mathbf{G}\mathbf{x}_n)^T\boldsymbol{\psi}^{-1}(\mathbf{y}_n - \mathbf{G}\mathbf{x}_n)\right) \tag{2.2}$$

Mixture coefficients. We first derive a Gibbs sampling step for the individual elements of the IBP matrix, $\mathbf{Z}$. Integrating out the (d, k)-th element

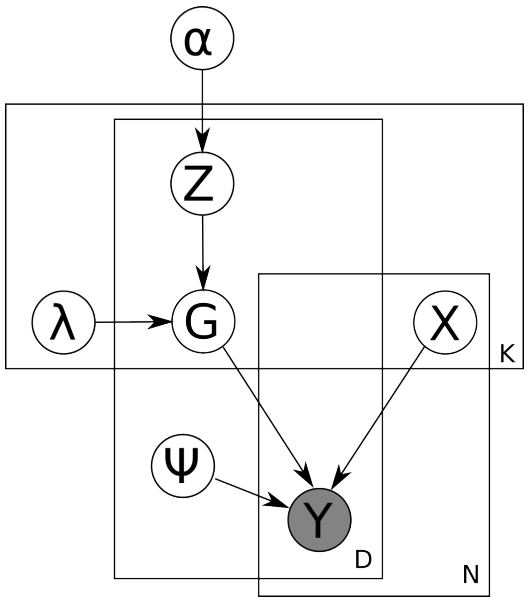


Figure 2.1: Graphical model

of the factor loading matrix, g_{dk}, in Equation 2.1 we obtain

$$\frac{P(\mathbf{Y}|Z_{dk}=1,-)}{P(\mathbf{Y}|Z_{dk}=0,-)} = \frac{\int P(\mathbf{Y}|g_{dk},-)\mathcal{N}\left(g_{dk};0,\lambda_k^{-1}\right)dg_{dk}}{P(\mathbf{Y}|g_{dk}=0,-)} \tag{2.3}$$

$$= \sqrt{\frac{\lambda_k}{\lambda}}\exp\left(\frac{1}{2}\lambda\mu^2\right) \tag{2.4}$$

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

$$\frac{P(Z_{dk}=1|-)}{P(Z_{dk}=0|-)} = \frac{m_{-d,k}}{N-1-m_{-d,k}} \tag{2.5}$$

where $m_{-d,k}$ is the number of dimensions for which factor k is active, excluding the current dimension d. Multiplying Equations 2.4 and 2.5 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 2.4.

Adding new features. $\mathbf{Z}$ is a matrix with infinitely many columns, but 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. Let κ_d be 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. Note that due to the form of the prior for elements of $\mathbf{Z}$ in Equation 2.5, $\kappa_d = 0$ for all d after a sampling sweep of $\mathbf{Z}$.

Figure 2.2 illustrates κ_d for a sample $\mathbf{Z}$ matrix.

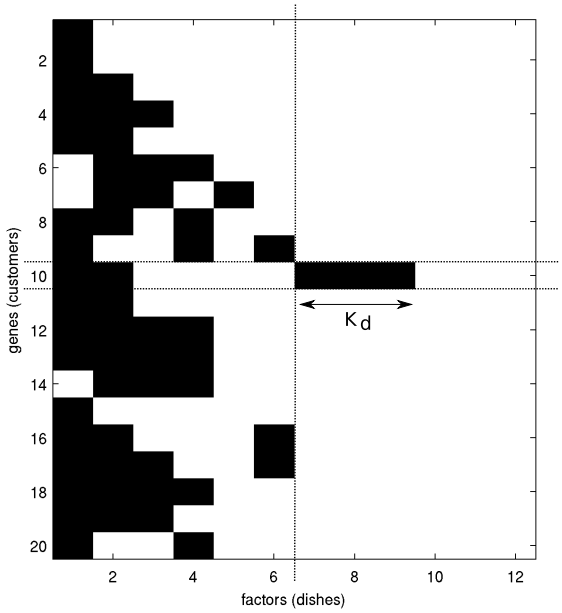


Figure 2.2: A diagram to illustrate the definition of κ_d, for $d = 10$.

New features are proposed by sampling κ_d with a MH step. It is possible 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 $\mathbf{g}^T$ as part of the proposal. Thus the proposal

is $\xi = \{\kappa_d, \mathbf{g}\}$, and we propose a move $\xi \to \xi^*$ with probability $J(\xi^*|\xi)$. In this case $\xi = \emptyset$ since as noted above $\kappa_d = 0$ initially. The simplest proposal, following Meeds et al. [2006], would be to use the prior on ξ^*, i.e.

$$J(\xi) = P(\kappa_d|\alpha) \cdot p(\mathbf{g}|\kappa_d, \lambda_k) = \text{Poisson}\,(\kappa_d; \gamma) \cdot N(\mathbf{g}; 0, \lambda_k^{-1})$$

where $\gamma = \frac{\alpha}{D-1}$.

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.

$$J(\kappa_d) = (1-\pi)\text{Poisson}\,(\kappa_d; \lambda\gamma) + \pi\mathbf{1}(\kappa_d = 1) \tag{2.6}$$

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 $\min(1, a_{\xi\to\xi^*})$ where

$$a_{\xi\to\xi^*} = \frac{P(\xi^*|\text{rest}, Y)J(\xi|\xi^*)}{P(\xi|\text{rest}, Y)J(\xi^*|\xi)} = \frac{P(Y|\xi^*, \text{rest})P(\kappa_d|\alpha)p(\mathbf{g}|\kappa_d, \lambda_k)}{P(Y|\text{rest})J(\kappa_d)p(\mathbf{g}|\kappa_d, \lambda_k)} = a_l \cdot a_p \tag{2.7}$$

where

$$a_l = \frac{P(Y|\xi^*, \text{rest})}{P(Y|\text{rest})} \tag{2.8}$$

$$a_p = \frac{P(\kappa_d|\alpha)}{J(\kappa_d)} = \frac{\text{Poisson}\,(\kappa_d; \gamma)}{\text{Poisson}\,(\kappa_d; \lambda\gamma)} \tag{2.9}$$

Note that we can simply take $J(\xi|\xi^*) = 1$ since $\xi = \emptyset$. To calculate a_l we need the collapsed likelihood under the new proposal:

$$P(Y_{d:}|\xi^*, -) = \prod_{n=1}^{N} \int P(Y_{dn}|\xi^*, \mathbf{x}'_n, -)P(\mathbf{x}'_n)d\mathbf{x}' \tag{2.10}$$

$$= \prod_{n=1}^{N} (2\pi\psi_d^{-1})^{-\frac{1}{2}}(2\pi)^{\frac{\kappa_d}{2}}|\mathbf{M}|^{-\frac{1}{2}} \exp\left(\frac{1}{2}(\mathbf{m}_n^T\mathbf{M}\mathbf{m}_n - \psi_d^{-1}\hat{E}_{dn}^2)\right) \tag{2.11}$$

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 matrix of residuals $\hat{\mathbf{E}} = \mathbf{Y} - \mathbf{G}\mathbf{X}$. Note that : denotes taking a "slice" of a matrix. The likelihood under the current sample is:

$$P(Y_{d:}|\xi, -) = \prod_{n=1}^{N}(2\pi\psi_d^{-1})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\psi_d^{-1}\hat{E}_{dn}^2\right) \tag{2.12}$$

Substituting these likelihood terms into the expression for the ratio of likelihood terms, a_l, gives

$$a_l = (2\pi)^{\frac{N\kappa_d}{2}}|\mathbf{M}|^{-\frac{N}{2}} \exp\left(\frac{1}{2}\sum_n \mathbf{m}_n^T\mathbf{M}\mathbf{m}_n\right) \tag{2.13}$$

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 parameters. We can choose to sample the IBP strength parameter α, with conjugate Gamma(e, f) prior (note that we use the inverse scale parameterisation of the Gamma distribution). The conditional prior of Equation (1.11), acts as the likelihood term and the posterior update is as follows:

$$P(\alpha|\mathbf{Z}) \propto P(\mathbf{Z}|\alpha)P(\alpha) = \text{Gamma}\left(\alpha; K_+ + e, f + H_D\right) \tag{2.14}$$

where K_+ is the number of active sources.

If the two parameter IBP (see Section 1.8.4) is being used, we can sample β by a Metropolis-Hasting's step with acceptance probability $\min(1, r_{\beta\to\beta^*})$. By Equation (2.7) we know that setting the proposal distribution equal to the prior, i.e. $J(\beta^*|\beta) = P(\beta^*) = \text{Gamma}(1, 1)$, results in $r_{\beta\to\beta^*}$ being equal to the ratio of likelihoods, in this case $\frac{P(Z|\alpha,\beta^*)}{P(Z|\alpha,\beta)}$ as given in Equation (1.14).

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 $t \in [1, \ldots, T]$ we have

$$P(\mathbf{x}_t|-) \propto P(\mathbf{y}_t|\mathbf{x}_t, -)P(\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_t; \boldsymbol{\mu}_t, \boldsymbol{\Lambda}\right) \tag{2.15}$$

where $\boldsymbol{\Lambda} = \mathbf{G}^T\boldsymbol{\psi}^{-1}\mathbf{G} + I$ and $\boldsymbol{\mu}_t = \boldsymbol{\Lambda}^{-1}\mathbf{G}^T\boldsymbol{\psi}^{-1}\mathbf{y}_t$. 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 inverting it is $O(K^3)$. Calculating $\boldsymbol{\mu}_t$ is order $O(KD)$ and must be calculated for all N $\mathbf{x}_t$'s, a total of $O(NKD)$. Thus sampling $\mathbf{X}$ is order $O(K^2 + K^3 + NKD)$.

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

However, if the variances are allowed to be different for each column of $\mathbf{G}$, we set $\lambda_k \sim \text{Gamma}(c, d)$, and the posterior update is given by $\lambda_k|G \sim \text{Gamma}(c + \frac{m_k}{2}, d + \sum_d G_{dk}^2)$. In this case we may also wish to share power across factors, in which case we also sample d. Putting a Gamma prior on d such that $d \sim \text{Gamma}(c_0, d_0)$, the posterior update is $d|\lambda_k \sim \text{Gamma}(c_0 + cK, d_0 + \sum_{k=1}^K \lambda_k)$

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

However, if the noise is only assumed to be independent, then each dimension has a separate variance, whose inverse is given a Gamma prior: $\psi_d^{-1} \sim \text{Gamma}(a, b)$ which gives the posterior update $\psi_d^{-1}|- \sim \text{Gamma}(a + \frac{N}{2}, b + \sum_n E_{dn}^2)$. If b is given prior distribution $\text{Gamma}(a_0, b_0)$ the Gibbs update is $b|- \sim \text{Gamma}(a_0 + aD, b_0 + \sum_{d=1}^D \psi_d^{-1})$.

Algorithm 1 One iteration of the nsFA sampler

```
for d = 1 to D do
  for k = 1 to K do
    Sample Z_dk using Equation 2.5
  end for
  Sample κ_d using Equation 2.7
end for
for n = 1 to N do
  Sample X_:n using Equation 2.15
end for
Sample α, φ, λ_g as detailed above.
```

2.3 Results

We compare the following models:

- FA - Bayesian Factor Analysis
- A - Factor Analysis with ARD prior to determine active sources
- S - Sparse Factor Analysis, using the finite IBP
- NS - The proposed model: Nonparametric Sparse Factor Analysis
- W - Bayesian Factor Regression Model of West et al. [2007].
- F - The sparse Factor Analysis method of Fokoue [2004], Fevotte and Godsill [2006] and Archambeau and Bach [2009]

Note that all of these models can be learned using the software package we provide simply by using appropriate settings.

2.3.1 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. We generate random datasets with $N = 100$ samples by drawing $\mathbf{G}$ and $\mathbf{X}$ from a zero mean unit variance Gaussian, calculating $\mathbf{Y} = (\mathbf{G} \odot \mathbf{Z})\mathbf{X} + \mathbf{E}$, where $\mathbf{E}$ is Gaussian white noise with variance set to give a signal to noise ratio of 10.

Here we will define the reconstruction error, E_r as

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

where $\hat{G}, \hat{K}$ are the inferred quantities. Our aim is to minimise over permutations, but we do this by greedily minimising the alignment of true and inferred factors. We average this error over the last ten samples of the MCMC run. Note that this error function does not penalise inferring extra spurious factors, so we will investigate this possibility separately.

Figure 2.3.1 shows the reconstruction error for each method and different numbers of latent features, across ten random datasets and 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 $\hat{K} = 20$ (FA20) the variance is also very high, since the four spurious features fit noise. Using an ARD prior on the features (A) improves the performance, and overfitting no longer occurs. The reconstruction error is actually less for $\hat{K} = 20$, but this is an artifact due to the reconstruction error not penalising additional spurious features in the inferred $\mathbf{G}$. Sparse factor analysis (S), the finite version of the full infinite model, performs very well. The Bayesian Factor Regression Model (W) performs significantly better than the ARD factor analysis (A), but not as well as our sparse model (S). 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 (F) only performs marginally better than A, 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 proposed here and in Rai and Daumé III [2008] performs very well. With fixed $\alpha = 1$ (NSa1) or inferring α we see very similiar performance. Sharing power between elements of the noise variance (NSsn) also seems to reduce the variance of the sampler, which is sensible in this example since the noise was in fact isotropic.

Figure 2.4 shows histograms for the number of latent features inferred for the nonparametric sparse model. This represents an approximate posterior over K. For fixed $\alpha = 1$ the distribution is centered around the true value of $K = 16$, with minimal bias ($\mathbb{E}K = 16.1$). The variance is significant (standard deviation of 1.46), but is reasonable considering the noise level (SNR=10) and that in some of the random datasets, elements of $\mathbf{Z}$ which are 1 could be 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 $\mathbf{Z}$. In this case, the sampler always converged to at least 16 features, but would also sometimes infer spurious featues from noise (results not shown). When inferring α some bias and skew are noticable. The mean of the posterior is now at 18.3 with standard deviation 2.0, suggesting there is little to gain from sampling α for this particular dataset. This could be seen as just "good

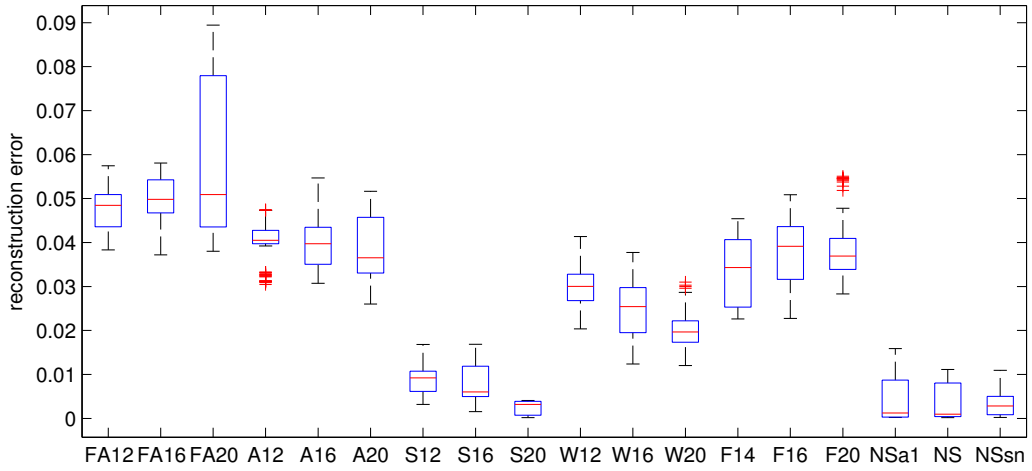


Figure 2.3: Boxplot of reconstruction errors for simulated data derived from the *E. Coli* connectivity matrix of Kao et al. [2004]. Ten datasets were generated and the reconstruction error calculated for the last ten samples from each algorithm. Numbers refer to the number of latent factors used, K. *a1* denotes fixing $\alpha = 1$. *sn* denotes sharing power between noise dimensions.

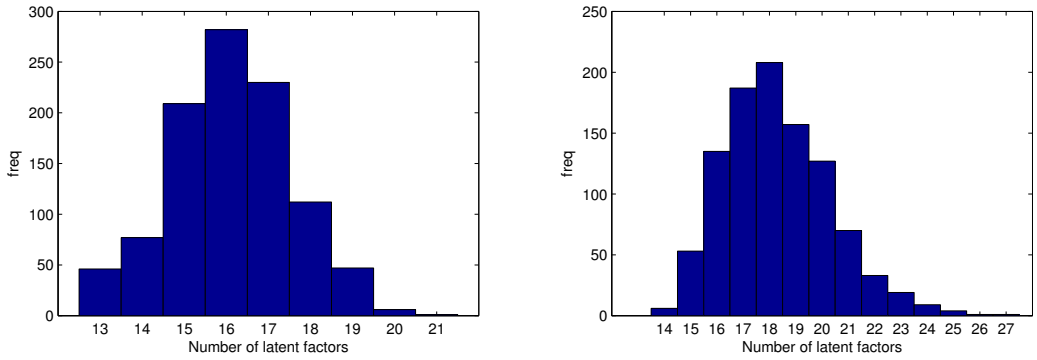


Figure 2.4: Histograms of the number of latent features inferred by the non-parametric sparse FA sampler for the last 100 samples out of 1000. *Left:* With $\alpha = 1$. *Right:* Inferring α.

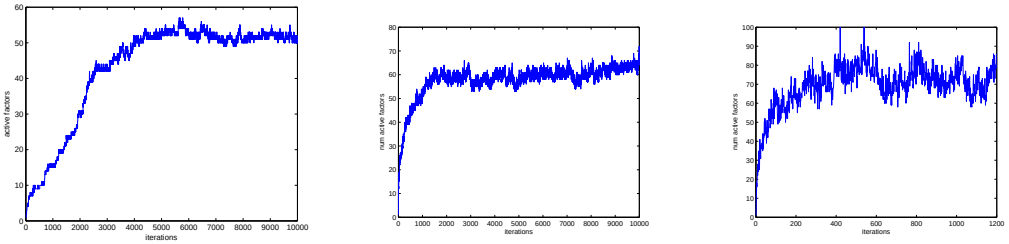
luck" that $\alpha = 1$ happened to be a good choice here.

2.3.2 Convergence

nsFA can suffer from slow convergence if the number of new features is drawn from the prior. Figure 2.5 shows how the different proposals for κ_d effect how quickly the sampler reaches a sensible number of features. If we use the prior as the proposal distribution, mixing is very slow, taking around 5000 iterations to converge, as shown in Figure 2.5(a). If a mass of 0.1 is added at $\kappa_d = 1$, then the sampler reaches the equilibrium number of features in around 1500 iterations, as shown in Figure 2.5(b)). However, if we try to add features even faster, for example by setting the factor $\lambda = 50$ in Equation 2.6, then the sampling noise is greatly increased, as shown in Figure 2.5(c), and the computational cost also increases significantly because so many spurious features are proposed only to be rejected.

2.3.3 Biological Data

To assess the performance of each algorithm on the biological data where no ground truth is available, we calculated the test set log likelihood under the posterior. 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



(a) Prior. (b) Prior plus $0.1\mathbb{I}(\kappa = 1)$. (c) Factor $\lambda = 50$.

Figure 2.5: The effect of different proposal distributions for the number of new features.

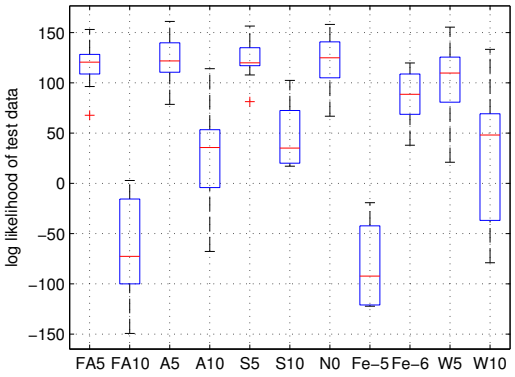
signal to noise ratio across gene expression level probes.

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

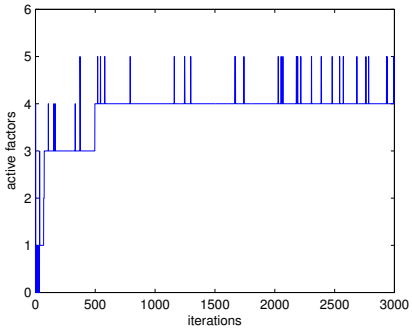
Figure 2.6(a) shows the test log likelihood achieved by the various algorithms on the *E. Coli* dataset, including 100 genes at 24 time-points. On this simple dataset incorporating sparsity doesn't improve predictive performance. Overfitting the number of latent factors does damage performance, although 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 2.6(b).

Breast cancer dataset from West et al. [2007]

Figure 2.7(a) shows the test log probability for the breast cancer dataset of West et al. [2007], including 226 genes across 251 individuals. 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 probability was calculated using the final 100 MCMC samples. The settings used for each algorithm are available on the author's website. 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 (A), giving improved performance when using 20 features and only slightly reduced performance when 40 features are used. The sparse finite model (S) shows an advantage over A in terms of predictive performance as long as underfitting does not occur: performance



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

is comparable when using only 10 features. The performance of the sparse nonparametric model (N) 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 (F) was run with $K = 20$ and various settings of the hyperparameter d which controls the overall sparsity of the solution (Fe-3 corresponds to setting $d = 10^{-3}$ and so on). The model's predictive performance depends strongly on the setting of this parameter, with results approaching the performance of the sparse models (S and N) for $d = 10^{-4}$. The performance of BFRM (W) on this dataset is noticeably worse than the other sparse models.

As described in Section 2.2, sampling $\mathbf{Z}$ and $\mathbf{G}$ takes order $O(NKD)$ operations per iteration, and sampling $\mathbf{X}$ takes $O(K^2 + K^3 + NKD)$. However, for the moderate values encoutered 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 2.7(c) 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

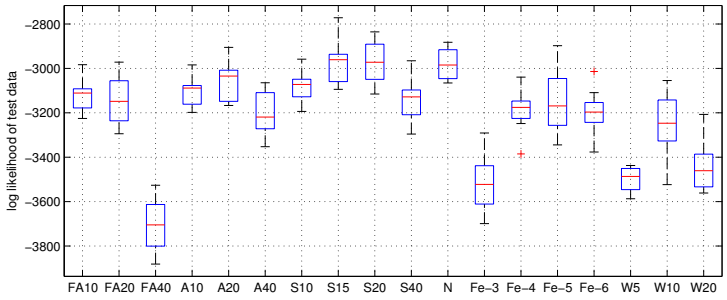
on $\mathbf{X}$, must be contributing. The computational cost of adding the ARD prior is negligible (A). For the sparse finite model The CPU time per iteration is just over double for the sparse finite model (S), but the cost actually decreases with increasing K, because the sparsity of the solution increases to avoid overfitting. There are less 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 (N) is somewhat higher than for the finite model because of the cost of the feature birth and death process. However, Figure 2.7(b) 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 (S15) and cheaper than choosing an unnecessarily large finite model (S20 and S40). Fokoue's method (F) 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 F is continuous, making search easier but requiring a normal random variable for every element of $\mathbf{G}$. W pays a considerable computational cost for both the hierarchial sparsity prior and the DP prior on $\mathbf{X}$.

Prostate cancer dataset from Yu et al. [2004]

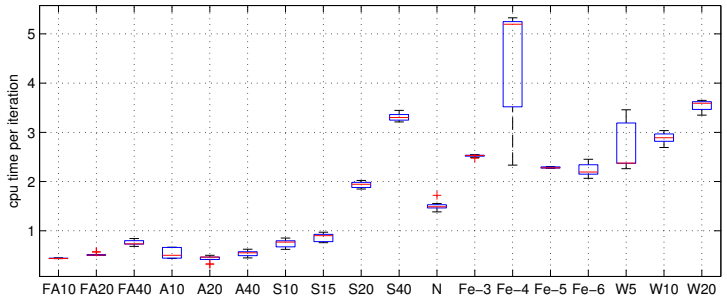
Figure 2.8(b) shows the predictive performance of A, F and N on the prostate cancer dataset of Yu et al. [2004], including 12557 genes across 171 individuals. The large number of genes in this dataset makes inference slower, but the problem is manageable since the computational complexity is linear in the number of genes. Despite the large number of genes, the appropriate number of latent factors, in terms of maximising predictive performance, is still small, around 10 (nsFA infers a median of 12 factors). This may seem small relative to the number of genes, but it should be pointed out that the genes included in the breast cancer and *E. Coli* datasets are those capturing the most variability. Running 1000 iterations of nsFA takes under 8 hours.

2.4 Discussion

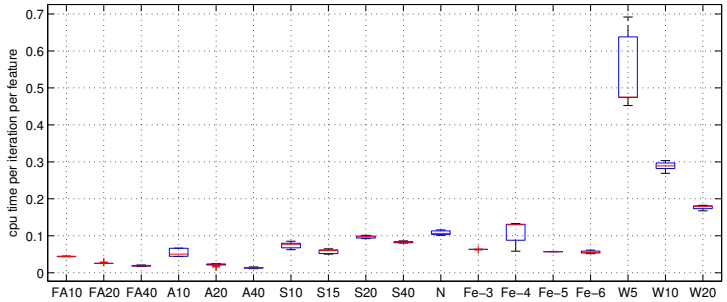
We have seen that in both the *E. Coli* and breast cancer datasets that sparsity can improve predictive performance, as well as providing a more easily interpretable solution. Using the IBP to provide sparsity is straightforward, and allows the number of latent factors to be inferred within a well defined



(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.

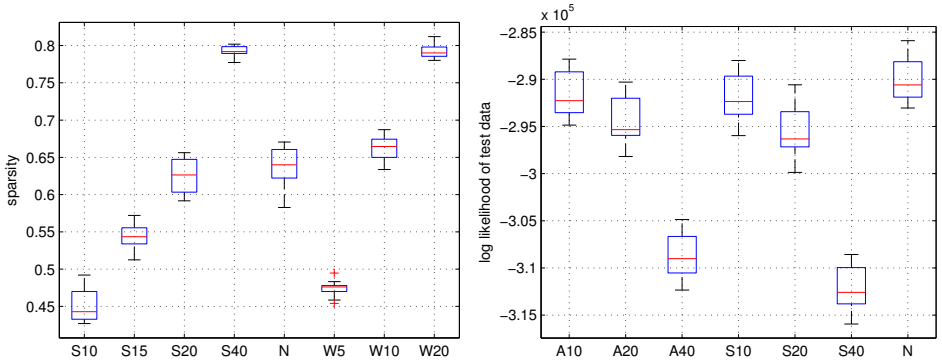


(b) CPU time per iteration, averaged across the 3000 iteration run.



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

Figure 2.7: Results on breast cancer dataset ($N = 251, D = 226$, 3000 MCMC iterations).



(a) Sparsity, calculated as the number of zero elements of $\mathbf{G}$ divided by the total number of elements ($N \times K$).

(b) Results on Prostate cancer dataset from Yu et al. [2004], including 12557 genes across 171 individuals (1000 MCMC iterations).

Figure 2.8: Further results on biological data.

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 using appropriate ARD-like priors, we find this is typically more computationally expensive than the birth and death process of the IBP. The BFRM model's more complex prior may have improved interpretability but at a cost of more computationally involved inference and reduced predictive performance. Manual model selection is an alternative but is time consuming. Finally we show that running nsFA on full gene expression datasets with 10000+ genes is feasible so long as the number of latent factors remains relatively small. An interesting direction for this 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 interpretibility by associating latent variables with specific transcription factors.

Chapter 3

Data Parallelisation in the Indian Buffet Process

Bioinformatics problems are becoming increasingly large scale. Gene expression data will typically include tens of thousands of probes, clinical datasets may have thousands of individuals and thousands of measurements, and SNP (Single Nucleotide Polymorphism) arrays can have a million probes per sample. Nonparametric models provide a framework to learn the appropriate size models to use for these huge datasets, but scaling cutting edge collapsed sampling algorithms to these sizes is challenging in terms of both CPU time and memory usage.

Advances in multicore and distributed computing provide one answer to this challenge: if each processor can consider only a small part of the data, then inference in these large datasets might become more tractable. However, such *data parallelisation* of inference is nontrivial—while simple models might only require pooling a small number of sufficient statistics [Chu et al., 2007], correct inference in more complex models can depend on frequent sychronization or worse yet, having to communicate entire probability distributions between processors. Building on work on approximate asynchronous multicore inference for topic models [Asuncion et al., 2008], we develop a message passing framework for data-parallel Bayesian inference applicable to nonparametric models.

To achieve efficient parallelisation, we exploit the idea, recently introduced by Doshi-Velez and Ghahramani [2009], and described in Section 1.8.6, of maintaining a distribution over parameters while sampling. This idea, coupled with a message passing scheme over processors, makes it possible

to distribute inference over many processors while sacrificing little accuracy in inference. We demostrate our approach by scaling IBP inference to a problem with 57,000 observations, the largest application to date. As most elements of our procedure are general to data parallelisation in other models, our work opens up the use of the IBP and similar models in data-intensive fields, enabling these modeling technologies to be harnessed in a broad range of applications.

3.1 Latent Feature Model

We associate with Z, the feature assignment matrix, a feature matrix A with rows that parameterise the effect that possessing each feature has on the data. Given these matrices, we write the probability of the data as $P(X|Z, A)$. Our work requires that $P(A|X, Z)$ can be computed or approximated efficiently by an exponential family distribution. Specifically, we apply our techniques to both a fully-conjugate linear-Gaussian model and non-conjugate Bernoulli model.

Linear Gaussian Model. This models an $N \times D$ real-valued data matrix X as a product:

$$X = ZA + \epsilon, \tag{3.1}$$

where Z is the binary feature-assignment matrix and A is a K by D real-valued matrix with an independent Gaussian prior $N(0, \sigma_a^2)$ on each element. Each element of the N by D noise matrix ϵ is independent with a $N(0, \sigma_n^2)$ distribution. Given Z and X, the posterior on the features A is Gaussian, given by mean and covariance

$$\mu^A = (Z^T Z + \frac{\sigma_x^2}{\sigma_a^2} I)^{-1} Z^T X \tag{3.2}$$

$$\Sigma^A = \sigma_x^2 (Z^T Z + \frac{\sigma_x^2}{\sigma_a^2} I)^{-1} \tag{3.3}$$

Bernoulli Model. This models an $N \times D$ binary-valued data matrix X using a leaky noisy-or likehood for each element:

$$P(X_{nd} = 1|Z, A) = 1 - \epsilon\, \lambda^{\sum_k Z_{nk} A_{kd}}. \tag{3.4}$$

Each element of the A matrix is binary with independent $\mathsf{Bernoulli}(p_A)$ priors. The parameters ϵ and λ determine how "leaky" and how "noisy" the or-function is, respectively. Typical hyperparameter values are $\epsilon = 0.95$ and $\lambda = 0.2$. The posterior $P(A|X, Z)$ cannot be computed in closed form; however, a mean-field variational posterior in which we approximate $P(A|X, Z)$ with as product of independent Bernoulli variables $\prod_{k,d}^{K,D} q_{kd}(a_{kd})$ can be readily derived.

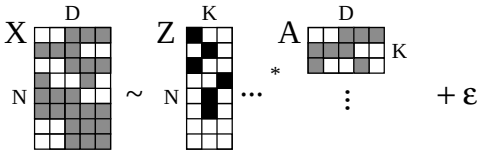
3.2 Parallel Inference

We partition the data set between processors, using X^p and Z^p to denote the portions of the data X and IBP matrix Z assigned to processor p. We also experimented with instead dividing the matrix Z by features, assigning some subset of features to each processor. This approach performed very poorly because each processor is trying to adjust its features to fit the same residual $X - ZA$. When the features are combined, this residual will have been fitting multiple times, so the error will always increase. Therefore we describe only our results for parallelising by samples.

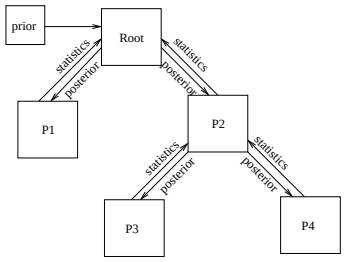
For the fully uncollapsed model (which mixes very poorly), data parallelisation is straightforward. The matrix A and stick lengths π of the IBP must be sampled at the parent node, and sent to each processor. Each processor then samples $P(Z_p|X_p, A, \pi)$, and sends the updated Z_p to the parent. No approximation is introduced, but the bandwidth required is high and this sampler is known to have poor mixing properties because A and Z are highly correlated in the posterior.

We develop a procedure for approximate, parallel inference in the IBP which combines Markov chain Monte Carlo (MCMC) with message passing to approximate the accelerated sampler of [Doshi-Velez and Ghahramani, 2009].

Our algorithm leverages the fact that the observations X are independent given a distribution over the parameters A and the counts from the IBP matrix Z. In Doshi-Velez and Ghahramani [2009], the distribution $P(A|X_{-n}, Z_{-n})$ was used to derive an accelerated sampler for sampling Z_n, where n indexes a data point and $-n$ is the set excluding that point. Similarly, for parallel inference each processor p will maintain a distribution $P^p(A|X_{-n}, Z_{-n})$, where P^p is an approximation to the distribution $P(A|X_{-n}, Z_{-n})$. The distributions P^p can be updated efficiently via message passing between



(a) Representation of the linear-Gaussian model. The data X is generated from the product of the feature assignment matrix Z and feature matrix A. In the Bernoulli model, the product ZA adjusts the probability of $X = 1$



(b) Message passing process. The processors send sufficient statistics of the likelihood up to the root, which calculates and sends the full (exact) posterior down to the processors.

Figure 3.1: Diagrammatic representation of the model structure and the message passing process.

the processors.

The inference alternates between three steps:

- Message passing: processors communicate to compute the exact $P(A|X, Z)$.
- Gibbs sampling: processors sample a new set of Z^p's in parallel.
- Hyperparameter sampling: a designated root processor resamples hyperparameters, which are propagated to other processors

Because all of the processors are sampling Z at once, the posteriors $P^p(A|X, Z)$ used by each processor are no longer exact. However, we show empirically in that this approximation has little effect on the inference.

Message Passing The full posterior on the features A is given by $P(A|Z, X)$. Bayes Rule gives us the following factorisation:

$$P(A|Z, X) \propto P(A) \prod_p P(X^p|Z^p, A) \tag{3.5}$$

If the prior $P(A)$ and the likelihoods $P(X^p|Z^p, A)$ are part of conjugate exponential family models, then the product in equation (3.5) is equivalent to summing the sufficient statistics of the likelihoods from all of the processors. In the linear-Gaussian model, these statistics correspond to mean vectors and

covariance matrices (handled more readily in information mean and precision form); in the Bernoulli model, the statistics correspond to counts on how often each element A_{kd} is equal to one. The linear-Gaussian messages have size $O(K^2 + KD)$, and the Bernoulli messages $O(KD)$. For nonparametric models such as the IBP, the number of features K grows as $O(\log N)$. This slow growth means that messages remain compact and we can efficiently scale to large datasets.

The most straightforward way to accurately compute the full posterior is to network the processors in a tree architecture. The sufficient statistics for the feature posterior are summed via message passing along the tree (which is an instance of belief propagation and is exact). Specifically, the message s from processor p to processor q is given by

$$s_{p\to q} = l^p + \sum_{r\in N(p)\backslash q} s_{r\to p}$$

where $N(p)\backslash q$ are the processors attached to p besides q and l^p are the sufficient statistics from processor p. A dummy neighbour containing the statistics of the prior is connected to one of the processors. We call this processor the root. Also summed are the counts $m_k^p = \sum_{n\in X^p} Z_{nk}^p$, the popularity of feature k within processor p. Figure 3.1(b) shows a cartoon of the message passing process.

Gibbs Sampling In general, Z_{nk} can be Gibbs-sampled using Bayes rule

$$P(Z_{nk}|Z_{-nk}, X) \propto P(Z_{nk}|Z_{-nk})P(X|Z).$$

The probability $P(Z_{nk}|Z_{-nk})$ depends on the total number of observations and the number of observations m_k for which feature k is active. The message passing provides each processor with an accurate value for m_k, from which it may compute $m_k^{-p} = m_k - m_k^p$ using its current count m_k^p. Each processor can update its own m_k^p as it samples its Z^p; it assumes m_k^{-p} stays fixed, which is a good approximation for popular features.

For conjugate models, we evaluate the likelihood $P(X|Z)$ via the integral

$$P(X|Z) \propto \int_A P(X_n|Z_n, A)P(A|Z_{-n}, X_{-n})dA,$$

where the partial posterior $P(A|Z_{-n}, X_{-n})$ is given by

$$P(A|Z_{-n}, X_{-n}) \propto \frac{P(A|Z, X)}{P(X_n|Z_n, A)}. \tag{3.6}$$

Because the model is conjugate, the partial posterior in (3.6) can be efficiently computed by subtracting observation n's contribution to the sufficient statistics.

For non-conjugate models, we can use a variational distribution $Q(A)$ to approximate $P(A|X,Z)$ during message passing. Usually $Q(A)$ is a projection of P onto an exponential family, so computing the partial posterior $Q^{-p}(A)$ is identical to the conjugate case. This can be used for MCMC by initialise a Gibbs sampler with a draw of A from the partial posterior $Q^{-p}(A)$. Samples of A from the uncollapsed sampler are used to compute the sufficient statistics for the likelihood $P(X|Z)$.

Hyperparameter Resampling The IBP concentration parameter α and hyperparameters of the likelihood can also be sampled during inference. Resampling α depends only on the total number of active features; thus it can easily be resampled at the root and propagated to the other processors. In the linear-Gaussian model, the posteriors on the noise and feature variances (starting from gamma priors) depend on various squared-errors, which can also be computed in a distributed fashion.

For more general, non-conjugate models, resampling the hyperparameters requires two steps. In the first step, a hyperparameter value is proposed by the root and propagated to the processors. The processors each compute the likelihood of the current and proposed hyperparameter values and propagate this value back to root. The root evaluates a Metropolis step for the hyperparameters and propagates the decision back to the leaves. The two-step approach *does not* introduce any latency in the resampling so long as the synchronous mode of operation is used.

Asynchronous Operation So far we have discussed message passing, Gibbs sampling, and hyperparameter resampling occuring in separate phases. In practice, these phases may be performed asynchronously: between its Gibbs sweeps, each processor updates its feature posterior based on the most current messages it has received and sends likelihood messages to its parent. Likewise, the root continuously resamples hyperparameters and propagates the values down through the tree. This allows faster processors to share information and perform more inference on their data instead of waiting for slower processors.

When performing parallel inference in the IBP, a few factors need to be

considered with care. Other parallel inference for nonparametric models, such as the HDP [Asuncion et al., 2008], simply matched features by their index, that is, assumed that the i^{th} feature on processor p was also the i^{th} feature on processor q. In the IBP, we find that this indiscriminate feature merging is often disastrous when adding or deleting features; care must taken to ensure that the features stay aligned.

We experimented with various solutions to this problem. One restriction which helps considerably is to only allow unused features to be deleted at the start of an outer iteration, straight after message passing, so that we know these features will be deleted globally. Otherwise the features on different processors become misaligned. We also tried only allowing one processor at a time to add features. This could even be implemented in the asynchronous case using a single "I'm allowed to add features" token passed around by the processors. However, we found that this generally made adding new features too slow, and since new features are generally quite uncertain anyway, concatenating them across processors was not a problem.

Feature Sub-sampling to Scale to Larger Datasets Both likelihood models discussed above require the product ZA to be computed to evaluate each Gibbs update. Normally, this computation requires $O(KD)$ elementary multiplication and addition operations. The overall complexity can be reduced because of the local nature of the Gibbs updates, but it still remains $O(K)$ or $O(D)$, depending on whether the updates are for A or Z, respectively.

If the IBP prior is a reasonable model of the data, we expect the number of features K in a dataset to be $O(\log(N))$. As N grows large, therefore, computations that depend on K are also potentially slow. For larger datasets, we sample only a subset of the K features in each iteration. The features to be sampled is chosen randomly at the start of each Gibbs iteration. As such, over an infinite run of the sampler, each feature will be sampled infinitely often and no additional approximation is introduced. Note that if the features to sample are chosen somehow based on the current state, the Metropolis factor would be altered and Gibbs sampling would no longer be valid.

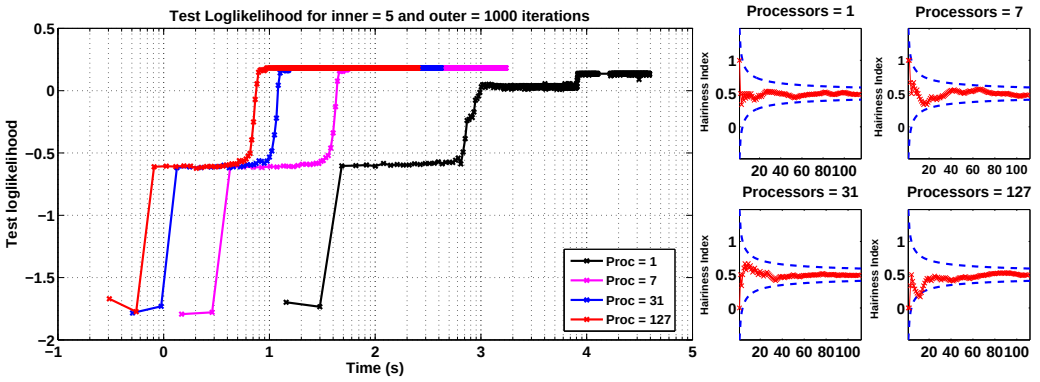


Figure 3.2: Change in likelihood for various numbers of processors over the simulation time. The corresponding hairiness index plots are shown on the left.

3.3 Analysis of Mixing Properties

We ran a series experiments on 10,000 block images of [Griffiths and Ghahramani, 2005] to study the effects of various sampler configurations on running time and performance, as well as the mixing properties of the sampler. We set 5000 data points as test data and the remainder as training data. Figure 3.2 shows the loglikelihood on the test data using 1, 7, 31 and 127 parallel processors simulated in software, using 1000 parallel (outer) iterations and 5 Gibbs (inner) iterations. The parallel samplers are able to reach the same test likelihood levels as the serial algorithm, but with significant savings in running time. It is interesting to note the characteristic shape of the test likelihood, which is similar across all testing regimes. This shape is indicative of the manner in which the features are learnt. Initially, a large number of features are added, which provides improvements in the test likelihood. This is then followed by a refinement phase, where excess features are pruned, providing further improvements.

To gain insight into the tradeoff in choosing between the number of Gibbs (inner) iterations and parallel (outer) iterations, we show the effective number of samples for various numbers of inner iterations, after the chain has been thinned and burnt-in, in Figure 3.3(a). The number of effective samples

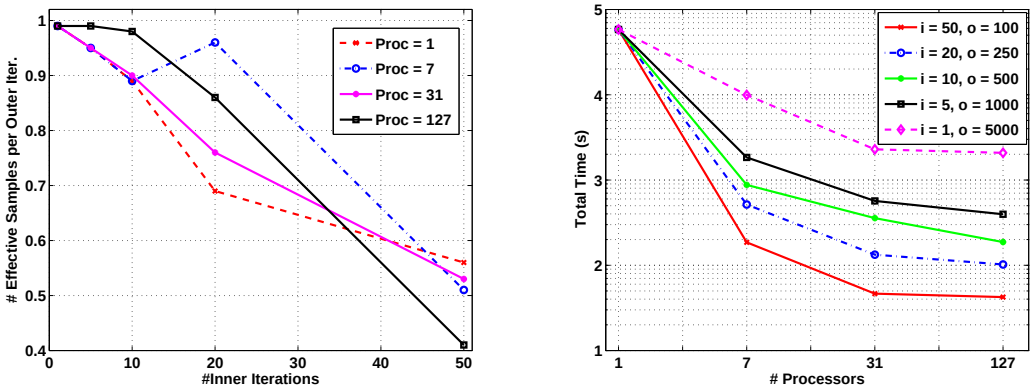


Figure 3.3: Effects of changing the number of inner iterations on: (a) The effective sample size (b) Total running time (Gibbs and Message passing).

is an MCMC monitoring technique which indicates the loss in efficiency of the Markov chain. We defer to [Robert and Casella, 2004] for further details. With increasing number of inner iterations we find the effective sample size rapidly decreases, implying poor mixing of the Markov chain. For example, with $P = 31$ processors and 20 inner iterations the effective sample size has dropped to 0.75 *per outer iteration*. So per *inner* iteration the algorithm is $20/0.75 \approx 27$ times less efficient than the serial sampler.

Figure 3.3(b) shows the total running time for various combinations of inner and outer iterations. Each combination of inner and outer iterations is set so that the total number of iterations is 5000. The Gibbs time per outer iteration, is the maximum time taken to complete the Gibbs (inner) iterations by the set of processors. The total time is the sum of these Gibbs times and the message passing time. Using the maximum is important since time spent waiting for the slowest processor to complete is the limitation of the synchronous scheme.

As the number of processors becomes large, the cost of message passing becomes a limiting factor. This is an artifact of simulating the parallel system, which overestimates the message passing time. In the true parallel system, the cost of message passing is negligible, as discussed in Section 3.4. While time is saved using a larger number of inner iterations, this is at the expense of the sampling efficiency. Using a small number of inner iterations is thus preferred, and we use 3 inner iterations in our experiments.

Table 3.1: Dataset descriptions

Dataset	N	D	Description
AR Faces [Mart'inez and Kak, 2001]	2600	1598	faces with lighting, accessories (real-valued)
Piano [Poliner and Ellis, 2007]	57931	161	STDFT of a piano recording (real-valued)
Flickr [Kollar and Roy, 2009]	646724	1000	indicators of image tags (binary-valued)

Table 3.2: Loglikelihoods on test data for real-world datasets for the serial, synchronous and asynchronous inference types..

Dataset	Serial $p = 1$	Sync $p = 16$	Async $p = 16$
AR Faces	-4.74	-4.77	-4.84
Piano	–	-1.182	-1.228
Flickr	—	-0.0584	

3.4 Realworld Experiments

We tested our parallel scheme on three real world datasets on a 16 node cluster using Matlab Distributed Computing Engine. Implementing this was one of my main contributions to the project. The first dataset was a set of 2600 frontal face images with 1598 dimensions [Mart'inez and Kak, 2001]. While not extremely large, the high-dimensionality of the dataset makes it challenging for other inference approaches. The piano dataset [Poliner and Ellis, 2007] consisted of 57931 samples from a 161-dimensional short-time discrete Fourier transform of a piano piece. Finally, the binary-valued Flickr dataset [Kollar and Roy, 2009] indicated whether each of 1000 popular keywords occurred in the tags of 100,000 images from Flickr. Table 3.1 summarises the data.

Figure 3.4 shows a timing analysis of the faces and music datasets. The Gibbs time per iteration is improved almost linearly as we increase the number of processors, giving for example, a speedup of 14 times in the case of

music for $p = 16$. The message passing time is negligible and is 7% of the Gibbs time for the faces data and 0.1% of the Gibbs time for the music data for $p = 16$ parallel processors. However, waiting for synchronisation becomes significant which motivates motivates the asynchronous inference.

Table 3.2 shows the performance of the parallel inference procedures, and shows that the performance is comparable across the various methods. Figure 3.4(c) compares the times for running inference serially, synchronously and asynchronously with $P = 16$. We find that the asynchronous inference is 1.64 times faster than the synchronous case, reducing the computational time from 11.8s per iteration to 7.2s.

3.5 Discussion

We demonstrated an effective algorithm for data parallel inference in the IBP. The attraction of our algorithm is that it is able to approximate the cutting edge accelerated sampler of Doshi-Velez and Ghahramani [2009], which is non-trivial. Since the messages required for the linear Gaussian model are quite compact, being $0(K^2+KD)$, the belief propagation phase is not computationally expensive and probably should be performed every iteration given the rapid decrease in effective sample size when increased inner iterations are used. As well as providing a significant speed increase, splitting the data over multiple compute nodes also greatly reduces the memory requirements for a large dataset, which could become a limiting factor otherwise.

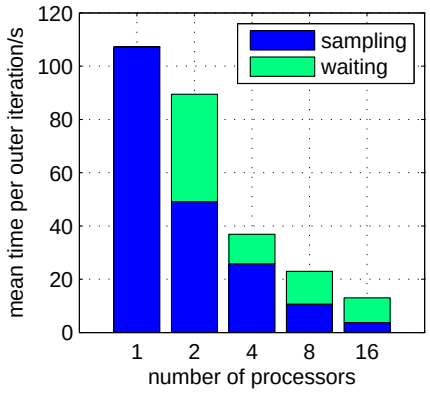
One limitation of our experimental setup was that the 16 nodes used were actually two computers with eight cores each. The large waiting times for synchronous we saw could be an artifact of this: if some shared resources such as the L1 or L2 cache were "hogged" by one processors others would run more slowly. The greedy processor would then have to wait for the others to catch up. On any cluster similar effects could occur however, for example as load varies with other jobs being submitted or completed.

There are issues with the asynchronous sampler. Firstly, its implementation using the Matlab Distributed Computing Engine (MDCE) is far from straightforward. The MDCE has no mechanism for "queuing" messages, so a processor cannot send a message without it being received. Thus each processor must constantly check during its sampling computation whether any neighbours are trying to send messages to it. Secondly, the rate at which information is propagated through the tree of processors is reduced. In the

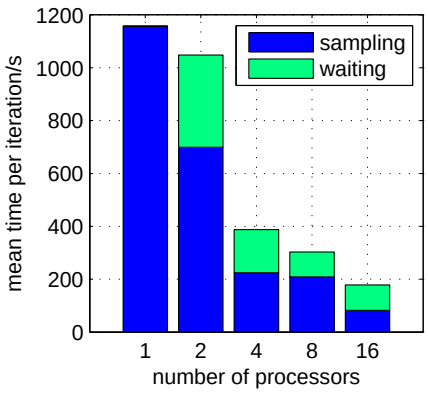
synchronous case, information every processor reaches every other processor every time the message passing is performed. In the asynchronous case, every time a processor completes its inner sampling loop it only propagates information along one edge. Thus in a tree of depth T it will take $2T$ outer iterations for information to be exchanged between the further leaves. Thirdly, the restriction of only deleting globally unused features is no longer possible because the processors do not know the global counts for the Z matrix. Thus the features may become misaligned on different processors. This manifests itself in the "jerky" progress of the log joint for the asynchronous algorithm seen in Figure 3.4(c).

There are ways these problems could be addressed. To allow synchronous message passing with asynchronous sampling all processors could be given a time window in which to sample, for example, one second. Some processors might complete more inner iterations in this time, improving mixing, but would not have to wait long for slower processors. Global synchronous message passing would propagate information throughout the processor tree more efficiently and allow only globally unused features to be deleted.

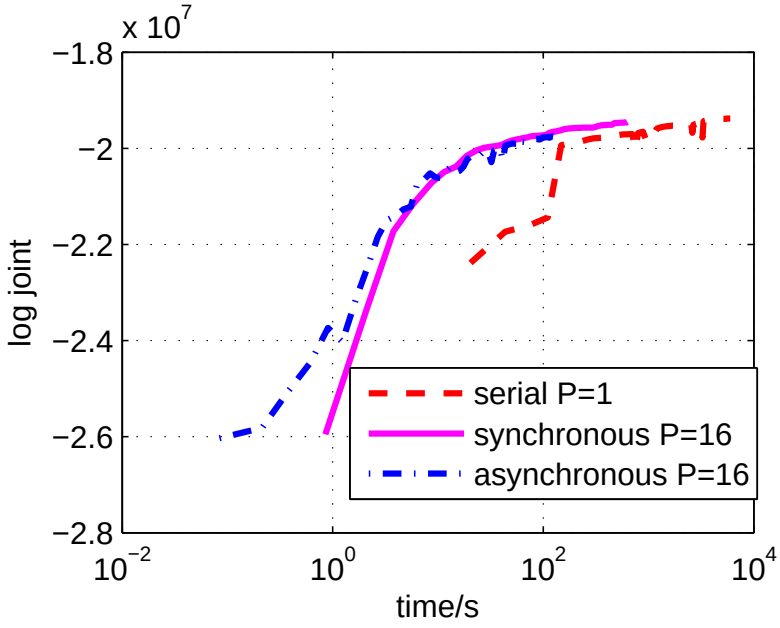
Unfortunately it is not straightforward to extend the accelerated sampler, and therefore our parallel inference scheme, to the non-parametric sparse Factor Analysis model. Although the parameter posterior is Gaussian given the mixing matrix G, sampling elements of Z is more complicated because of the need to integrate out the corresponding real-valued element of G. Doing that approximately would be interesting future work.



(a) Timing analysis for faces dataset



(b) Timing analysis for music dataset



(c) Timing comparison for different approaches

Figure 3.4: Bar charts comparing sampling time and waiting times for synchronous parallel inference.

Chapter 4

Plan for the future

One of the attractions of Machine Learning is its position on the interface between theory and application. I hope that my PhD will continue to incorporate both aspects, and to emphasise this I have broken down my proposed future work into theory and application projects.

4.1 Theory: Non-conjugate VMP

As part of an internship at Microsoft Research with Tom Minka I have been working on extending Variational Message Passing (VMP) [Winn et al., 2005] to non-conjugate-exponential family factors, within the Infer.NET software framework. Our method involves calculating an exponential family distribution approximation to the factor which ensures that the gradient of the Kullback-Leibler divergence with respect to the variational parameters is the same as for the true factor. This allows VMP to be applied in a much wider class of models than was previously possible. I would be interested in continuing this work, for example to fitting variational approximations to non-parametric priors such as the Pitman-Yor process.

4.2 Theory: Hierarchical scale mixtures of Dirichlets

Using the non-conjugate VMP method described in Section 4.1 I have been able to implement a factor representing a Dirichlet prior. In the variational

approximation the mean of the distribution is represented by another Dirichlet distribution, and the precision (the "total count") parameter by a Gamma distribution. Since the mean is Dirichlet distributed, a hierarchical prior can be constructed. This would allow power to be shared across clusters or samples.

4.3 Theory: Nonparametric message passing

Projection VMP allows us to find the closest exponential family distribution in terms of KL divergence to a particular factor. Stick breaking priors such as the Dirichlet Process and Pitman Yor process that form the basis of many non-parametric models, can equally be incorporated into this framework. The method would find the closest truncated Dirichlet in KL-divergence to the true infinite dimensional posterior. Incorporating these features into Infer.NET has the advantage of allowing them to be easily combined with different model components. For example, once a Dirichlet Process factor is implemented, this would allow DP mixture models, Hierachical DP models, and Indian Buffet Process models to be defined and infered.

4.4 Application: A probabilistic model of skin conditions

Bayesian modeling offers the ability to model rich, structured data, coping with missing data, taking into account prior domain knowledge and uncertainty in measurements. I have started a collaboration with the Twin's Research Department at King's College London. For 15 years they have been collecting a vast resource of clinical data on around 10,000 female twins [Spector, December 2006]. The 6000 recorded phenotypes available cover a whole range of conditions, but since my collaborator is a clinical skin doctor, we are planning to focus on this area.

For a subset of 800 individuals, gene expression measurements have been made for three different tissue types - skin, muscle and white blood cells, and these individuals have also been genotyped on high density chips. My focus is currently on phenotypic modeling, where I aim to model various skin conditions incorporating medical prior knowledge. In such a large, complex dataset, effective visualisation and data exploration is invaluable, which I am

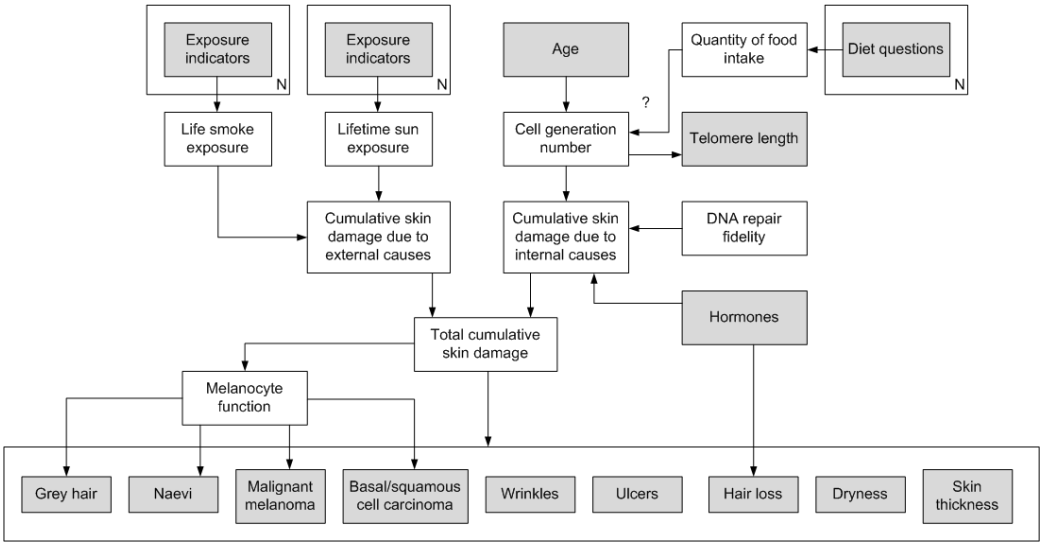


Figure 4.1: Initial model of skin and aging.

starting to achieve extending John Winn's "Vizual" program, which loads data, calculates mutual information between variables, and shows these connections as a graph. A particular challenge of this dataset is the longitudinal nature: different individuals have visited the department varying numbers of times and at varying intervals. At each visit, although some clinical phenotypes are always measured, others are not, or slightly different measurements of the same underlying condition are made. Thus one can imagine a modeling hierarchy: at the bottom disparate but related measurements are combined into more robust latent factors which are highly informative of a "true" attribute of an individual, e.g. how much they sunbathe, whether they have an elevated mole count. At the top level, these variables are combined using medical understanding to test hypotheses, cluster conditions and find the key variation in the data. Figure 4.1 shows an initial outline of the ageing processes in skin which we aim to model.

Term	4	5	6	7	8	9	10
Publish projection VMP work							
Message passing tutorial paper							
Hierachical scale mixtures of Dirichlets							
A probabilistic model of skin conditions							
Combining phenotype, genotype and GE							
Nonparametric message passing							
Thesis writing							

Figure 4.2: Plan for remainder of the period of study

4.5 Application: Correlating genotype, gene expression and phenotype

The UK twins database is one of the first times phenotypic, gene expression and genetic data has been available on the such a large, coherent cohort. In collaboration with Leopold Parts at the Wellcome Trust Sanger Centre I intend to work on finding correlated signals across these different data types. We aim to develop expressive models of the individual domains: rich phenotype models incorporating medical understanding, sparse Factor Analysis models of underlying processes in gene expression, and Hidden Markov Models [Huang et al., 2007] or full ancestral recombination graphs [Minichiello and Durbin, 2006] of genotype variance. We hope that that our power to detect true correlation between these domains will be increased by using these domain specific models. In fact, previous work has shown that removing unobserved environmental and experimental factors from gene expression data before correlating to genotype significantly increases power to detect eQTLs [Stegle et al., 2008].

4.6 Tutorial paper: Message passing algorithms

I plan to work on a tutorial paper with Jurgen van Gael and Philipp Hennig on message passing algorithms. Both Variational Message Passing [Winn et al., 2005] and Expectation Propagation [Minka, 2001] are well developed deterministic algorithms for Bayesian inference. In Minka [2005] alpha-

divergences were used to put both VMP and EP into a common framework. Various developments have extended the functionality of these algorithms: gates for representing arbitrary mixture models [Minka and Winn, 2008], and power plates for calculating partition functions [Qi et al., 2005]. This literature is quite daunting to the non-expert, being highly technical in nature and generally not geared for a practioner's needs. The various concepts are also explained best in disparate papers. These factors motivate a tutorial paper explaining VMP and EP, drawing the link between the two, and explaining the basic concepts required to use these methods in an applied setting.

4.7 Journal Paper on Accelerated Sampling

We plan to write a journal paper with Finale and Shakir combining the accelerated sampler and parallel sampling work, applying both to the nsFA model.

Bibliography

C. Archambeau and F. Bach. Sparse probabilistic projections. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 73–80, Vancouver, Canada, 2009. MIT Press.

Arthur Asuncion, Padhraic Smyth, and Max Welling. Asynchronous distributed learning of topic models. In *Advances in Neural Information Processing Systems 21*, 2008.

Christopher M. Bishop. Bayesian PCA. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 382–388, Cambridge, MA, USA, 1999. MIT Press. ISBN 0-262-11245-0.

Carlos M. Carvalho, Nicholas G. Polson, and James G. Scott. Handling sparsity via the horseshoe. *Journal of Machine Learning Research*, 136: 2144–2162, 2009.

Ali Taylan Cemgil, Cedric Fevotte, and Simon J. Godsill. Blind separation of sparse sources using variational EM. In *Proc. 13th European Signal Processing Conference (EUSIPCO05)*, 2005.

C.T. Chu, S.K. Kim, Y.A. Lin, Y.Y. Yu, G. Bradski, A.Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *Advances in Neural Information Processing Systems*, page 281. MIT Press, 2007.

F. Doshi-Velez and Z. Ghahramani. Accelerated inference for the Indian buffet process. In *Proceedings of the International Conference on Machine Learning*, 2009.

F. Doshi-Velez, K. T. Miller, J. Van Gael, and Y. W. Teh. Variational inference for the Indian buffet process. In *Proceedings of the Intl. Conf. on Artificial Intelligence and Statistics*, volume 12, pages 137–144, 2009.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

C. Fevotte and S.J. Godsill. A Bayesian approach for blind separation of sparse sources. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(6):2174–2188, Nov. 2006. ISSN 1558-7916. doi: 10.1109/TSA.2005.858523.

Ernest Fokoue. Stochastic determination of the intrinsic structure in bayesian factor analysis. Technical report, Statistical and Applied Mathematical Sciences Institute, 2004.

John Geweke. Bayesian treatment of the independent student-t linear model. *Journal of Applied Econometrics*, 8:19–40, 1993.

Z. Ghahramani, T.L. Griffiths, and P. Sollich. Bayesian nonparametric latent feature models. In *Bayesian Statistics 8*. Oxford University Press, 2007.

Zoubin Ghahramani and Matthew J. Beal. Propagation algorithms for variational Bayesian learning. In *In Advances in Neural Information Processing Systems 13*, pages 507–513. MIT Press, 2001.

T. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. Technical Report 1, Gatsby Computational Neuroscience Unit, 2005. URL `citeseer.ist.psu.edu/article/griffiths05infinite.html`.

H Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24:417–441, 1933.

Jim C. Huang, Anitha Kannan, and John Winn. Bayesian association of haplotypes and non-genetic factors to regulatory and phenotypic variation in human populations. *Bioinformatics*, 23(13):i212–221, 2007. doi: 10.1093/bioinformatics/btm217.

Ishwaran and J. Sunil Rao. Js: Spike and slab variable selection: frequentist and bayesian strategies. *Annals of Statistics*, 2003.

Hemant Ishwaran and J. Sunil Rao. Spike and slab gene selection for multigroup microarray data. *J. Amer. Statist. Assoc*, 100:764–780, 2005.

W James and C Stein. Estimation with quadratic loss. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statisitics and Probability*, volume 1, pages 361–379. University of California Press, 1961.

Katy C Kao, Young-Lyeol Yang, Riccardo Boscolo, Chiara Sabatti, Vwani Roychowdhury, and James C Liao. Transcriptome-based determination of multiple transcription regulator activities in escherichia coli by using network component analysis. *Proc Natl Acad Sci U S A*, 101(2):641–646, Jan 2004. doi: 10.1073/pnas.0305287101. URL `http://dx.doi.org/10.1073/pnas.0305287101`.

David Knowles and Zoubin Ghahramani. Infinite sparse factor analysis and infinite independent components analysis. In *7th International Conference on Independent Component Analysis and Signal Separation*, pages 381–388, 2007.

Thomas Kollar and Nick Roy. Utilizing object-object and object-scene context when planning to find things. In *International Conference on Robotics and Automation*, 2009.

D J C Mackay. Bayesian nonlinear modeling for the prediction competition. *ASHRAE Transactions*, 100:1053–1062, 1994.

Aleix M. Mart'inez and Avinash C. Kak. PCA versus LDA. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23:228–233, 2001.

Edward Meeds, Zoubin Ghahramani, Radford Neal, and Sam Roweis. Modeling dyadic data with binary latent factors. In *Neural Information Processing Systems*, volume 19, 2006. URL `http://www.cs.toronto.edu/ roweis/publications.html`.

Mark J. Minichiello and Richard Durbin. Mapping trait loci by use of inferred ancestral recombination graphs. *The American Journal of Human Genetics*, 79(5):910 – 922, 2006. ISSN 0002-9297. doi: DOI: 10.1086/508901.

Thomas P. Minka. Automatic choice of dimensionality for PCA. Technical report, MIT Media Lab, 2000.

Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-800-1.

Tom Minka. Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research, 2005.

Tom Minka and John Winn. Gates: A graphical notation for mixture models. Technical Report MSR-TR-2008-185, Microsoft Research, 2008.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2:559–572, 1901.

Graham E. Poliner and Daniel P. W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP J. Appl. Signal Process.*, 2007(1): 154–154, 2007.

Yuan Alan Qi, Martin Szummer, and Thomas P. Minka. Bayesian conditional random fields. In *Ninth International Conference on Artificial Intelligence and Statistics (AISTATS-2005)*, 2005.

Piyush Rai and Hal Daumé III. The infinite hierarchical factor regression model. In *Neural Information Processing Systems*, Vancouver, Canada, 2008. URL `http://pub.hal3.name/daume08ihfrm`.

C. R. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, second edition, 2004.

Sam Roweis. Em algorithms for pca and spca. In *in Advances in Neural Information Processing Systems*, pages 626–632. MIT Press, 1998.

Tim D. Spector. The uk adult twin registry (TwinsUK). *Twin Research and Human Genetics*, 9:899–906(8), December 2006.

Oliver Stegle, Anitha Kannan, Richard Durbin, and John Winn. Accounting for non-genetic factors improves the power of eqtl studies. *Research in Computational Molecular Biology*, pages 411–422, 2008.

Y. W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. In *Proceedings of the Intl. Conf. on Artificial Intelligence and Statistics*, volume 11, pages 556–563, 2007.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(3):611–622, 1999. ISSN 13697412. URL http://www.jstor.org/stable/2680726.

M Wainwright and M I Jordan. Graphical models, exponential families and variational inference. Technical report, Department of Statistics, UC Berkeley, 2003.

Mike West, Jeffrey Chang, Joe Lucas, Joseph R Nevins, Quanli Wang, and Carlos Carvalho. High-dimensional sparse factor modelling: Applications in gene expression genomics. Technical report, ISDS, Duke University, 2007. URL http://ftp.stat.duke.edu/WorkingPapers/05-15.html.

John Winn, Christopher M. Bishop, and Tommi Jaakkola. Variational message passing. *Journal of Machine Learning Research*, 6:661–694, 2005.

Frank Wood and Thomas L. Griffiths. Particle filtering for nonparametric Bayesian matrix factorization. In *Advances in Neural Information Processing Systems*, volume 19, pages 1513–1520, 2007.

Gale Young. Maximum likelihood estimation and factor analysis. *Psychometrika*, 6(1):49–53, February 1941. URL http://ideas.repec.org/a/spr/psycho/v6y1941i1p49-53.html.

Yan Ping Yu, Douglas Landsittel, Ling Jing, Joel Nelson, Baoguo Ren, Lijun Liu, Courtney McDonald, Ryan Thomas, Rajiv Dhir, Sydney Finkelstein, George Michalopoulos, Michael Becich, and Jian-Hua Luo. Gene expression alterations in prostate cancer predicting tumor aggression and preceding development of malignancy. *J Clin Oncol*, 22(14):2790–2799, Jul 2004. doi: 10.1200/JCO.2004.05.158. URL http://dx.doi.org/10.1200/JCO.2004.05.158.