LEARNING LATENT STRUCTURES VIA BAYESIAN NONPARAMETRICS : NEW MODELS AND EFFICIENT INFERENCE

by

Piyush Rai

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

School of Computing

The University of Utah

August 2013

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Piyush Rai**

has been approved by the following supervisory committee members:

Hal Daume III	, Chair	**August 16, 2012** Date Approved
Jordan Boyd-Graber	, Member	**January 29, 2013** Date Approved
Lawrence Carin	, Member	**January 29, 2013** Date Approved
Thomas P. Fletcher	, Member	**August 16, 2012** Date Approved
Suresh Venkatasubramanian	, Member	**August 16, 2012** Date Approved

and by **Alan L. Davis**, Chair of

the Department of **School of Computing**

and by Donna M. White, Interim Dean of The Graduate School.

ABSTRACT

Latent structures play a vital role in many data analysis tasks. By providing compact yet expressive representations, such structures can offer useful insights into the complex and high-dimensional datasets encountered in domains such as computational biology, computer vision, natural language processing, etc. Specifying the right complexity of these latent structures for a given problem is an important modeling decision. Instead of using models with an *a priori* fixed complexity, it is desirable to have models that can adapt their complexity *as the data warrant*. Nonparametric Bayesian models are motivated precisely based on this desideratum by offering a flexible modeling paradigm for data without limiting the model-complexity *a priori*. The flexibility comes from the model's ability to adjust its complexity adaptively with data.

This dissertation is about nonparametric Bayesian learning of two specific types of latent structures: (1) low-dimensional *latent features* underlying high-dimensional observed data where the latent features could exhibit interdependencies, and (2) *latent task structures* that capture how a set of learning tasks relate with each other, a notion critical in the paradigm of Multitask Learning where the goal is to solve multiple learning tasks jointly in order to borrow information across similar tasks.

Another focus of this dissertation is on designing efficient approximate inference algorithms for nonparametric Bayesian models. Specifically, for the nonparametric Bayesian *latent feature model* where the goal is to infer the binary-valued latent feature assignment matrix for a given set of observations, the dissertation proposes two approximate inference methods. The first one is a search-based algorithm to find the *maximum-a-posteriori* (MAP) solution for the latent feature assignment matrix. The second one is a sequential Monte-Carlo-based approximate inference algorithm that allows processing the data one-example-at-a-time while being space-efficient in terms of the storage required to represent the posterior distribution of the latent feature assignment matrix.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I will forever be thankful to Hal Daume for being all I could ask for in an adviser. Hal provided me with just the right amount of ”hand-holding” in my initial days when I was totally new to machine learning, had faith in me, and then gave me independence and flexibility while I started forging my own path. Hal is incredible in so many ways. The way he balances technical discussions with the right amount of technical rigor and intuition is a rare quality. I will always strive to match the high standards he has set both as a researcher and a teacher.

I thank Suresh Venkatasubramanian for being a great source of motivation, for his infectious enthusiasm about research, for the constant encouragement, and useful advice. Suresh has the amazing ability of seeing the connections between concepts from seemingly different topics, and discussions with him always brought useful insights. I thank Jordan Boyd-Graber, Larry Carin, and Tom Fletcher for serving on my PhD committee and providing helpful feedback. I thank Scott DuVall at the VA Healthcare System for involving me in various projects in his group and helping me contribute. I thank Sneha Kasera for the encouragement and also involving me in the project on perimeter distinction. I thank Matthias Seeger for having me as a summer intern where I learned a great deal about variational methods and compressive sensing. I also thank Jeff Phillips - although I did not have many interactions with him, I always learned something whenever we did meet. I am thankful to Ellen Riloff for sheltering me in the NLP lab for all these years. Many thanks go to the excellent School of Computing staff, especially Ann Carlstrom and Karen Feinauer for all the help with the paperwork, logistics, and beyond.

I must thank all my co-authors - Amit Goyal, Jiarong Jiang, Abhishek Kumar, Amrish Kapoor, Alexandre Passos, Avishek Saha, Anusua Trivedi, and Junxing Zhang - for collaborating with me on various projects. I thank Abhishek for various brainstorming sessions on phone and skype. Interactions with him always led to useful insights on many problems.

I thank Avishek for his constant volley of questions and discussions which resulted in various research ideas. It was great collaborating with Alexandre. Discussions with him were always insightful. I admire Jiarong for her perseverance. I also thank members of the NLP/ML/Data group at Utah - Nathan, Ruihong, Lalindra, Ashequl, Siddharth Patwardhan, David Price, Sean Igo, Jagadeesh, Arvind, John Moeller, and Parasaran - for their company and various discussions. I thank the students of the Machine Learning class I taught in Fall 2011. Interactions with them gave me new perspectives into teaching (thanks to Ross Whitaker for giving me the opportunity to teach this class).

My friends in Utah made my stay here truly worthwhile. Words would never be enough to express what I feel for Niladrish and Anusua. I will always cherish their constant care and support, and all the fun and wonderful times spent with them. I was also extremely lucky to have as roommates (at different points in time) Sachin Goyal, Naveen Muralimanohar, Avishek Saha, Soumya Kar, and (possibly my last roommate) Arijit Banerjee. I thank all of them for their caring ways, friendship, and their great company. I thank Swetha for the friendship, and for the exquisite use of her artistic sensibilities in drawing my avatars. Thanks to Protonu Basu for his visits to my house and *making* me make tea for him.

Finally, this thesis would never have come into being without the love and support of my parents and family. I dedicate this thesis to my mother and father whose strength through the struggles and hardships of life kept me motivated and driven. They may not be eloquent with their wishes but I know they must be very happy to see me reach this point. Special thanks to Madhavi for her love and distractions.

CHAPTER 1

INTRODUCTION

The ubiquity of complex and high-dimensional datasets is presenting ever-increasing challenges in modern-day data analysis problems. More and more application domains are nowadays witnessing the phenomena of data-deluge: advances in microarray technology have made it feasible to acquire high-throughput gene-expression measurements; the explosion of the world-wide-web has led to the creation of text and other multimedia collections of enormous scales; prevalence of networks of various types (social networks, coauthorship networks, etc.) has generated huge amounts of data about the social-personal preferences of people; and so on.

Translating this wealth of information into useful knowledge is not always easy and often requires uncovering and understanding the latent structures that underlie these data. A natural but principled way of accomplishing this is to come up with a statistical model of the data generation process in terms of these underlying latent structures, and explaining the data in terms of these structure. Such an explanation of the data can help in uncovering the complex relationships underlying the data and, by providing a succinct and rich representation, can also help in dealing with problems resulting from the noisy and high-dimensional nature of the data.

A key question is how to model these latent structures. Probabilistic modeling (Bishop, 2006), by its virtue of providing a flexible and natural generative model of the data, is an appealing way of modeling the data. In particular, taking a Bayesian approach to probabilistic modeling allows incorporating prior knowledge about these structures and gives a principled and coherent way of performing inference in the model. This is done by specifying a *prior distribution* on the model parameters and using the Bayes rule to compute the *posterior distribution* of the model parameters given data.

Complexity control is an important issue while specifying any model. Bayesian methods provide an elegant way of accomplishing this by endowing each model parameter with a prior distribution which (implicitly) acts as a regularizer. However, specifying the right level of complexity remains a challenge. Parametric prior distributions assume a fixed model complexity that is independent of the data. This is undesirable since fixing the model complexity *a priori* before even seeing the data seems unnatural. Ideally, it is desirable to have models that are flexible enough to adjust their complexity as warranted by the data.

Nonparametric Bayesian methods (Gershman and Blei, 2012) are designed precisely with this motivation. These methods provide a flexible modeling paradigm for data without restricting the model complexity *a priori*. This flexibility is desired as it avoids the need for doing model-selection, which is both a time-consuming and error-prone process. Moreover, nonparametric methods allow the model complexity to adapt itself as more and more data are observed. This flexibility is desired as the model can "create parameters" to explain the data as and when the data warrant it. This is more appropriate than having a model with a predefined model with a *fixed* complexity that does not depend on data.

This thesis focuses on developing new nonparametric Bayesian models for learning latent structures, and designing efficient inference methods for these models. Specifically, two types of latent structures are considered in this thesis: (1) low-dimensional latent factors underlying high-dimensional data, with the additional property that the latent factors are not independent of each other but are related via an *a priori* unknown structure, and (2) latent task structures capturing how a set of multiple learning tasks (e.g., classification or regression) relate to each other, and leveraging this task structure for sharing information across multiple tasks in order to improve learning. This paradigm is commonly known as learning to learn (Heskes, 2000) or Multitask Learning (Caruana, 1997).

Efficient inference in nonparametric Bayesian models remains an open problem. To this end, this thesis presents two efficient inference methods for the Indian Buffet Process (Ghahramani et al., 2007), which is a nonparametric Bayesian *latent feature model*. In particular, for the nonparametric Bayesian latent feature model, which posits each observation as being generated by a small (and *a priori* unknown) number of latent features, the thesis presents two inference methods: (1) a search-based *maximum-a-posteriori* (MAP) inference method, and (2) a Sequential-Monte-Carlo-based fully Bayesian infer-

ence method that allows processing one observation at a time, while maintaining a compact approximation of the posterior distribution.

1.1 Overview of Methods and Contributions

Here, we give a brief overview of the methods developed as part of this thesis. In particular, the thesis can be divided into three parts: (1) designing nonparametric Bayesian latent factor models for high-dimensional data, (2) designing nonparametric Bayesian models for capturing and leveraging the latent relatedness structure for jointly solving multiple learning tasks, and (3) designing efficient inference methods for nonparametric Bayesian latent feature models.

1.1.1 Nonparametric Bayesian Latent Factor Models

The first contribution of this thesis is a nonparametric Bayesian Factor Analysis model with the following key properties: (1) the number of latent factors need not be known, (2) the latent factors are not assumed to be independent of each other, and (3) not all observed features in the data are considered relevant for the Factor Analysis task. In particular, (2) is of particular interest in many problems. For example, in gene-expression analysis where the factors correspond to biological pathways, the pathways are known to be related with each other. In topic-modeling-based text analysis, factors correspond to topics and the topics tend to be related with each other (by varying degrees); see Figure 1.1 for a pictorial illustration. Having a Factor Analysis model that captures the dependencies among the factors is therefore desirable. Our model also naturally extends for the task of *factor regression* (West, 2003), which involves simultaneous learning of latent factors and predicting the responses associated with each sample, given a set of training samples with their responses.

The nonparametric latent factor model (Ghahramani et al., 2007) has a limitation that it can only learn latent factors underlying a single feature representation of the objects. Often, however, objects are associated with multiple feature representations. For example, a given collection of webpages can be represented using different types of features such as the page-text, the anchor-text on hyperlinks pointed towards them, the images appearing in them, the social tags associated with them, and so on. For such cases, the thesis presents a nonparametric Bayesian Canonical Correlation Analysis (CCA) model that allows learning

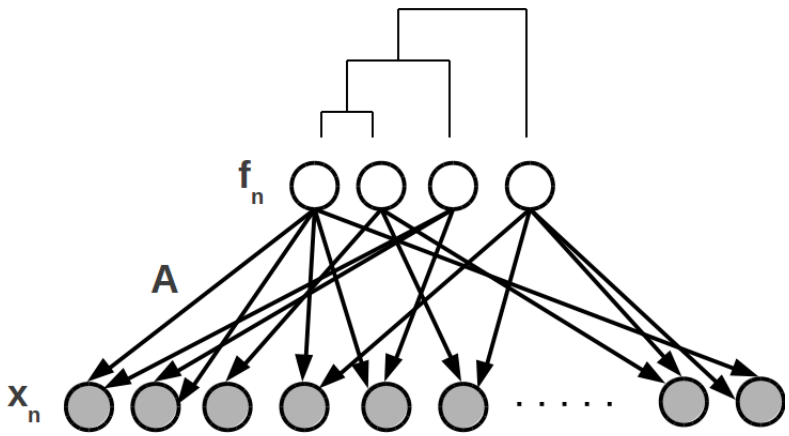


Figure 1.1. Factor Analysis with relationship among latent factors. $\boldsymbol{x}_n$ is a high dimensional observation, $\boldsymbol{f}_n$ are the low-dimensional latent factors, and $\boldsymbol{A}$ is the factor-loading matrix.

latent factors shared across multiple feature representations (or modalities). Another useful application of such a model is for the problem of multilabel prediction using CCA where one modality is the features in each example and the other modality is the responses/labels associated with each example, and the role of CCA is to perform a response/label-guided latent feature extraction. These latent features can then be used with a supervised learner.

1.1.2 Nonparametric Bayesian Learning of Latent Task Structures

The second contribution of this thesis is designing efficient inference methods for the nonparametric latent feature model (Ghahramani et al., 2007), a general, nonparametric Bayesian framework for inferring how a set of learning problems relate to each other, and leveraging this knowledge to *jointly* solve these problems. This problem setting is commonly known as Multitask Learning (Caruana, 1997). Multitask Learning critically relies on the assumption of how different tasks relate with each other. An incorrect assumption not supported by the dataset can end up hurting the performance. It is therefore desirable to have a Multitask Learning model that, instead of having an *a priori* fixed notion of task relatedness, can *adapt* its assumption based on the data. With this motivation in mind, the thesis presents a generative model of the task parameters (e.g., the weight vectors of a linear classification/regression model) assuming that the task parameters of multiple tasks are drawn from a *shared* Mixture of Factor Analyzers (MFA) model (Ghahramani and Hinton, 1997). By giving a nonparametric Bayesian treatment, the resulting model achieves

considerable modeling flexibility and is shown to subsume several previously proposed Multitask Learning models as its special cases, while being more flexible and robust than these models.

1.1.3 Efficient Inference for the Nonparametric Latent Feature Models

The third contribution of this thesis is designing efficient inference methods for nonparametric Bayesian methods, in particular, for the Indian Buffet Process (IBP), which is a nonparametric latent feature model (Ghahramani et al., 2007).

To this end, the thesis develops two approximate inference methods

- The first method is a beam-search-based approximate *maximum-a-posteriori* (MAP) inference method for the IBP. This method is motivated by the fact that in many practical cases, we do not require the full posterior distribution of the latent feature assignment matrix but only seek the best, highest probability sample from the posterior distribution. In such cases, a fast method that can provide the MAP estimate may be more desirable than sampling-based methods such as MCMC, or optimization-based methods such as variational inference that explore the full posterior distribution, and are therefore usually slow.

- The second method is a Sequential-Monte-Carlo-based inference method which provides samples from the full posterior distribution and has the appealing property that it can process the observations in an online manner (i.e., one observation at a time). This is desirable both for scalability as well as for many practical scenarios where observations arrive one at a time. Moreover, our proposed method is an improvement over the existing SMC-based method for the IBP as it allows incorporating the most recent observation in the inference. The earlier proposed SMC method for the IBP ignores the most recent observation. We show that our proposed method leads to improved inference quality as well as considerably more succinct representation of the posterior distribution as compared to the standard SMC-based inference for the IBP (Wood and Griffiths, 2007).

1.1.4 Thesis Statement

Nonparametric Bayesian methods, combined with efficient inference strategies, can provide flexible ways to design models that can (a) learn low-dimensional latent features from high-dimensional data, (b) infer *relatedness* of these latent features, and (c) solve multiple related learning problems by inferring latent *shared predictive structures*.

1.2 Thesis Organization

The rest of the chapters of the thesis are organized as follows:

Chapter 2 provides a brief background on the models and concepts on which the subsequent chapters are based. In particular, it talks about nonparametric Bayesian methods such as the Dirichlet Process, the Indian Buffet Process, and the Kingman's Coalescent. In addition, the chapter provides a brief background on Factor Analysis and Multitask Learning.

Chapter 3 describes the nonparametric Bayesian Factor Analysis model. We discuss how the model learns the correct number of latent factors, allows the factors to be related via an *a priori* unknown hierarchy, and filters away noisy features in the data for more robust Factor Analysis.

Chapter 4 describes the multiview generalization of the nonparametric latent factor model. In particular, we describe how we can use the Indian Buffet Process to design a nonparametric Bayesian version of the Canonical Correlation Analysis model.

Chapter 5 describes a nonparametric Bayesian model we propose for Multitask Learning. The model is based on the assumption that the weight vectors of a collection of potentially related tasks live on a low-dimensional subspace. This is equivalent to the weight vectors being generated as a linear combination of a set of *basis tasks*. We describe how taking a Factor Analysis model on the weight vector, Multitask Learning can be accomplished and how using the Indian Buffet Process allows us to circumvent model selection issues in such a model.

Chapter 6 builds on the model described in Chapter 5. We show how replacing the single factor analyzer by a mixture of factor analyzers allows us to capture considerably richer notions of task relatedness and can provide a general framework for modeling task relatedness.

Chapter 7 describes our proposed beam-search algorithm for doing approximate MAP inference for the IBP. By experimental comparisons with other state-of-the-art methods, we show that this method can be a viable alternative to methods based on sampling or variational inference.

Chapter 8 describes our proposed Sequential-Monte-Carlo-based (SMC) inference method for the IBP, and discusses its differences with the standard SMC-based inference method for the IBP proposed in (Wood and Griffiths, 2007).

Chapter 9 presents a discussion and concludes with some possible directions for future work.

CHAPTER 2

BACKGROUND

This chapter provides a brief background on nonparametric Bayesian methods, in particular the Dirichlet Process, the Indian Buffet Process, and the Kingman's Coalescent, which would be used as building blocks for the models developed in this thesis. The chapter also provides a brief background on Latent Factor Analysis, Mixture of Factor Analyzers, and Multitask Learning for which the proposed models in the thesis have been developed.

2.1 Nonparametric Bayesian Methods

In any data analysis task, choosing the appropriate model complexity is a critical issue. For example, in data clustering, one needs to specify the number of clusters; in dimensionality reduction, one needs to specify the dimensionality of the lower-dimensional space; in regression or classification, one needs to specify the functional form of the input-output relationship, which is typically a parametric model defined by a fixed set of parameters. In all these cases, the number of parameters (number of clusters, dimensionality of the lower-dimensional space, or the number of parameters in the regression/classification model) do not depend on the data and need to be specified *a priori*.

Nonparametric Bayesian methods take an entirely different approach to this problem of model selection. Instead of prespecifying the model complexity *a priori*, these methods assume the model to have an unbounded complexity to begin with and the eventual complexity to be determined by the amount of data. Essentially, these methods can adapt the model complexity by creating parameters as and when dictated by the data. Note that the name *nonparametric* here is somewhat a misnomer. It does not mean that the model does not have any parameters. It means that the number of parameters is potentially infinite but limited by the data. What is important here is that it does not need to be specified *a priori*.

2.2 Dirichlet Process

The Dirichlet Process (DP) is a prior distribution over discrete distributions (Ferguson, 1973). Discreteness implies that if one draws samples from a distribution drawn from the DP, the samples will cluster: new samples take the same value as older samples with some positive probability. A DP is defined by two parameters: a concentration parameter α and a base measure G_0. The sampling process defining the DP draws the first sample from the base measure G_0. Each subsequent sample would take on a new value drawn from G_0 with a probability proportional to α, or reuse a previously drawn value with probability proportional to the number of samples having that value. This property makes it suitable as a prior for effectively infinite mixture models, where the number of mixtures can grow as new samples are observed. Our mixture of factor analyzers-based MTL model uses the DP to model the mixture components so we do not need to specify their number *a priori*.

2.3 Indian Buffet Process

The Indian Buffet Process (IBP) (Ghahramani et al., 2007) is a nonparametric Bayesian prior that defines a distribution over infinite binary matrices. The IBP was originally motivated by the need to model the latent feature structure of a given set of observations. The IBP, due to its flexibility, has been a model of choice in variety of nonparametric Bayesian applications, such as for factorial structure learning, learning causal structures, modeling dyadic data, modeling overlapping clusters, and others (Ghahramani et al., 2007).

In the latent feature model, each observation can be thought of as consisting of a set of latent features. Given an $N \times D$ matrix $\mathbf{X}$ of N observations having D features each, we can consider a decomposition of the form $\mathbf{X} = \mathbf{ZA} + \mathbf{E}$ where $\mathbf{Z}$ is an $N \times K$ binary feature-assignment matrix describing which features are present in each observation. $Z_{n,k}$ is 1 if feature k is present in observation n, and is otherwise 0. $\mathbf{A}$ is a $K \times D$ matrix of feature scores, and the matrix $\mathbf{E}$ consists of observation-specific noise. A crucial issue in such models is the choosing the number K of latent features. The standard formulation of IBP lets us define a prior over the binary matrix $\mathbf{Z}$ such that it can have an unbounded number of columns and thus can be a suitable prior in problems dealing with such structures.

The IBP derivation starts by defining a finite model for K many columns of a $N \times K$ binary matrix.

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

Here $m_k = \sum_i Z_{ik}$. In the limiting case, as $K \to \infty$, it as was shown in (Ghahramani et al., 2007) that the binary matrix **Z** generated by IBP is equivalent to one produced by a sequential stochastic process. This process can be best understood by a culinary analogy of customers coming to an Indian restaurant and selecting dishes from an infinite array of dishes. In this analogy, customers represent observations (rows of **X**) and dishes represent latent features (columns of **Z**). Customer 1 selects $Poisson(\alpha)$ dishes to begin with. Thereafter, each incoming customer n selects an existing dish k with a probability m_k/N, where m_k denotes how many previous customers chose that particular dish. The customer n then goes on further to additionally select $Poisson(\alpha/n)$ new dishes. This process generates a binary matrix **Z** with rows representing customer and columns representing dishes. Many real-world datasets have a sparseness property, which means that each observation depends only on a subset of all the K latent features. This means that the binary matrix **Z** is expected to be reasonably sparse for many datasets. This makes IBP a suitable choice for capturing the underlying sparsity in addition to automatically discovering the number of latent features. Figure 2.1 shows a pictorial illustration of the IBP.

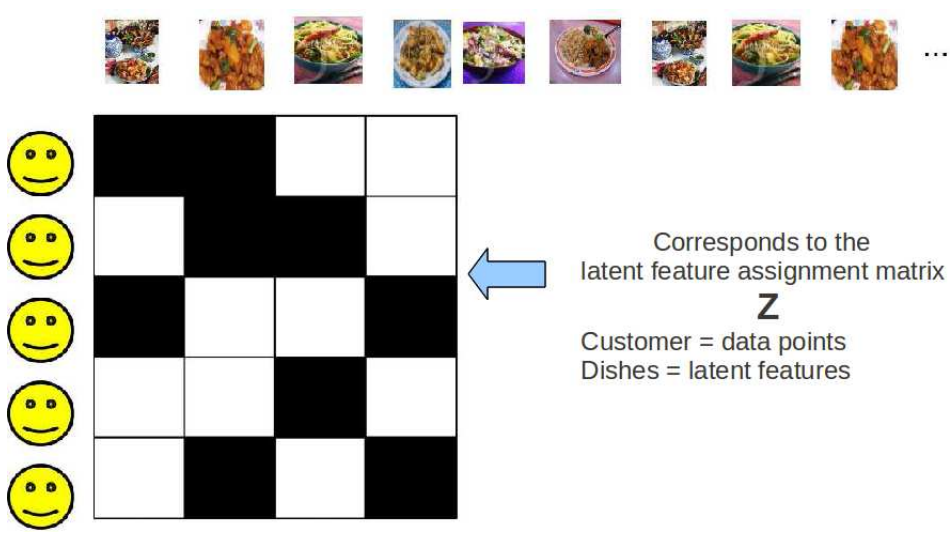


Figure 2.1. Pictorial illustration of the IBP with $N = 4$ and eventual $K = 4$ unique latent features. In the IBP, customers correspond to datapoints and dishes correspond to latent features.

2.4 Kingman's Coalescent

Our model makes use of a latent hierarchical structure over factors; we use Kingman's Coalescent (Kingman, 1982) as a convenient prior distribution over hierarchies. Kingman's Coalescent originated in the study of population genetics for a set of single-parent organisms. The Coalescent is a nonparametric model over a countable set of organisms. It is most easily understood in terms of its finite dimensional marginal distributions over n individuals, in which case it is called an n-coalescent. We then take the limit $n \to \infty$. In our case, the individuals are *factors*.

The n-coalescent considers a population of n organisms at time $t = 0$. We follow the ancestry of these individuals backward in time, where each organism has exactly one parent at time $t < 0$. The n-coalescent is a continuous-time, partition-valued Markov process, which starts with n singleton clusters at time $t = 0$ and evolves *backward*, coalescing lineages until there is only one left. We denote by t_i the *time* at which the ith coalescent event occurs (note $t_i \leq 0$), and $\delta_i = t_{i-1} - t_i$ the time between events (note $\delta_i > 0$). Under the n-coalescent, each pair of lineages merges independently with exponential rate 1; so $\delta_i \sim \mathcal{E}xp\left(\binom{n-i+1}{2}\right)$. With probability one, a random draw from the n-coalescent is a binary tree with a single root at $t = -\infty$ and n individuals at time $t = 0$. We denote the tree structure by π. The marginal distribution over tree topologies is uniform and independent of coalescent times; and the model is infinitely exchangeable. We therefore consider the limit as $n \to \infty$, called *the coalescent.* See Figure 2.2 for a pictorial illustration.

Once the tree structure is obtained, one can define an additional Markov process to evolve over the tree. One common choice is a Brownian diffusion process. In Brownian diffusion in D dimensions, we assume an underlying diffusion covariance of $\mathbf{\Lambda} \in \mathbb{R}^{D \times D}$ p.s.d. The root is a D-dimensional vector drawn $\boldsymbol{z}$. Each nonroot node in the tree is drawn Gaussian with mean equal to the value of the parent, and variance $\delta_i \mathbf{\Lambda}$, where δ_i is the time that has passed.

Recently, Teh et al. (Teh et al., 2008) proposed efficient bottom-up agglomerative inference algorithms for the coalescent. These (approximately) maximize the probability of π and δs, marginalizing out internal nodes by Belief Propagation. If we associate with each node in the tree a *mean* $\boldsymbol{y}$ and *variance* $\boldsymbol{v}$ message, we update messages as Eq (2.1), where i is the current node and li and ri are its children.

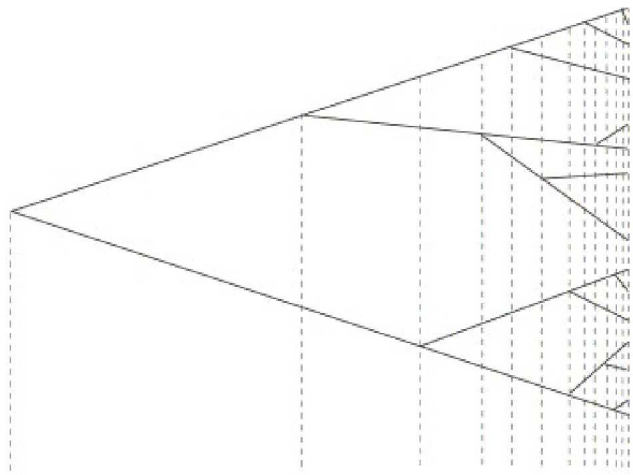


Figure 2.2. Pictorial illustration of an n-coalescent with $n = 15$ individuals

$$\boldsymbol{v}_i = \left[(\boldsymbol{v}_{li} + (t_{li} - t_i)\boldsymbol{\Lambda})^{-1} + (\boldsymbol{v}_{ri} + (t_{ri} - t_i)\boldsymbol{\Lambda})^{-1}\right]^{-1} \tag{2.1}$$
$$\boldsymbol{y}_i = \left[\boldsymbol{y}_{li}(\boldsymbol{v}_{li} + (t_{li} - t_i)\boldsymbol{\Lambda})^{-1} + \boldsymbol{y}_{ri}(\boldsymbol{v}_{ri} + (t_{ri} - t_i)\boldsymbol{\Lambda})^{-1}\right]^{-1} \boldsymbol{v}_i$$

2.5 Factor Analysis

Factor Analysis (Bartholomew and Knott, 1999) is the task of explaining data by means of a small set of latent factors. One of the first applications of Factor Analysis can be found in the psychology community in an attempt to explain intelligence using a small set of latent traits or factors. More formally, given a set of observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, Factor Analysis attempts to explain each observation $\boldsymbol{x}_n \in \mathbb{R}^D$ using a smaller number of latent factors $\boldsymbol{f}_n \in \mathbb{R}^K$ ($K \ll D$) as follows:

$$\boldsymbol{x}_n = \boldsymbol{A}\boldsymbol{f}_n + \varepsilon_n$$

where $\boldsymbol{A}$ denotes the *factor loading matrix* of size $D \times K$ and ε_n denotes the observation-specific noise (typically assumed to be Gaussian) not explained by the latent factors. Figure 2.3 shows a pictorial illustration of a standard Factor Analysis model.

2.6 Mixture of Factor Analyzers

A mixture of factor analyzers (MFA) model generalizes the standard Factor Analysis model by assuming that for each observation, first we select a factor analyzer from a collection of factor analyzers and then generate the observation using *that* factor analyzer.

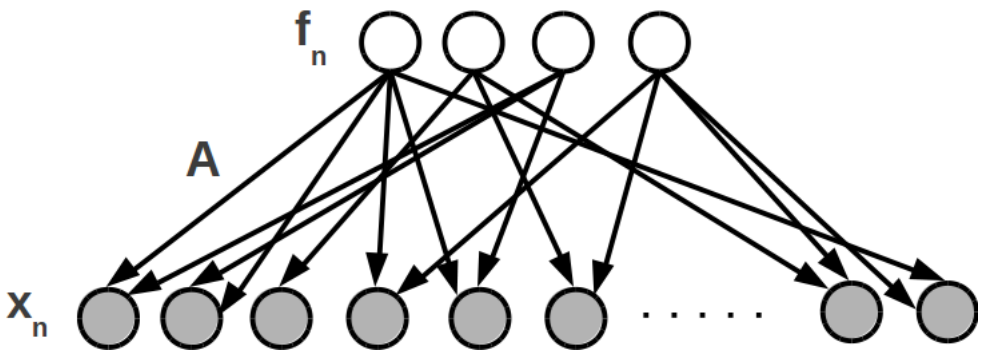


Figure 2.3. A basic Factor Analysis model. $\boldsymbol{x}_n$ is a high-dimensional observation, $\boldsymbol{f}_n$ are the low-dimensional latent factors, $\boldsymbol{A}$ is the factor-loading matrix.

Suppose $z(n)$ denotes the index of the chosen factor analyzer for the n-th observation $\boldsymbol{x}_n$. The generative story for this observation under the MFA can be written as:

$$\boldsymbol{x}_n = \boldsymbol{\mu}_{z(n)} + \boldsymbol{A}_{z(n)}\boldsymbol{f}_n + \varepsilon_n$$

Note that, unlike the standard Factor Analysis, in an MFA, we also have a mean $\boldsymbol{\mu} \in \mathbb{R}^D$ associated with each factor analyzer. Therefore, each factor analyzer is parameterized by a pair $\{\boldsymbol{\mu}, \boldsymbol{A}\}$ of mean and a factor loading matrix.

An MFA model can also be seen as a local dimensionality reduction method with different local factor analyzers performing dimensionality reduction in different regions of space. Seen another way, an MFA model performs data clustering, while simultaneously performing dimensionality reduction within each cluster. This can be especially useful in clustering high-dimensional data when the number of datapoints is small. Standard clustering methods such as a mixture of Gaussian would be prone to overfitting in such high-dimensional, small sample-size cases because it fits a mixture of *full-rank* Gaussians. On the other hand, an MFA can be seen as fitting a mixture of low-rank Gaussians (note that a factor analyzer is akin to a low-rank Gaussian), thereby preventing overfitting.

2.7 Multitask Learning

Learning problems do not exist in a vacuum. Often, one is tasked with developing not one, but many classifiers for different tasks. In these cases, there is often not enough data to learn a good model for each task individually—real-world examples are prioritizing

email messages across many users' inboxes (Aberdeen et al., 2011) and recommending items to users on web sites (Ning and Karypis, 2010). In these settings it is advantageous to transfer or share information across tasks. Multitask Learning (MTL) (Caruana, 1997) encompasses a range of techniques to share statistical strength across models for various tasks and allows learning even when the amount of labeled data for each individual task is very small. Most MTL methods achieve this improved performance by assuming some notion of similarity across tasks. For example:

- Parameters of all the tasks are close to a shared *mean* parameter. Probabilistically, this is equivalent to the parameters of all the tasks being drawn from a shared Gaussian distribution (Chelba and Acero, 2006).

- Parameters of all the tasks exhibit a clustering structure (Jacob and Bach, 2008, Xue et al., 2007b). Probabilistically, this is equivalent to the parameters of all the tasks being drawn from a mixture of Gaussian distributions.

- Parameters of all the tasks live on a low-dimensional subspace (Rai and Daumé III, 2010), or all the tasks have a common set of relevant features (Argyriou et al., 2007).

- Task relationships can be captured by modeling the task covariance matrix (Bonilla et al., 2007, Zhang and Yeung, 2010).

Choosing the model whose task similarity assumptions are consistent for the given Multitask Learning problem is critical. Incorrect assumptions, however, can end up degrading the performance.

CHAPTER 3

NONPARAMETRIC BAYESIAN SPARSE LATENT FACTOR MODEL

In this chapter, we describe our proposed nonparametric Bayesian Factor Analysis model that simultaneously learns the number of factors as well as the relationships among the factors. Moreover, our method also allows simultaneously doing feature selection so that only relevant features in the data participate in Factor Analysis.

3.1 Introduction

Factor Analysis is the task of explaining data by means of a set of *latent factors*. Factor *regression* couples this analysis with a prediction task, where the predictions are made solely on the basis of the factor representation. The latent factor representation achieves two-fold benefits: (1) discovering the latent *process* underlying the data; (2) simpler predictive modeling through a compact data representation. In particular, (2) is motivated by the problem of prediction in the *"large P small N"* paradigm (West, 2003), where the number of features P greatly exceeds the number of examples N, potentially resulting in overfitting.

We address three fundamental shortcomings of standard Factor Analysis approaches (Beal et al., 2005, Sabatti and James, 2005, Sanguinetti et al., 2006, West, 2003): (1) we do not assume a known number of factors; (2) we do not assume factors are independent; (3) we do not assume all features are relevant to the Factor Analysis. Our motivation for this work stems from the task of reconstructing regulatory structure from gene-expression data. In this context, factors correspond to regulatory pathways. Our contributions thus parallel the needs of gene pathway modeling. In addition, we couple predictive modeling (for factor regression) within the Factor Analysis framework itself, instead of having to

model it separately.

Our factor regression model is fundamentally nonparametric. In particular, we treat the gene-to-factor relationship nonparametrically by proposing a sparse variant of the Indian Buffet Process (IBP) (Ghahramani et al., 2007), designed to account for the sparsity of relevant genes (features). We *couple* this IBP with a hierarchical prior over the factors. This prior explains the fact that pathways are fundamentally related: some are involved in transcription, some in signaling, some in synthesis. The nonparametric nature of our sparse IBP requires that the hierarchical prior *also* be nonparametric. A natural choice is Kingman's coalescent (Kingman, 1982), a popular distribution over infinite binary trees.

Since our motivation is an application in bioinformatics, our notation and terminology will be drawn from that area. In particular, *genes* are *features*, *samples* are *examples*, and *pathways* are *factors*. However, our model is more general. An alternative application might be to a collaborative filtering problem, in which case our genes might correspond to movies, our samples might correspond to users, and our pathways might correspond to genres. In this context, all three contributions of our model still make sense: we do not know how many movie genres there are; some genres are closely related (romance to comedy versus to action); many movies may be spurious.

Our model uses a variant of the Indian Buffet Process (Section 2.3) to model the feature-factor (i.e., gene-pathway) relationships. We further use Kingman's Coalescent (Section 2.4) to model latent pathway hierarchies.

3.2 Nonparametric Bayesian Factor Regression

Recall the standard Factor Analysis problem: $\mathbf{X} = \mathbf{AF} + \mathbf{E}$, for standardized data $\mathbf{X}$. $\mathbf{X}$ is a $P \times N$ matrix consisting of N samples $[\boldsymbol{x}_1, ..., \boldsymbol{x}_N]$ of P features each. $\mathbf{A}$ is the factor loading matrix of size $P \times K$ and $\mathbf{F} = [\boldsymbol{f}_1, ..., \boldsymbol{f}_N]$ is the factor matrix of size $K \times N$. $\mathbf{E} = [\boldsymbol{e}_1, ..., \boldsymbol{e}_N]$ is the matrix of idiosyncratic variations. K, the number of factors, is known.

Recall that our goal is to treat the Factor Analysis problem nonparametrically, to model feature relevance, and to model hierarchical factors. For expository purposes, it is simplest to deal with each of these issues in turn. In our context, we begin by modeling the gene-factor relationship nonparametrically (using the IBP). Next, we propose a variant of IBP to model gene relevance. We then present the hierarchical model for inferring factor

hierarchies. We conclude with a presentation of the full model and our mechanism for modifying the Factor *Analysis* problem to factor *regression*.

3.2.1 Nonparametric Gene-Factor Model

We begin by directly using the IBP to infer the number of factors. Although IBP has been applied to nonparametric Factor Analysis in the past (Ghahramani et al., 2007), the standard IBP formulation places IBP prior on the factor matrix (**F**), associating *samples* (i.e., a set of features) with factors. Such a model assumes that the sample-factor relationship is sparse. However, this assumption is inappropriate in the gene-expression context where it is not the factors themselves but the *associations* among genes and factors (i.e., the factor loading matrix **A**) that are sparse. In such a context, each sample depends on all the factors, but each gene within a sample usually depends only on a small number of factors.

Thus, it is more appropriate to model the factor loading matrix (**A**) with the IBP prior. Note that since **A** and **F** are related with each other via the number of factors *K*, modeling **A** nonparametrically allows our model to also have an unbounded number of factors.

For most gene-expression problems (West, 2003), a binary factor loadings matrix (**A**) is inappropriate. Therefore, we instead use the Hadamard (element-wise) product of a binary matrix **Z** and a matrix **V** of reals. **Z** and **V** are of the same size as **A**. The Factor Analysis model, for each sample i, thus becomes: $\boldsymbol{x}_i = (\boldsymbol{Z} \odot \boldsymbol{V})\boldsymbol{f}_i + \boldsymbol{e}_i$. We have $\mathbf{Z} \sim \mathcal{IBP}(\alpha, \beta)$. α and β are IBP hyperparameters and have vague gamma priors on them. Our initial model assumes no factor hierarchies and hence the prior over **V** would simply be a Gaussian: $\mathbf{V} \sim \mathcal{N}or(0, \sigma_v^2\mathbf{I})$ with an inverse-gamma prior on σ_v. **F** has a zero mean, unit variance Gaussian prior, as used in standard Factor Analysis. Finally, $\boldsymbol{e}_i = \mathcal{N}or(0, \boldsymbol{\Psi})$ models the idiosyncratic variations of genes where $\boldsymbol{\Psi}$ is a $P \times P$ diagonal matrix ($diag(\Psi_1, ..., \Psi_P)$). Each entry Ψ_P has an inverse-gamma prior on it.

3.2.2 Feature Selection Prior

Typical gene-expression datasets are of the order of several thousands of genes, most of which are *not* associated with any pathway (factor). In the above, these are accounted for only by the idiosyncratic noise term. A more realistic model is that certain genes simply do not participate in the Factor Analysis. In the culinary analogy, some of the genes that enter

the restaurant leave before selecting any dishes. We will refer to such genes as "spurious". We add an additional prior term to account for such spurious genes; effectively leading to a sparse solution (over the rows of the IBP matrix). It is important to note that this notion of sparsity is fundamentally *different* from the conventional notion of sparsity in the IBP. The sparsity in IBP is over *columns*, not *rows*. To see the difference, recall that the IBP contains a "rich get richer" phenomenon: frequently selected factors are more likely to get reselected. Consider a truly spurious gene and ask whether it is likely to select any factors. If some factor k is already frequently used, then *a priori*, this gene is more likely to select it. The only downside to selecting it is the data likelihood. By setting the corresponding value in $\mathbf{V}$ to zero, there is no penalty.

Our sparse-IBP prior is identical to the standard IBP prior with one exception. Each customer (gene) p is associated with Bernoulli random variable T_p that indicates whether it samples *any* dishes. The $\mathbf{T}$ vector is given a parameter ρ, which, in turn, is given a Beta prior with parameters a, b.

3.2.3 Hierarchical Factor Model

In our basic model, each column of the matrix $\boldsymbol{Z}$ (and the corresponding column in $\boldsymbol{V}$) is associated with a factor. These factors are considered unrelated. To model the fact that factors are, in fact, related, we introduce a factor hierarchy. Kingman's coalescent (Kingman, 1982) is an attractive prior for integration with IBP for several reasons. It is nonparametric and describes exchangeable distributions. This means that it can model a varying number of factors. Moreover, efficient inference algorithms exist (Teh et al., 2008).

3.2.4 Full Model and Extension to Factor Regression

Our proposed graphical model is depicted in Figure 3.1. The key aspects of this model are the IBP prior over $\mathbf{Z}$, the sparse binary vector $\mathbf{T}$, and the coalescent prior over $\mathbf{V}$.

In standard Bayesian factor regression (West, 2003), Factor Analysis is followed by the regression task. The regression is performed only on the basis of $\mathbf{F}$, rather than the full data $\mathbf{X}$. For example, a simple linear regression problem would involve estimating a K-dimensional parameter vector $\boldsymbol{\theta}$ with regression value $\boldsymbol{\theta}^\top \mathbf{F}$. Our model, on the other hand, integrates the factor regression component in the nonparametric Factor Analysis framework itself. We do so by prepending the responses y_i to the expression vector $\boldsymbol{x}_i$ and joining the

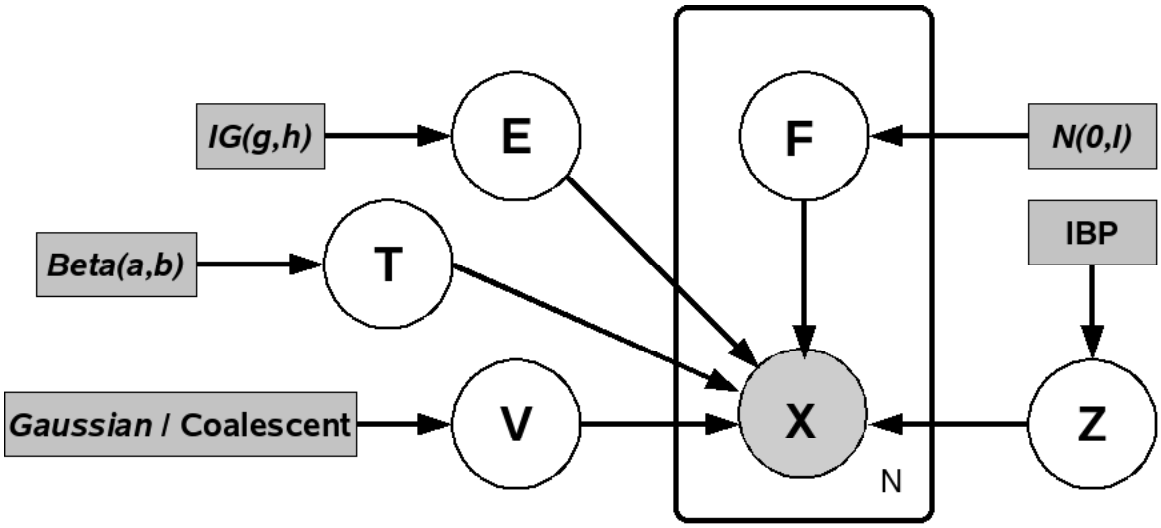


Figure 3.1. The graphical model for nonparametric Bayesian Factor Regression. **X** consists of response variables as well.

training and test data (see Figure 3.2). The unknown responses in the test data are treated as missing variables to be iteratively imputed in our MCMC inference procedure. It is straightforward to see that it is equivalent to fitting another sparse model relating factors to responses. Our model thus allows the Factor Analysis to take into account the regression task as well. In case of binary responses, we add an extra probit regression step to predict binary outcomes from real-valued responses.

3.3 Inference

Exact inference is intractable in our model and, therefore, we use Gibbs sampling with a few Metropolis-Hastings steps to perform approximate inference.

3.3.1 Sampling the IBP Matrix Z

Sampling **Z** consists of sampling existing dishes, proposing new dishes and accepting or rejecting them based on the acceptance ratio in the associated M-H step. For sampling existing dishes, an entry in **Z** is set as 1 according to $p(Z_{ik} = 1|\mathbf{X}, Z_{-ik}, \mathbf{V}, \mathbf{F}, \mathbf{\Psi}) \propto \frac{m_{-i,k}}{(P+\beta-1)} p(\mathbf{X}|\mathbf{Z}, \mathbf{V}, \mathbf{F}, \mathbf{\Psi})$ whereas it is set as 0 according to $p(Z_{ik} = 0|\mathbf{X}, Z_{-ik}, \mathbf{V}, \mathbf{F}, \mathbf{\Psi}) \propto \frac{P+\beta-1-m_{-i,k}}{(P+\beta-1)} p(\mathbf{X}|\mathbf{Z}, \mathbf{V}, \mathbf{F}, \mathbf{\Psi})$. $m_{-i,k} = \sum_{j \neq i} Z_{jk}$ is how many other customers chose dish k.

For sampling new dishes, we use an M-H step where we simultaneously propose $\mathbf{J} =$

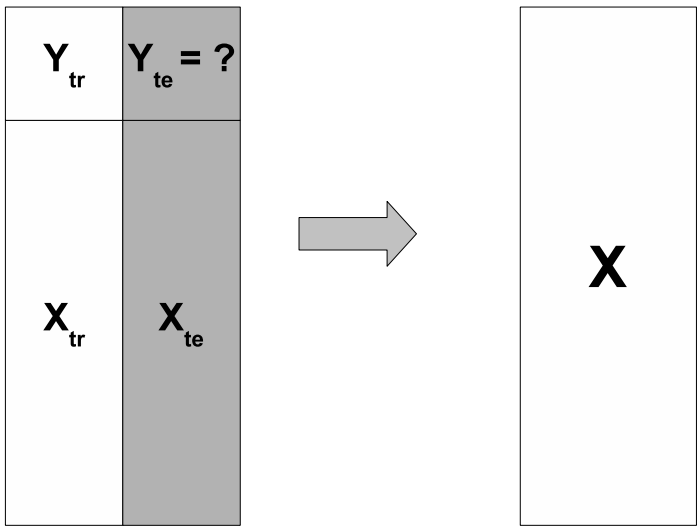


Figure 3.2. Training and test data are combined together and test responses are treated as missing values to be imputed.

$(K^{new}, V^{new}, F^{new})$ where $K^{new} \sim Poisson(\alpha\beta/(\beta + P - 1))$. We accept the proposal with an acceptance probability (following (Meeds et al., 2007)) given by $a = \min\{1, \frac{p(rest|\mathbf{J}^*)}{p(rest|\mathbf{J})}\}$. Here, $p(rest|\eta)$ is the likelihood of the data given parameters η. We propose V^{new} from its prior (either Gaussian or Coalescent) but, for faster mixing, we propose F^{new} from its posterior.

Sampling V^{new} from the coalescent is slightly involved. As shown pictorially in Figure 3.3, proposing a new column of $\boldsymbol{V}$ corresponds to adding a new leaf node to the existing coalescent tree. In particular, we need to find a sibling (s) to the new node y' and need to find an insertion point on the branch joining the sibling s to its parent p (the grandparent of y'). Since the marginal distribution over trees under the coalescent is uniform, the sibling s is chosen uniformly over nodes in the tree. We then use importance sampling to select an insertion time for the new node y' between t_s and t_p, according to the exponential distribution given by the coalescent prior (our proposal distribution is uniform). This gives an insertion point in the tree, which corresponds to the new parent of y'. We denote this new parent by p' and the time of insertion as t. The predictive density of the newly inserted node y' can be obtained by marginalizing the parent p'. This yields $\mathcal{N}or(\boldsymbol{y}_0, \boldsymbol{v}_0)$, given by:

$$\boldsymbol{v}_0 = [(\boldsymbol{v}_s + (t_s - t)\boldsymbol{\Lambda})^{-1} + (\boldsymbol{v}_p + (t - t_p)\boldsymbol{\Lambda})^{-1}]^{-1}$$

$$\boldsymbol{y}_0 = [\boldsymbol{y}_s/(v_s + (t_s - t)\boldsymbol{\Lambda}) + \boldsymbol{y}_p/(v_p + (t_p - t)\boldsymbol{\Lambda})]\boldsymbol{v}_0$$

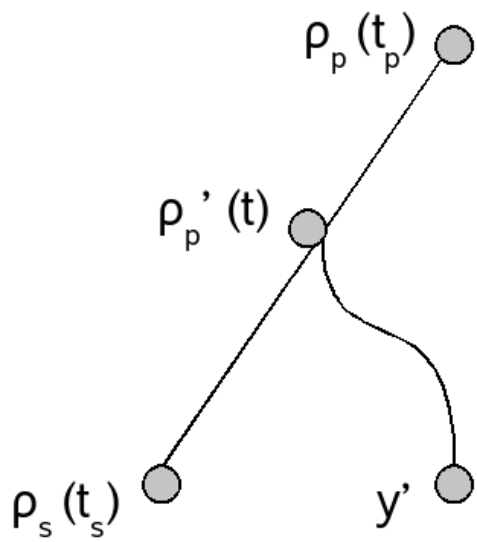


Figure 3.3. Adding a new node to the tree

Here, y_s and v_s are the messages passed *up* through the tree, while y_p and v_p are the messages passed *down* through the tree (compare to Eq (2.1)).

3.3.2 Sampling the Sparse IBP Vector T

In the *sparse IBP prior*, recall that we have an additional P-many variables T_p, indicating whether gene p "eats" any dishes. T_p is drawn from Bernoulli with parameter ρ, which, in turn, is given a $\mathcal{B}et(a, b)$ prior. For inference, we collapse ρ and $\mathbf{\Psi}$ and get Gibbs posterior over T_p of the form $p(T_p = 1|.) \propto (a + \sum_{q \neq p} T_p)\mathcal{S}tu(\boldsymbol{x}_p|(\mathbf{Z}_p \odot \mathbf{V}_p)\boldsymbol{F}, g/h, g))$ and $p(T_p = 0|.) \propto (b + P - \sum_{q \neq p} T_q)\mathcal{S}tu(\boldsymbol{x}_p|0, g/h, g)$, where $\mathcal{S}tu$ is the nonstandard Student's t-distribution. g, h are hyperparameters of the inverse-gamma prior on the entries of $\mathbf{\Psi}$.

3.3.3 Sampling the Real-valued Matrix V

For the case when **V** has a Gaussian prior on it, we sample **V** from its posterior

$$p(V_{g,j}|\mathbf{X}, \mathbf{Z}, \mathbf{F}, \mathbf{\Psi}) \propto \mathcal{N}or(V_{g,j}|\mu_{g,j}, \Sigma_{g,j})$$

where $\Sigma_{g,j} = (\sum_{i=1}^{N} \frac{F_{j,i}^2}{\Psi_g} + \frac{1}{\sigma_v^2})^{-1}$ and $\mu_{g,j} = \Sigma_{g,j}(\sum_{i=1}^{N} F_{j,i} X_{g,j}^*)\Psi_g^{-1}$. We define $X_{g,j}^* = X_{g,i} - \sum_{l=1, l \neq j}^{K} (A_{g,l} V_{g,l}) F_{l,i}$, and $\mathbf{A} = \mathbf{Z} \odot \mathbf{V}$. The hyperparameter σ_v on **V** has an inverse-gamma prior and posterior also has the same form. For the case with coalescent prior on **V**,

we have $\Sigma_{g,j} = (\sum_{i=1}^{N} \frac{F_{j,i}^2}{\Psi_g} + \frac{1}{v_{0j}})^{-1}$ and $\mu_{g,j} = \Sigma_{g,j}(\sum_{i=1}^{N} F_{j,i}X_{g,j}^*)(\Psi_g + \frac{y_{0g,j}}{v_{0j}})^{-1}$, where $\boldsymbol{y}_0$ and $\boldsymbol{v}_0$ are the Gaussian posteriors of the leaf node added in the coalescent tree (see Eq (2.1)), which corresponds to the column of **V** being sampled.

3.3.4 Sampling the Factor Matrix F

We sample for **F** from a normal distribution with mean $^{-} = \mathbf{A^T}(\mathbf{AA^T} + \mathbf{\Psi})^{-1}\mathbf{X}$ and covariance $\mathbf{\Sigma} = \mathbf{I} - \mathbf{A^T}(\mathbf{AA^T} + \mathbf{\Psi})^{-1}\mathbf{A}$, where $\mathbf{A} = \mathbf{Z} \odot \mathbf{V}$

3.3.5 Sampling the Idiosyncratic Noise Term

We place an inverse-gamma prior on the diagonal entries of $\mathbf{\Psi}$ and the posterior too is inverse-gamma: $p(\Psi_p|.) \propto \mathcal{IG}(g + \frac{N}{2}, \frac{h}{1+\frac{h}{2}tr(\mathbf{E}^T\mathbf{E})})$, where $\mathbf{E} = \mathbf{X} - (\mathbf{Z} \odot \mathbf{V})\mathbf{F}$.

3.3.6 Sampling IBP Hyperparameters

We sample the IBP hyperparameter α from its posterior: $p(\alpha|.) \sim \mathcal{G}am(K_+ + a, \frac{b}{1+bH_P(\beta)})$, where K_+ is the number of active features at any moment and $H_P(\beta) = \sum_{i=1}^{P} 1/(\beta + i - 1)$. β is sampled from a prior proposal using an M-H step.

3.3.7 Sampling the Factor Tree

We use the Greedy-Rate1 algorithm (Teh et al., 2008).

3.4 Related Work

A number of probabilistic approaches have been proposed in the past for the problem of gene-regulatory network reconstruction (Beal et al., 2005, Sabatti and James, 2005, Sanguinetti et al., 2006, West, 2003). Some take into account the information on the prior network topology (Sabatti and James, 2005), which is not always available. Most assume the number of factors is known. To get around this, one can perform model selection via Reversible Jump MCMC (Green, 1995) or evolutionary stochastic model search (Carvalho et al., 2008). Unfortunately, these methods are often difficult to design and may take quite long to converge. Moreover, they are difficult to integrate with other forms of prior knowledge (e.g., factor hierarchies). A somewhat similar approach to ours is the infinite independent component analysis (iICA) model of (Knowles and Ghahramani, 2007), which treats Factor Analysis as a special case of ICA. However, their model is limited to Factor Analysis and does not take into account feature selection, factor hierarchy, and factor

regression. As a generalization to the standard ICA model, (Bach and Jordan, 2003) proposed a model in which the components can be related via a tree-structured graphical model. It, however, assumes a fixed number of components.

Structurally, our model with Gaussian-**V** (i.e., no hierarchy over factors) is most similar to the Bayesian Factor Regression Model (BFRM) of (West, 2003). BFRM assumes a sparsity inducing mixture prior on the factor loading matrix **A**. Specifically, $A_{pk} \sim (1 - \pi_{pk})\delta_0(A_{pk}) + \pi_{pk}\mathcal{N}or(A_{pk}|0, \tau_k)$ where $\delta_0()$ is a point mass centered at zero. To complete the model specification, they define $\pi_{pk} \sim (1 - \rho_k)\delta_0(\pi_{pk}) + \rho_k\mathcal{B}et(\pi_{pk}|sr, s(1 - r))$ and $\rho_k \sim \mathcal{B}et(\rho_k|av, a(1 - v))$. Now, integrating out π_{pk} gives: $A_{pk} \sim (1 - v\rho_k)\delta_0(A_{pk}) + v\rho_k\mathcal{N}or(A_{pk}|0, \tau_k)$. It is interesting to note that the nonparametric prior of our model (factor loading matrix defined as $\mathbf{A} = \mathbf{Z} \odot \mathbf{V}$) is actually equivalent to the (parametric) sparse mixture prior of the BFRM as $K \to \infty$. To see this, note that our prior on the factor loading matrix **A** (composed of **Z** having an IBP prior, and **V** having a Gaussian prior), can be written as $A_{pk} \sim (1 - \rho_k)\delta_0(A_{pk}) + \rho_k\mathcal{N}or(A_{pk}|0, \sigma_v^2)$, if we define $\rho_k \sim \mathcal{B}et(1, \alpha\beta/K)$. It is easy to see that, for BFRM where $\rho_k \sim \mathcal{B}et(av, a(1 - v))$, setting $a = 1 + \alpha\beta/K$ and $v = 1 - \alpha\beta/(aK)$ recovers our model in the limiting case when $K \to \infty$.

3.5 Experiments

In this section, we report our results on synthetic and real datasets. We compare our nonparametric approach with the evolutionary-search-based approach proposed in (Carvalho et al., 2008), which is the nonparametric extension to BFRM.

We used the gene-factor connectivity matrix of *e-coli* network (described in (Pournara and Wernisch, 2007)) to generate a synthetic dataset having 100 samples of 50 genes and 8 underlying factors. Since we knew the ground truth for factor loadings in this case, this dataset was ideal to test for efficacy in recovering the factor loadings (binding sites and number of factors). We also experimented with a real gene-expression, breast-cancer dataset having 251 samples of 226 genes and 5 prominent underlying factors (we know this from domain knowledge).

3.5.1 Nonparametric Gene-Factor Modeling and Variable Selection

For the synthetic dataset generated by the *e-coli* network, the results are shown comparing the actual network used to generate the data (Figure 3.4), the inferred factor loading

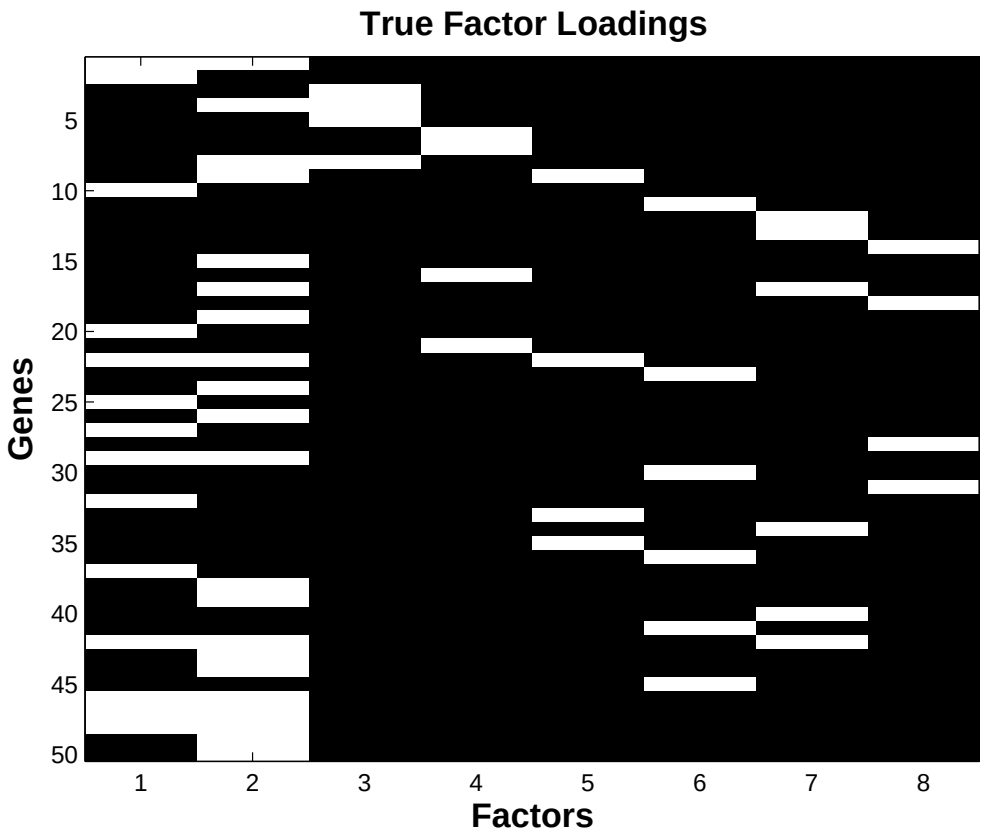


Figure 3.4. True factor loadings for the synthetic data with P=50, K=8 generated using connectivity matrix of *e-coli* data. White rectangles represent active sites. The data also have added noise with signal-to-noise-ratio of 10.

matrix by our method (Figure 3.5), and by BFRM (Figure 3.6). As shown in Figure 3.5, our method recovered exactly the same number (8) of factors, and almost exactly the same factor loadings (binding sites and number of factors) as the ground truth. In comparison, the BFRM based on evolutionary search overestimated the number of factors and the inferred loadings clearly seem to be off from the actual loadings (even modulo column permutations).

Our results on real data are shown in Figure 3.7, Figure 3.8, and Figure 3.9. To see the effect of variable selection for these data, we also introduced spurious genes by adding 50 random features in each sample. We observe the following: (1) Without variable selection being on, spurious genes result in an overestimated number of factors and falsely discovered factor loadings for spurious genes (see Figure 3.7), (2) Variable selection, when

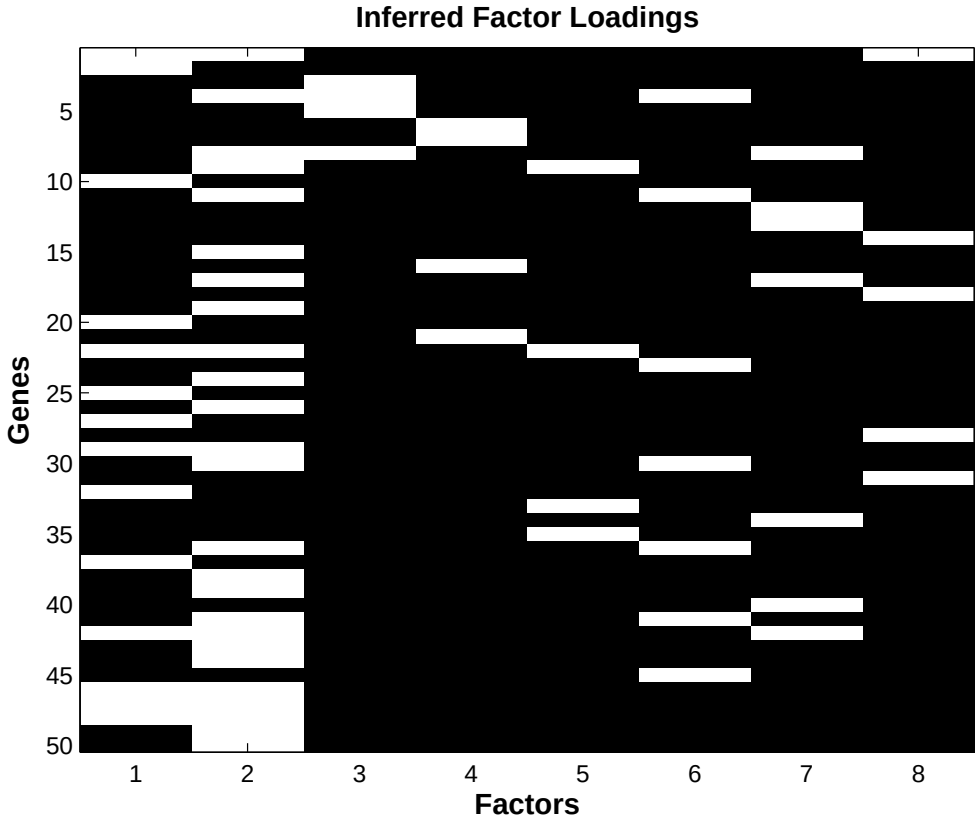


Figure 3.5. Inferred factor loadings (with our approach) for the synthetic data with P=50, K=8 generated using connectivity matrix of *e-coli* data.

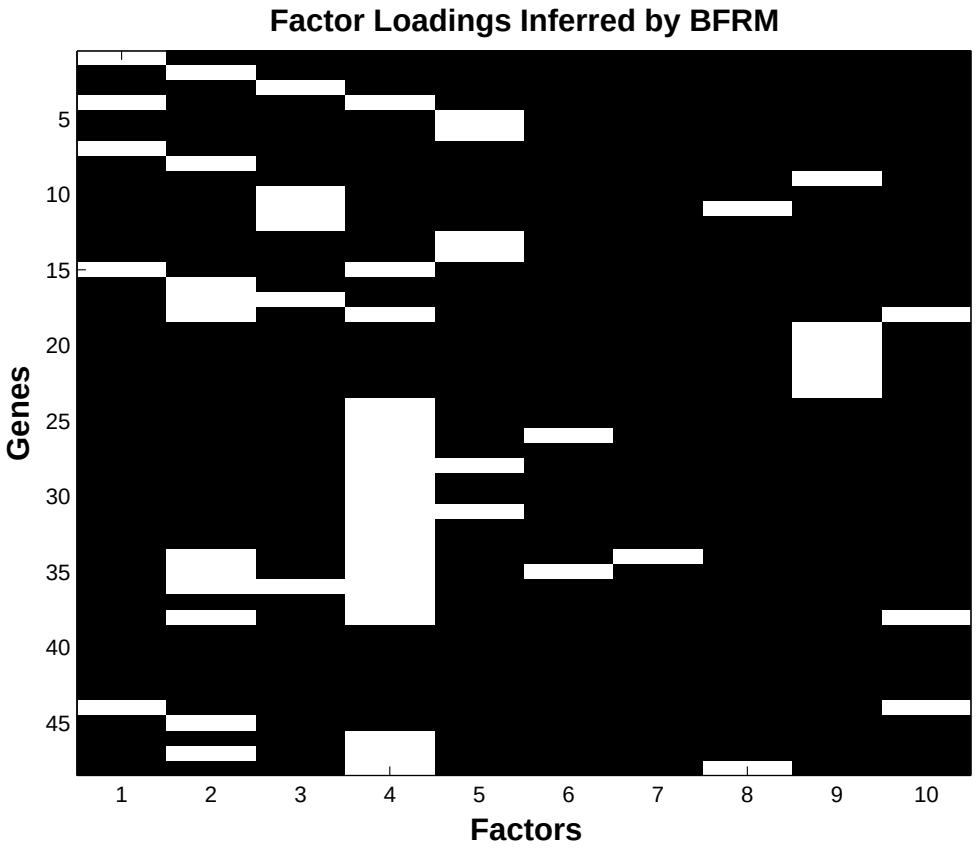


Figure 3.6. Inferred factor loadings with the evolutionary-search-based approach.

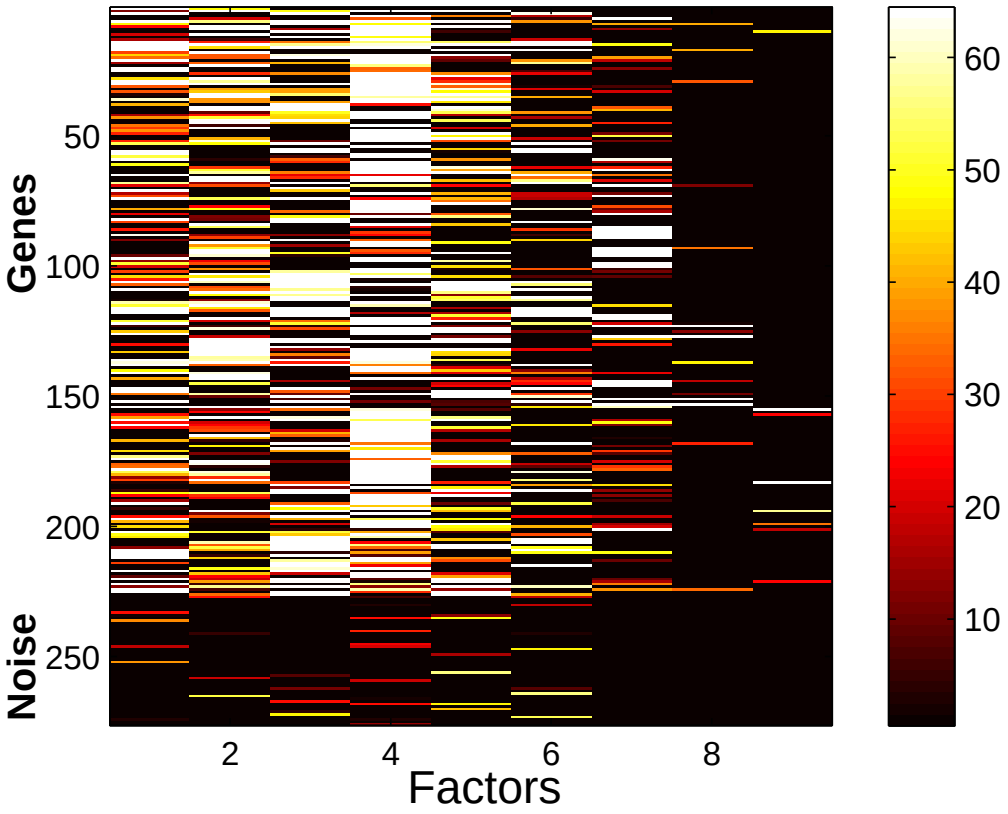


Figure 3.7. IBP based sparse Factor Analysis without feature selection

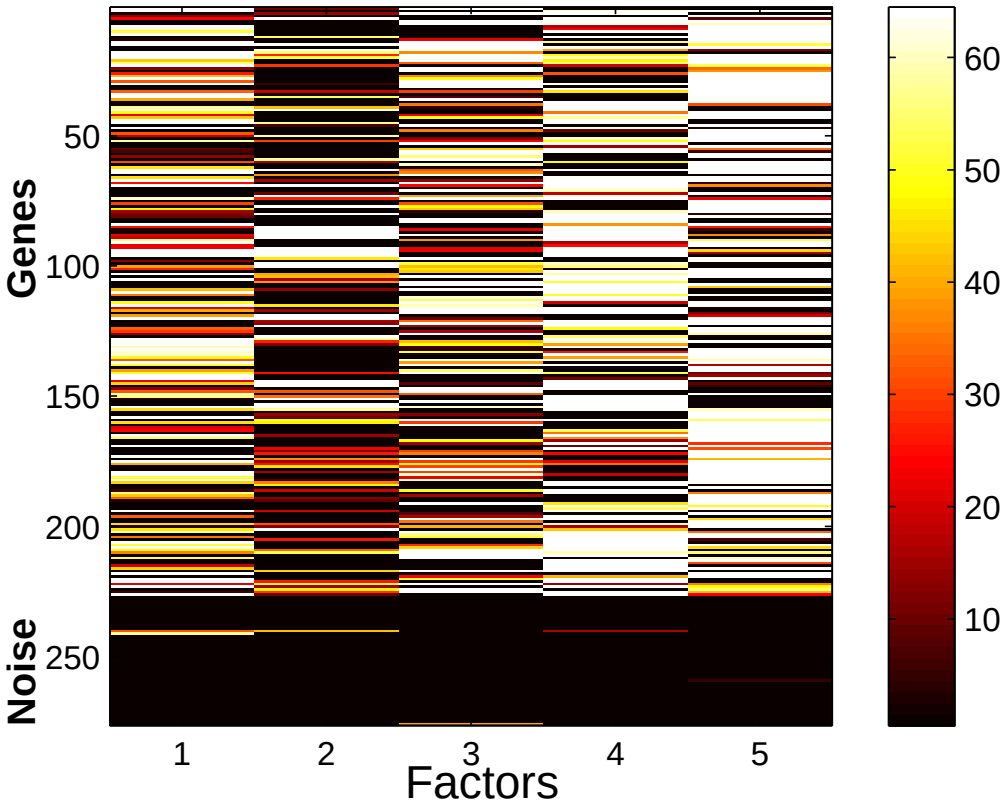


Figure 3.8. IBP based sparse Factor Analysis with variable selection

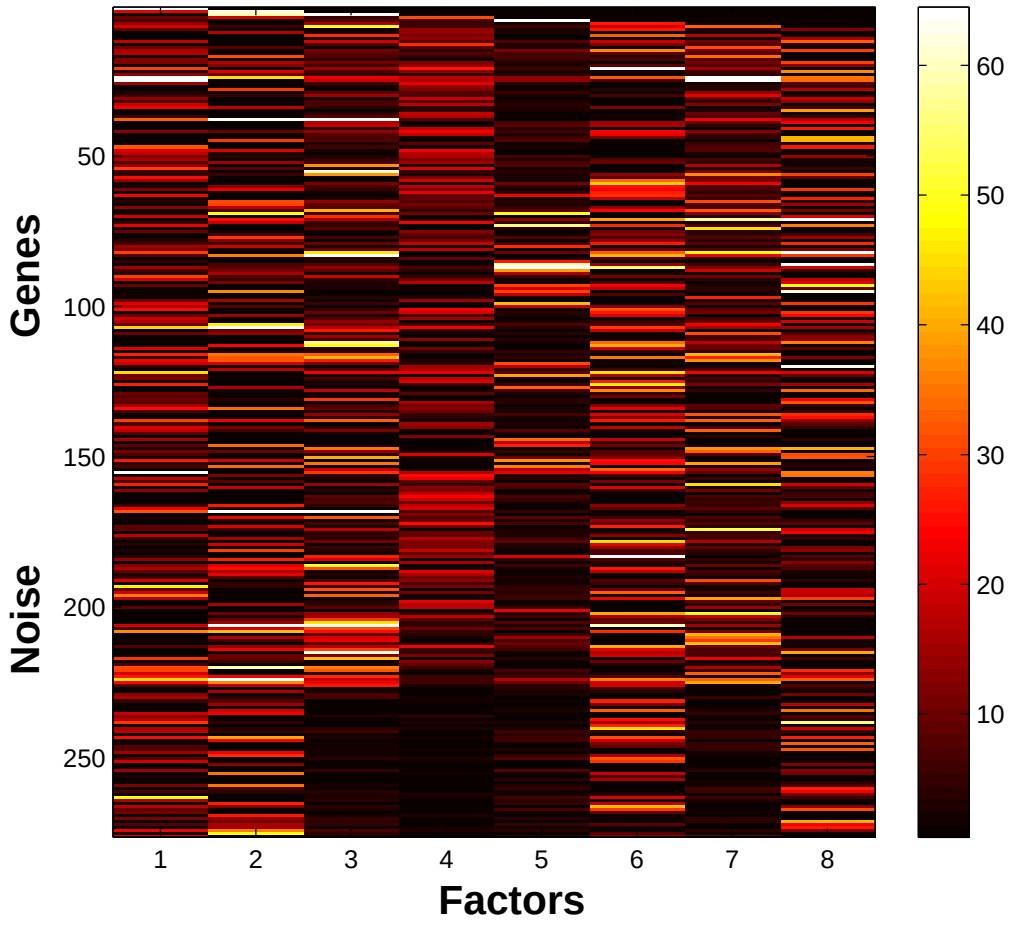
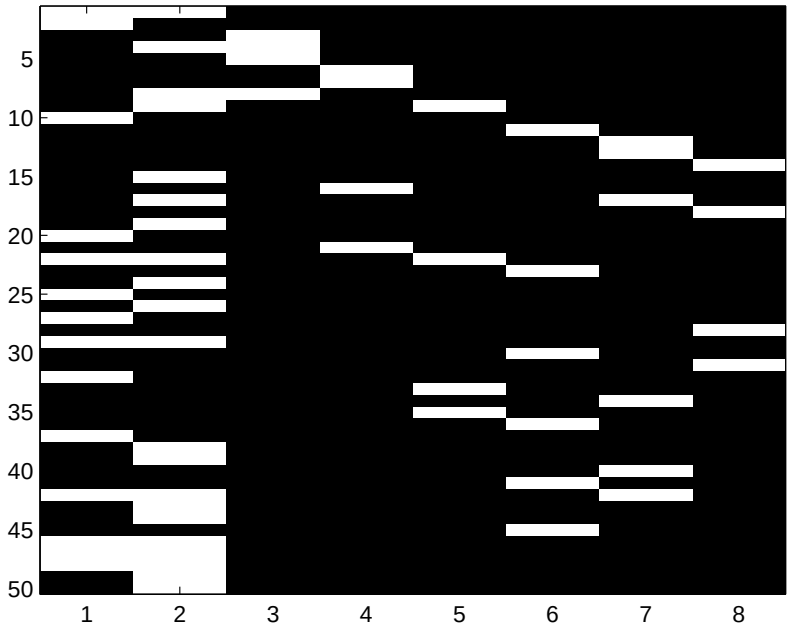


Figure 3.9. Bayesian factor regression model

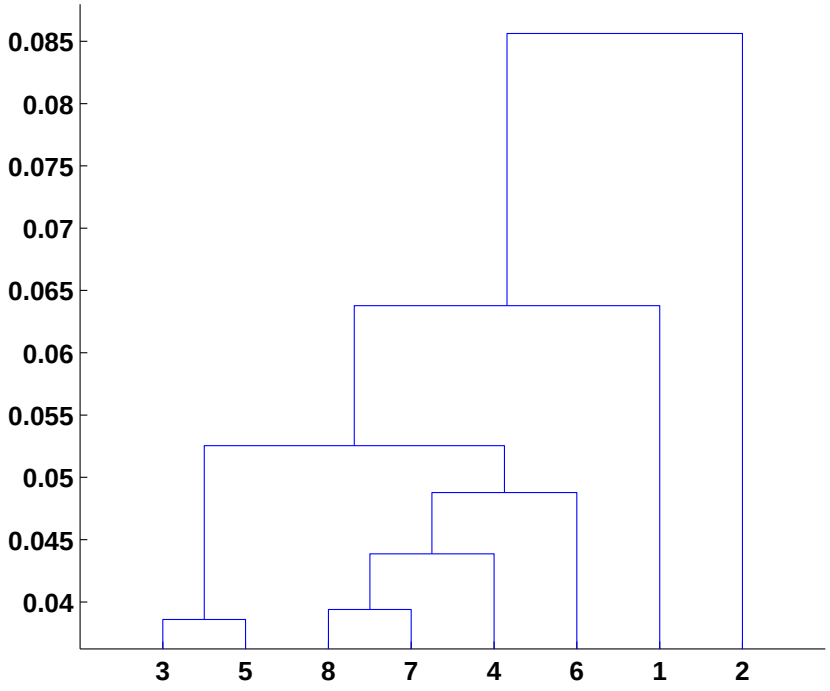
on, effectively filters out spurious genes, without overestimating the number of factors (see Figure 3.8). We also investigated the effect of noise on the evolutionary-search-based approach and it resulted in an overestimated number of factor, plus false discovered factor loadings for spurious genes (see Figure 3.9). To conserve space, we do not show here the cases when there are no spurious genes in the data, but it turns out that variable selection does not filter out any of 226 relevant genes in such a case.

3.5.2 Hierarchical Factor Modeling

Our results with hierarchical factor modeling are shown in Figure 3.10 and Figure 3.11 for synthetic and real data. As shown, the model correctly infers the gene-factor associations, the number of factors, and the factor hierarchy. There are several ways to interpret the hierarchy. From the factor hierarchy for *e-coli* data (Figure 3.10 (b)), we see that column-2 (corresponding to factor-2) of the **V** matrix is the most prominent one (it regulates the highest number of genes), and is closest to the tree-root, followed by column-2, to which it looks most similar. Columns corresponding to lesser prominent factors are located further down in the hierarchy (with appropriate relatedness). Figure 3.11 (b) can be interpreted in a similar manner for breast-cancer data. The hierarchy can be used to find factors in order of their prominence. The higher we chop off the tree along the hierarchy, the more prominent the factors, we discover, are. For instance, if we are only interested in the top 2 factors in *e-coli* data, we can chop off the tree above the sixth coalescent point. This is akin to the agglomerative clustering sense, which is usually done *post-hoc*. In contrast, our model discovers the factor hierarchies as part of the inference procedure itself. At the same time, there is no degradation of data reconstruction (in the mean-squared-error sense) and the log-likelihood, when compared to the case with Gaussian prior on **V** (see Figure 3.12 - they actually *improve*). We also show in Section 3.5.3 that hierarchical modeling results in better predictive performance for the factor regression task. Empirical evidences also suggest that the factor hierarchy leads to faster convergence since most of the unlikely configurations will never be visited as they are constrained by the hierarchy.

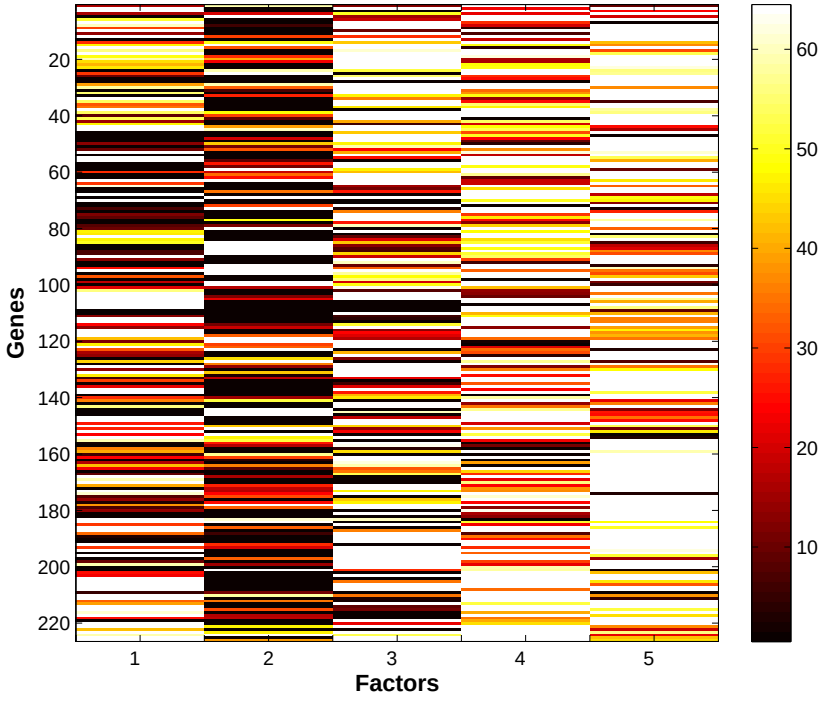


(a)

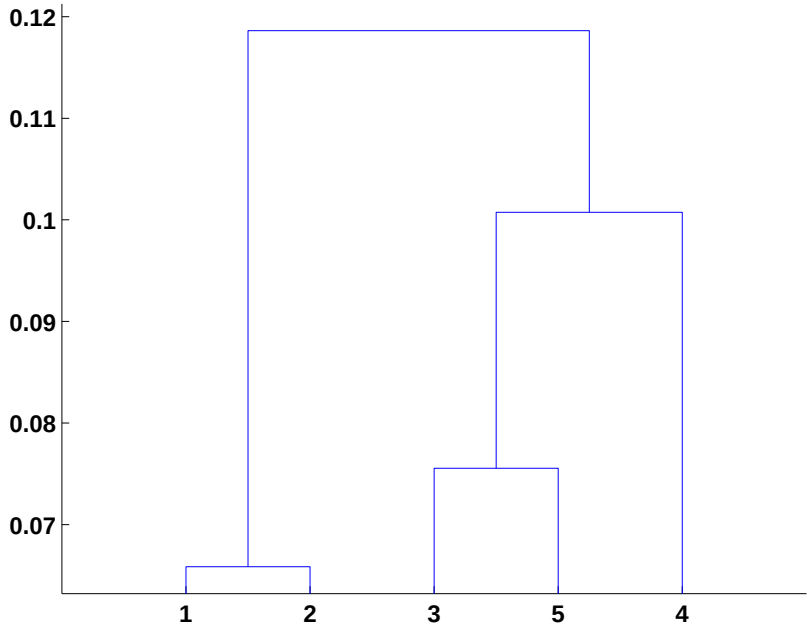


(b)

Figure 3.10. Hierarchical factor modeling results. (a) Factor loadings for *e-coli* data. (b) Inferred hierarchy for *e-coli* data.



(a)



(b)

Figure 3.11. Hierarchical factor modeling results. (a) Factor loadings for breast-cancer data. (b) Inferred hierarchy for breast-cancer data.

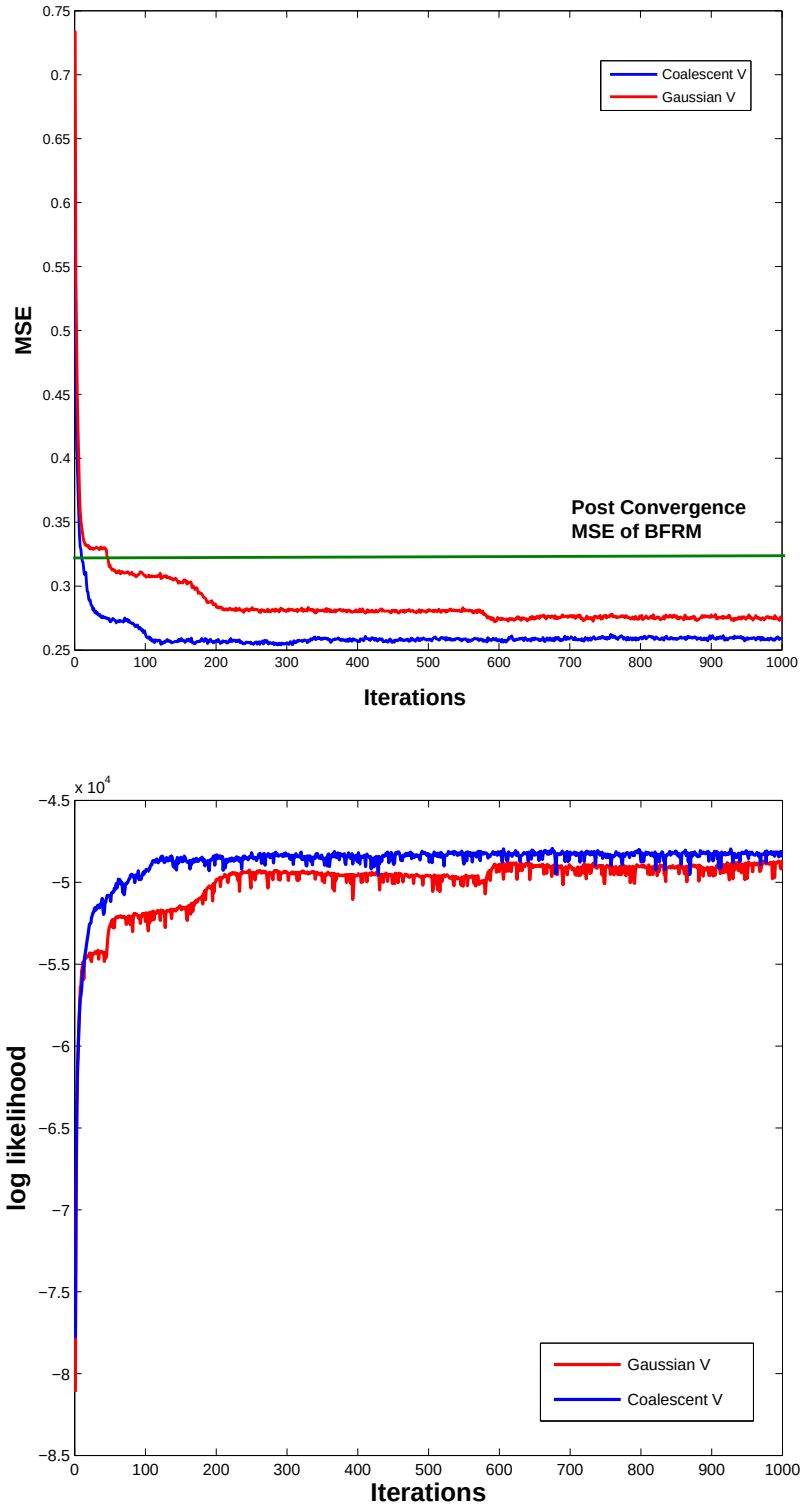


Figure 3.12. Convergence plots: (a) MSE on the breast-cancer data for BFRM (horizontal line), our model with Gaussian (top red curved line) and Coalescent (bottom blue curved line) priors. (b) Log-likelihoods for our model with Gaussian (bottom red curved line) and Coalescent (top blue curved line) priors.

3.5.3 Factor Regression

We report factor regression results for binary and real-valued responses and compare both variants of our model (Gaussian **V** and Coalescent **V**) against 3 different approaches: logistic regression, BFRM, and fitting a separate predictive model on the discovered factors (see Table 3.1 and Figure 3.12). The breast-cancer dataset had two binary response variables (phenotypes) associated with each sample. For this binary prediction task, we split the data into a training-set of 151 samples and test-set of 100 samples. This is essentially a transduction setting, as described in Section 3.2.4 and shown in Figure 3.2. For real-valued prediction task, we treated a 30x20 block of the data matrix as our held-out data and predicted it based on the rest of the entries in the matrix. This method of evaluation is akin to the task of image reconstruction (Verbeek et al., 2004). The results are averaged over 20 random initializations and the low error variances suggest that our method is fairly robust w.r.t. initializations.

3.6 Conclusions and Discussion

We have presented a fully nonparametric Bayesian approach to sparse factor regression, modeling the gene-factor relationship using a sparse variant of the IBP. However, the true power of nonparametric priors is evidenced by the ease of integration of task-specific models into the framework. Both gene selection and hierarchical factor modeling are straightforward extensions in our model that do not significantly complicate the inference procedure, but lead to improved model performance *and* more understandable outputs. We applied Kingman's coalescent as a hierarhical model on **V**, the matrix modulating the expression levels of genes in factors.

Table 3.1. Factor regression results

Model	**Binary (%error,std dev)**	**Real (MSE)**
LogReg	17.5 (1.6)	-
BFRM	19.8 (1.4)	0.48
Nor-V	15.8 (0.56)	0.45
Coal-V	14.6 (0.48)	0.43
PredModel	18.1 (2.1)	-

CHAPTER 4

NONPARAMETRIC BAYESIAN FACTOR ANALYSIS WITH MULTIPLE MODALITIES

In this chapter, we present a generalization of the nonparametric Bayesian latent factor model and show how we can extract latent factors shared between two or more modalities. In this chapter, we consider a special case of supervised dimensionality reduction for the multilabel prediction setting using Canonical Correlation Analysis (CCA) where the first modality is the features in the data and the second modality is the label matrix. However, our model is more general and can be applied for latent factor learning from multimodal data such as a collection of webpages that can be represented using different types of features such as the page-text, the anchor-text on hyperlinks pointed towards them, the images appearing in them, the social tags associated with them, and so on.

4.1 Introduction

Learning with examples having multiple labels is an important problem in machine learning and data mining. Such problems are encountered in a variety of application domains. For example, in text classification, a document (e.g., a newswire story) can be associated with multiple categories. Likewise, in bio-informatics, a gene or protein usually performs several functions. All these settings suggest a common underlying problem: predicting multivariate responses. When the responses come from a discrete set, the problem is termed as multilabel classification. The aforementioned setting is a special case of Multitask Learning (Caruana, 1997) when predicting each label is a task and all the tasks share a common source of input. An important characteristics of these problems is that the labels are not independent of each other but actually often have significant correlations

with each other. A naïve approach to learn in such settings is to train a separate classifier for each label. However, such an approach ignores the label correlations and leads to suboptimal performance (Ueda and Saito, 2003).

In this chapter, we show how Canonical Correlation Analysis (CCA) (Hotelling, 1936) can be used to exploit label relatedness, learning multiple prediction problems simultaneously. CCA is a useful technique for modeling dependencies between two (or more) sets of variables. One important application of CCA is in *supervised* dimensionality reduction, albeit in the more general setting where each example has several labels. In this setting, CCA on input-output pair $(\mathbf{X}, \mathbf{Y})$ can be used to project inputs $\mathbf{X}$ to a low-dimensional space directed by label information $\mathbf{Y}$. This makes CCA an ideal candidate for extracting useful predictive features from data in the context of multilabel prediction problems.

The classical CCA formulation, however, has certain inherent limitations. It is non-probabilistic, which means that it cannot deal with missing data, and precludes a Bayesian treatment, which can be important if the dataset size is small. An even more crucial issue is choosing the number of correlation components, which is traditionally dealt with by using cross-validation, or model-selection (Wang, 2007). Another issue is the potential sparsity (Sriperumbudur et al., 2009) of the underlying projections that is ignored by the standard CCA formulation.

Building upon the recently suggested probabilistic interpretation of CCA (Bach and Jordan, 2005), we propose a nonparametric, fully Bayesian framework that can deal with each of these issues. In particular, the proposed model can automatically select the number of correlation components, and effectively capture the sparsity underlying the projections. Our framework is based on the Indian Buffet Process (Ghahramani et al., 2007), a nonparametric Bayesian model to discover latent feature representation of a set of observations. In addition, our probabilistic model allows dealing with missing data and, in the supervised dimensionality reduction case, can incorporate *additional* unlabeled data one may have access to, making our CCA algorithm work in a semisupervised setting. Thus, apart from being a general, nonparametric, fully Bayesian solution to the CCA problem, our framework can be readily applied for learning useful predictive features from labeled (or *partially* labeled) data in the context of learning a set of related tasks.

This chapter is organized as follows. Section 4.2 introduces the CCA problem and

its recently proposed probabilistic interpretation. In Section 4.3, we describe our general framework for *infinite* CCA. Section 4.4 gives a concrete example of an application (multilabel learning) where the proposed approach can be applied. In particular, we describe a fully supervised setting (when the test data are not available at the time of training), and a semisupervised setting with partial labels (when we have access to test data at the time of training). We describe our experiments in Section 4.5, and discuss related work in Section 8.5, drawing connections of the proposed method with previously proposed ones for this problem.

4.2 Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) is a useful technique for modeling the relationships among a set of variables. CCA computes a low-dimensional *shared* embedding of a set of variables such that the correlation among the variables is maximized in the embedded space.

More formally, given a pair of variables $\mathbf{x} \in \mathbb{R}^{D_1}$ and $\mathbf{y} \in \mathbb{R}^{D_2}$, CCA seeks to find linear projections $\mathbf{u}_x$ and $\mathbf{u}_y$ such that the variables are maximally correlated in the projected space. The correlation coefficient between the two variables in the embedded space is given by

$$\rho = \frac{\mathbf{u}_x^T \mathbf{x}\mathbf{y}^T \mathbf{u}_y}{\sqrt{(\mathbf{u}_x^T \mathbf{x}\mathbf{x}^T \mathbf{u}_x)(\mathbf{u}_y^T \mathbf{y}\mathbf{y}^T \mathbf{u}_y)}}$$

Since the correlation is not affected by rescaling of the projections $\mathbf{u}_x$ and $\mathbf{u}_y$, CCA is posed as a constrained optimization problem.

$$\max_{\mathbf{u}_x, \mathbf{u}_y} \mathbf{u}_x^T \mathbf{x}\mathbf{y}^T \mathbf{u}_y, \textit{subject to} : \mathbf{u}_x^T \mathbf{x}\mathbf{x}^T \mathbf{u}_x = 1, \mathbf{u}_y^T \mathbf{y}\mathbf{y}^T \mathbf{u}_y = 1$$

It can be shown that the above formulation is equivalent to solving the following generalized eigen-value problem:

$$\begin{pmatrix} 0 & \mathbf{\Sigma_{xy}} \\ \mathbf{\Sigma_{yx}} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u_x} \\ \mathbf{u_y} \end{pmatrix} = \rho \begin{pmatrix} \mathbf{\Sigma_{xx}} & 0 \\ 0 & \mathbf{\Sigma_{yy}} \end{pmatrix} \begin{pmatrix} \mathbf{u_x} \\ \mathbf{u_y} \end{pmatrix}$$

where $\mathbf{\Sigma}$ denotes the covariance matrix of size $D \times D$ (where $D = D_1 + D_2$) obtained from the data samples $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$.

4.2.1 Probabilistic CCA

Bach and Jordan (Bach and Jordan, 2005) gave a probabilistic interpretation of CCA by posing it as a latent variable model. To see this, let $\mathbf{x}$ and $\mathbf{y}$ be two random vectors of size D_1 and D_2. Let us now consider the following latent variable model

$$\begin{aligned}
\mathbf{z} &\sim \mathcal{N}or(0, \boldsymbol{I}_K), \quad \min\{D_1, D_2\} \geq K \\
\mathbf{x} &\sim \mathcal{N}or(^{-}{}_x + \mathbf{W}_x\mathbf{z}, \boldsymbol{\Psi}_x), \quad \mathbf{W}_x \in \mathbb{R}^{D_1 \times K}, \boldsymbol{\Psi}_x \succeq 0 \\
\mathbf{y} &\sim \mathcal{N}or(^{-}{}_y + \mathbf{W}_y\mathbf{z}, \boldsymbol{\Psi}_y), \quad \mathbf{W}_y \in \mathbb{R}^{D_2 \times K}, \boldsymbol{\Psi}_y \succeq 0
\end{aligned}$$

Equivalently, we can also write the above as

$$[\mathbf{x}; \mathbf{y}] \sim \mathcal{N}or(\boldsymbol{\mu} + \mathbf{W}\mathbf{z}, \boldsymbol{\Psi})$$

where $\boldsymbol{\mu} = [\mu_x; \mu_y]$, $\mathbf{W} = [\mathbf{W}_x; \mathbf{W}_y]$, and $\boldsymbol{\Psi}$ is a block-diagonal matrix consisting of $\boldsymbol{\Psi}_x$ and $\boldsymbol{\Psi}_y$ on its diagonals. $[.;.]$ denotes row-wise concatenation. The latent variable $\mathbf{z}$ is shared between $\mathbf{x}$ and $\mathbf{y}$.

Bach and Jordan (Bach and Jordan, 2005) showed that, given the maximum likelihood solution for the model parameters, the expectations $\mathbb{E}(\mathbf{z}|\mathbf{x})$ and $\mathbb{E}(\mathbf{z}|\mathbf{y})$ of the latent variable $\mathbf{z}$ lie in the same subspace that classical CCA finds, thereby establishing the equivalence between the above probabilistic model and CCA.

The probabilistic interpretation opens doors to several extension of the basic setup proposed in (Bach and Jordan, 2005) which suggested a maximum likelihood approach for parameter estimation. However, it still assumes an *a priori* fixed number of canonical correlation components. In addition, another important issue is the sparsity of the underlying projection matrix, which is usually ignored.

4.3 The Infinite Canonical Correlation Analysis Model

Recall that the CCA problem can be defined as $[\mathbf{x}; \mathbf{y}] \sim \mathcal{N}or(\mathbf{W}\mathbf{z}, \boldsymbol{\Psi})$ (assuming centered data). A crucial issue in the CCA model is choosing the number of canonical correlation components, which is set to a fixed value in classical CCA (and even in the probabilistic extensions of CCA). In the Bayesian formulation of CCA, one can use the Automatic Relevance Determination (ARD) prior (Bishop, 1999) on the projection matrix $\mathbf{W}$ that gives a way to select this number. However, it would be more appropriate to have a principled way to automatically figure out this number based on the data.

We propose a nonparametric Bayesian model that selects the number of canonical correlation components automatically. More specifically, we use the Indian Buffet Process (Ghahramani et al., 2007) (Section 2.3) as a nonparametric prior on the projection matrix $\mathbf{W}$. The IBP prior allows $\mathbf{W}$ to have an unbounded number of columns which gives a way to automatically determine the dimensionality K of the latent space associated with $\mathbf{Z}$.

4.3.1 The Infinite CCA Model

In our proposed framework, the matrix $\mathbf{W}$ consisting of canonical correlation vectors is modeled using an IBP prior. However, since $\mathbf{W}$ can be real-valued and the IBP prior is defined only for binary matrices, we represent the $(D1 + D2) \times K$ matrix $\mathbf{W}$ as $(\mathbf{B} \odot \mathbf{V})$, where $\mathbf{B} = [\mathbf{B}_x; \mathbf{B}_y]$ is a $(D_1 + D_2) \times K$ binary matrix, $\mathbf{V} = [\mathbf{V}_x; \mathbf{V}_y]$ is a $(D_1 + D_2) \times K$ real-valued matrix, and $\odot$ denotes their element-wise (Hadamard) product. We place an IBP prior on $\mathbf{B}$ that automatically determines K, and a Gaussian prior on $\mathbf{V}$. Note that $\mathbf{B}$ and $\mathbf{V}$ have the same number of columns. Under this model, two random vectors $\mathbf{x}$ and $\mathbf{y}$ can be modeled as $\mathbf{x} = (\mathbf{B}_x \odot \mathbf{V}_x)\mathbf{z} + \mathbf{E}_x$ and $\mathbf{y} = (\mathbf{B}_y \odot \mathbf{V}_y)\mathbf{z} + \mathbf{E}_y$. Here, $\mathbf{z}$ is shared between $\mathbf{x}$ and $\mathbf{y}$, and $\mathbf{E}_x$ and $\mathbf{E}_y$ are observation-specific noise.

In the full model, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ is a $D_1 \times N$ matrix consisting of N samples of D_1 dimensions each, and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$ is another matrix consisting of N samples of D_2 dimensions each. Here is the generative story for our basic model (see Figure 4.1):

$$
\begin{aligned}
\mathbf{B} &\sim \mathcal{IBP}(\alpha) \\
\mathbf{V} &\sim \mathcal{N}or(0, \sigma_v^2 \mathbf{I}), \quad \sigma_v \sim IG(a, b) \\
\mathbf{Z} &\sim \mathcal{N}or(0, I) \\
[\mathbf{X}; \mathbf{Y}] &\sim \mathcal{N}or(\mathbf{B} \odot \mathbf{V})\mathbf{Z}, \mathbf{\Psi}),
\end{aligned}
$$

where $\mathbf{\Psi}$ is a block-diagonal matrix of size $D \times D$ where $D = (D_1 + D_2)$, with $\mathbf{\Psi}_x$ and $\mathbf{\Psi}_y$ on its diagonal. Both $\mathbf{\Psi}_x$ and $\mathbf{\Psi}_y$ have an inverse-Wishart prior on them.

Since our model is probabilistic, it can also deal with the problem when $\mathbf{X}$ or $\mathbf{Y}$ have missing entries. This is particularly important in the case of supervised dimensionality reduction (i.e., $\mathbf{X}$ consisting of inputs and $\mathbf{Y}$ associated responses) when the labels for some of the inputs are unknown, making it a model for *semisupervised* dimensionality reduction with partially labeled data. In addition, placing the IBP prior on the projection matrix $\mathbf{W}$

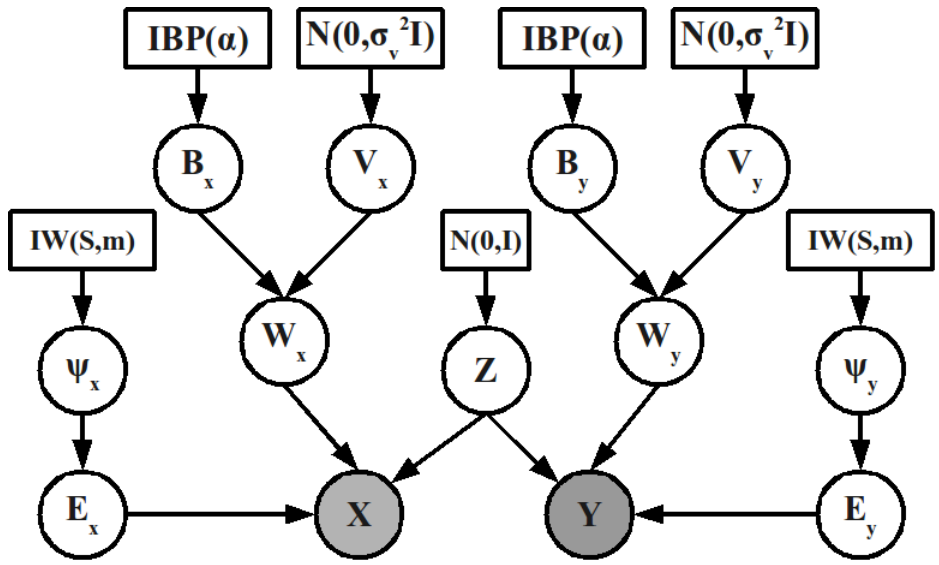


Figure 4.1. The graphical model depicts the fully supervised case when all variables X and Y are observed. The semisupervised case can have X and/or Y consisting of missing values as well. The graphical model structure remains the same

(via the binary matrix **B**) also helps in capturing the sparsity in **W** (see Results section for evidence).

4.3.2 Inference

We take a fully Bayesian approach by treating everything at latent variables and computing the posterior distributions over them. We use Gibbs sampling with a few Metropolis-Hastings steps to do inference in this model.

In what follows, **D** denotes the data $[\mathbf{X}; \mathbf{Y}]$, $\mathbf{B} = [\mathbf{B}_x; \mathbf{B}_y]$, and $\mathbf{V} = [\mathbf{V}_x; \mathbf{V}_y]$

4.3.3 Sampling B

Sampling the binary IBP matrix **B** consists of sampling existing dishes, proposing new dishes and accepting or rejecting them based on the acceptance ratio in the associated M-H step. For sampling existing dishes, an entry in $\mathbf{B}$ is set as 1 according to $p(B_{ik} = 1|\mathbf{D}, B_{-ik}, \mathbf{V}, \mathbf{Z}, \mathbf{\Psi}) \propto \frac{m_{-i,k}}{D} p(\mathbf{D}|\mathbf{B}, \mathbf{V}, \mathbf{F}, \mathbf{\Psi})$ whereas it is set as 0 according to $p(B_{ik} = 0|\mathbf{D}, B_{-ik}, \mathbf{V}, \mathbf{Z}, \mathbf{\Psi}) \propto \frac{D-m_{-i,k}}{D} p(\mathbf{D}|\mathbf{B}, \mathbf{V}, \mathbf{Z}, \mathbf{\Psi})$. $m_{-i,k} = \sum_{j \neq i} B_{jk}$ is how many other customers chose dish k.

For sampling new dishes, we use an M-H step where we simultaneously propose $\mathbf{J} = (K^{new}, V^{new}, Z^{new})$ where $K^{new} \sim Poisson(\alpha/D)$. We accept the proposal with an

acceptance probability given by $a = \min\{1, \frac{p(rest|\mathbf{J}^*)}{p(rest|\mathbf{J})}\}$. Here, $p(rest|\eta)$ is the probability of the data given parameters η. We propose V^{new} from its prior (Gaussian) but, for faster mixing, we propose Z^{new} from its posterior.

4.3.4 Sampling V

We sample the real-valued matrix **V** from its posterior, which is a normal distribution with covariance $\Sigma_{i,k} = (\sum_{n=1}^{N} \frac{Z_{k,n}^2}{\Psi_i} + \frac{1}{\sigma_v^2})^{-1}$ and mean $\mu_{i,k} = \Sigma_{i,k}(\sum_{n=1}^{N} A_{k,n} D_{i,k}^*)\Psi_i^{-1}$. We define $D_{i,k}^* = D_{i,n} - \sum_{l=1,l\neq k}^{K}(B_{i,l}V_{i,l})Z_{l,n}$. The hyperparameter σ_v on **V** has an inverse-gamma prior and the posterior also has the same form. Note that the number of columns in **V** is the same as the number of columns in the IBP matrix **B**.

4.3.5 Sampling Z

We sample for **Z** from its posterior, which is a normal distribution with mean $\bar{} = \mathbf{W^T}(\mathbf{WW^T}+\mathbf{\Psi})^{-1}\mathbf{D}$ and covariance $\mathbf{\Sigma} = \mathbf{I} - \mathbf{W^T}(\mathbf{WW^T}+\mathbf{\Psi})^{-1}\mathbf{W}$, where $\mathbf{W} = \mathbf{B} \odot \mathbf{V}$.

Note that, in our sampling scheme, we considered the matrices $\mathbf{B}_x$ and $\mathbf{B}_y$ as simply parts of the big IBP matrix **B**, and sampled them together using a single IBP draw. However, one could also sample them separately as two separate IBP matrices for $\mathbf{B}_x$ and $\mathbf{B}_y$. This would require different IBP draws for sampling $\mathbf{B}_x$ and $\mathbf{B}_y$ with some modification of the existing Gibbs sampler. Different IBP draws could result in a different number of nonzero columns in $\mathbf{B}_x$ and $\mathbf{B}_y$. To deal with this issue, one could sample $\mathbf{B}_x$ (say having K_x nonzero columns) and $\mathbf{B}_y$ (say having K_y nonzero columns) first, introduce extra dummy columns ($|K_x - K_y|$ in number) in the matrix having a smaller number of nonzero columns, and then set all such columns to zero. The effective K for each iteration of the Gibbs sampler would be $\max\{K_x, K_y\}$. A similar scheme could also be followed for the corresponding real-valued matrices $\mathbf{V}_x$ and $\mathbf{V}_y$, sampling them in conjunction with $\mathbf{B}_x$ and $\mathbf{B}_y$, respectively.

4.4 Multitask Learning Using Infinite CCA

Having set up the framework for infinite CCA, we now describe its applicability for the problem of Multitask Learning. In particular, we consider the setting when each example is associated with multiple labels. Here, predicting each individual label becomes a task to be learned. Although one can individually learn a separate model for each task, doing this would ignore the label correlations. This makes borrowing the information across

tasks crucial, making it imperative to share the statistical strength across all the task. With this motivation, we apply our infinite CCA model to capture the label correlations and to learn better predictive features from the data by projecting it to a subspace directed by label information. It has been empirically and theoretically (Yu et al., 2006) shown that incorporating label information in dimensionality reduction indeed leads to better projections if the final goal is prediction.

More concretely, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ be an $D \times N$ matrix of predictor variables, and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$ be an $M \times N$ matrix of the responses variables (i.e., the labels) with each $\mathbf{y}_i$ being an $M \times 1$ vector of responses for input $\mathbf{x}_i$. The labels can take real (for regression) or categorical (for classification) values. The infinite CCA model is applied on the pair $\mathbf{X}$ and $\mathbf{Y}$, which is akin to doing supervised dimensionality reduction for the inputs $\mathbf{X}$. Note that the generalized eigenvalue problem posed in such a supervised setting of CCA consists of cross-covariance matrix Σ_{XY} and label covariance matrix Σ_{YY}. Therefore, the projection takes into account both the input-output correlations and the label correlations. Such a subspace therefore is expected to consist of much better predictive features than one obtained by a naïve feature extraction approach such as simple PCA that completely ignores the label information, or approaches like Linear Discriminant Analysis (LDA) that do take into account label information but ignore label correlations.

Multitask learning using the infinite CCA model can be done in two settings: supervised and semisupervised, depending on whether or not the inputs of test data are involved in learning the shared subspace $\mathbf{Z}$.

4.4.1 Fully Supervised Setting

In the supervised setting, CCA is done on labeled data $(\mathbf{X}, \mathbf{Y})$ to give a single shared subspace $\mathbf{Z} \in \mathbb{R}^{K \times N}$ that is good across all tasks. A model is then learned in the $\mathbf{Z}$ subspace to learn M task parameters $\{\theta_m\} \in \mathbb{R}^{K \times 1}$ where $m \in \{1, \ldots, M\}$. Each of the parameters θ_m is then used to predict the labels for the test data of task m. However, since the test data are still D dimensional, we need to either separately project it down onto the K dimensional subspace and do predictions in this subspace, or "inflate" each task parameter back to D dimensions by applying the projection matrix $\mathbf{W}_x$ and do predictions in the original D dimensional space. The first option requires using the fact that $P(\mathbf{Z}|\mathbf{X}_{te}) \propto P(\mathbf{X}_{te}|\mathbf{Z})P(\mathbf{Z})$, which is a Gaussian $\mathcal{N}or(\mu_{Z|X}, \Sigma_{Z|X})$ with $\mu_{Z|X} = (\mathbf{W}_x^T \mathbf{\Psi}_x \mathbf{W}_x + \mathbf{I})^{-1} \mathbf{W}_x^T \mathbf{X}_{te}$ and $\Sigma_{Z|X} =$

$(\mathbf{W}_x^T \mathbf{\Psi}_x \mathbf{W}_x + \mathbf{I})^{-1}$. With the second option, we can inflate each learned task parameter back to D dimensions by applying the projection matrix $\mathbf{W}_x$. We choose the second option for the experiments. We call this fully supervised setting as model-1.

4.4.2 A Semisupervised Setting

In the semisupervised setting, we combine training data and test data (with unknown labels) as $\mathbf{X} = [\mathbf{X}_{tr}, \mathbf{X}_{te}]$ and $\mathbf{Y} = [\mathbf{Y}_{tr}, \mathbf{Y}_{te}]$ where the labels $\mathbf{Y}_{te}$ are unknown. The infinite CCA model is then applied on the pair $(\mathbf{X}, \mathbf{Y})$ and the parts of $\mathbf{Y}$ consisting of $\mathbf{Y}_{te}$ are treated as latent variables to be imputed. With this model, we get the embeddings also for the test data and thus training and testing both take place in the K dimensional subspace, unlike model-1 in which training is done in K dimensional subspace and predictions are made in the original D dimensional subspace. We call this semisupervised setting as model-2.

4.5 Experiments

Here, we report our experimental results on several synthetic and real-world datasets. We first show our results with the infinite CCA as a stand-alone algorithm for CCA by using it on a synthetic dataset, demonstrating its effectiveness in capturing the canonical correlations. We then also report our experiments on applying the infinite CCA model to the problem of Multitask Learning on two real-world datasets.

4.5.1 Infinite CCA Results on Synthetic Data

In the first experiment, we demonstrate the effectiveness of our proposed infinite CCA model in discovering the correct number of canonical correlation components, and in capturing the sparsity pattern underlying the projection matrix. For this, we generated two datasets of dimensions 25 and 10, respectively, with each having 100 samples. For this synthetic dataset, we knew the ground truth (i.e., the number of components, and the underlying sparsity of projection matrix). In particular, the dataset had 4 correlation components with a 63% sparsity in the true projection matrix. We then ran both the classical CCA and the infinite CCA algorithm on this dataset. Looking at *all* the correlations discovered by classical CCA, we found that it discovered 8 components having significant correlations, whereas our model correctly discovered exactly 4 components in the first place (we extract the MAP samples for $\mathbf{W}$ and $\mathbf{Z}$ output by our Gibbs sampler). Thus, on this

small dataset, standard CCA indeed seems to be finding spurious correlations, indicating a case of overfitting (the overfitting problem of classical CCA was also observed in (Klami and Kaski, 2007) when comparing Bayesian versus classical CCA). Furthermore, as expected, the projection matrix inferred by the classical CCA had no exact zero entries and even after thresholding significantly small absolute values to zero, the uncovered sparsity was only about 25%. On the other hand, the projection matrix inferred by the infinite CCA model had 57% exact zero entries and 62% zero entries after thresholding very small values, thereby demonstrating its effectiveness in also capturing the sparsity patterns.

4.5.2 Infinite CCA Applied to Multilabel Prediction

In the second experiment, we use the infinite CCA model to learn a set of related task in the context of multilabel prediction. For our experiments, we use two real-world multilabel datasets (Yeast and Scene) from the UCI repository. The Yeast dataset consists of 1500 training and 917 test examples, each having 103 features. The number of labels (or tasks) per example is 14. The Scene dataset consists of 1211 training and 1196 test examples, each having 294 features. The number of labels per example for this dataset is 6. We compare the following models for our experiments.

- Full: Train separate classifiers (SVM) on the full feature set for each task.
- PCA: Apply PCA on training and test data and then train separate classifiers for each task in the low-dimensional subspace. This baseline ignores the label information while learning the low-dimensional subspace.
- CCA: Apply classical CCA on training data to extract the shared subspace, learn separate model (i.e., task parameters) for each task in this subspace, project the task parameters back to the original D dimensional feature space by applying the projection $\mathbf{W}_x$, and do predictions on the test data in this feature pace.
- Model-1: Use our supervised infinite CCA model to learn the shared subspace using only the training data (see Section 4.4.1).
- Model-2: Use our semisupervised infinite CCA model to *simultaneously* learn the shared subspace for both training and test data (see Section 4.4.2).

The performance metrics used are overall accuracy, F1-Macro, F1-Micro, and AUC (Area Under ROC Curve). For PCA and CCA, we chose K that gives the best performance, whereas this parameter was learned automatically for both of our proposed models. The results are shown in Table-4.1. As we can see, both the proposed models do better than the other baselines. Of the two proposed model, we see that model-2 does better in most cases, suggesting that it is useful to incorporate the test data while learning the projections. This is possible in our probabilistic model since we could treat the unknown $\mathbf{Y}$s of the test data as latent variables to be imputed while doing the Gibbs sampling.

We note here that our results are with cases where we only had access to a small number of related task (Yeast has 14, Scene has 6). We expect the performance improvements to be even more significant when the number of (related) tasks is high.

4.6 Related Work

A number of approaches have been proposed in the recent past for the problem of supervised dimensionality reduction of *multilabel* data. The few approaches that exist include Partial Least Squares (Arenas-García et al., 2006), multilabel informed latent semantic indexing (Yu et al., 2005), and multilabel dimensionality reduction using dependence maximization (MDDM) (Zhou, 2008). None of these, however, deal with the case when the data are only partially labeled. Somewhat similar in spirit to our approach is the work on supervised probabilistic PCA (Yu et al., 2006) that extends probabilistic PCA to the setting when we also have access to labels. However, it assumes a fixed number of components and does not take into account sparsity of the projections.

The CCA-based approach to supervised dimensionality reduction is more closely related to the notion of dimension reduction for regression (DRR), which is formally defined as finding a low-dimensional representation $\mathbf{z} \in \mathbb{R}^K$ of inputs $\mathbf{x} \in \mathbb{R}^D$ ($K \ll D$) for predicting multivariate outputs $\mathbf{y} \in \mathbb{R}^M$. An important notion in DRR is that of sufficient dimensionality reduction (SDR) (Fukumizu et al., 2004, Globerson and Tishby, 2003), which states that given $\mathbf{z}$, $\mathbf{x}$ and $\mathbf{y}$ are conditionally independent, i.e., $\mathbf{x} \perp\!\!\!\perp \mathbf{y}|\mathbf{z}$. As we can see in the graphical model shown in Figure 4.1, the probabilistic interpretation of CCA yields the same condition with $\mathbf{X}$ and $\mathbf{Y}$ being conditionally independent given $\mathbf{Z}$.

Among the DRR-based approaches to dimensionality reduction for real-valued multi-

Table 4.1. Results on the multilabel classification task. Bold face indicates the best performance. Model-1 and Model-2 scores are averaged over 10 runs with different initializations.

Model	**Yeast**				**Scene**			
	Acc	F1-macro	F1-micro	AUC	Acc	F1-macro	F1-micro	AUC
Full	0.5583	0.3132	0.3929	0.5054	0.7565	0.3445	0.3527	0.6339
PCA	0.5612	0.3144	0.4648	0.5026	0.7233	0.2857	0.2734	0.6103
CCA	0.5441	0.2888	0.3923	0.5135	0.7496	0.3342	0.3406	0.6346
Model-1	0.5842	0.3327	0.4402	0.5232	0.7533	0.3630	0.3732	0.6517
Model-2	**0.6156**	**0.3463**	**0.4954**	**0.5386**	**0.7664**	**0.3742**	**0.3825**	**0.6686**

label data, Covariance Operator Inverse Regression (COIR) exploits the covariance structures of both the inputs and outputs (Kim and Pavlovic, 2009). Please see (Kim and Pavlovic, 2009) for more details on the connection between COIR and CCA. Besides the DRR-based approaches, the problem of extracting useful features from data, particularly with the goal of making predictions, has also been considered in other settings. The information bottleneck (IB) method (Tishby, Pereira, and Bialek, Tishby et al.) is one such example. Given input-output pairs $(\mathbf{X}, \mathbf{Y})$, the information bottleneck method aims to obtain a compressed representation $\mathbf{T}$ of $\mathbf{X}$ that can account for $\mathbf{Y}$. IB achieves this using a single tradeoff parameter to represent the tradeoff between the *complexity* of the representation of $\mathbf{X}$, measured by $I(\mathbf{X}; \mathbf{T})$, and the *accuracy* of this representation, measured by $I(\mathbf{T}; \mathbf{Y})$, where $I(.;.)$ denotes the mutual information between two variables. In another recent work (Ji and Ye, 2009), a joint learning framework is proposed, which performs dimensionality reduction and multilabel classification simultaneously.

In the context of CCA as a stand-alone problem, sparsity is another important issue. In particular, sparsity improves model interpretation and has been gaining lots of attention recently. Existing works on sparsity in CCA include the double barrelled lasso, which is based on a convex least squares approach (Shawe-Taylor, 2008), and CCA as a sparse solution to the generalized eigenvalue problem (Sriperumbudur et al., 2009), which is based on constraining the cardinality of the solution to the generalized eigenvalue problem to obtain a sparse solution. Another recent solution is based on a direct greedy approach, which bounds the correlation at each stage (Wiesel et al., 2008).

The probabilistic approaches to CCA include the works of (Klami and Kaski, 2007) and (Archambeau and Bach, 2008), both of which use an automatic relevance determination

(ARD) prior (Bishop, 1999) to determine the number of relevant components, which is a rather ad-hoc way of doing this. In contrast, a nonparametric Bayesian alternative proposed here is a more principled method to determine the number of components.

We note that the sparse Factor Analysis model proposed in (Rai and Daumé III, 2008) actually falls out as a special case of our proposed infinite CCA model if one of the datasets ($\mathbf{X}$ or $\mathbf{Y}$) is absent and the noise covariance matrix $\mathbf{\Psi}$ is diagonal. Besides, the sparse Factor Analysis model is limited to Factor Analysis whereas the proposed model can be seen as an infinite generalization of both an unsupervised problem (sparse CCA), and (semi)supervised problem (dimensionality reduction using CCA with full or partial label information), with the latter being especially relevant for Multitask Learning in the presence of multiple labels.

Finally, Multitask Learning has been tackled using a variety of different approaches, primarily depending on what notion of task relatedness is assumed. Some of the examples include tasks generated from an IID space (Baxter, 2000), and learning multiple tasks using a hierarchical prior over the task space (Daumé III, 2009, Xue et al., 2007b), among others. In this work, we consider multilabel prediction in particular, based on the premise that a set of such related tasks share an underlying low-dimensional feature space (Ji et al., 2008) that captures the task relatedness.

4.7 Conclusion

We have presented a nonparametric Bayesian model for the Canonical Correlation Analysis problem to discover the dependencies between a set of variables. In particular, our model does not assume a fixed number of correlation components and this number is determined automatically based only on the data. In addition, our model enjoys sparsity, making the model more interpretable. The probabilistic nature of our model also allows dealing with missing data. Finally, we also demonstrate the model's applicability to the problem of multilabel learning where our model, directed by label information, can be used to automatically extract useful predictive features from the data.

CHAPTER 5

MULTITASK LEARNING USING NONPARAMETRIC BAYESIAN PREDICTOR SUBSPACES

In this chapter, we present a nonparametric Bayesian model for the problem of Multitask Learning. Our model is based on the assumption that the task parameters (e.g., the weight vectors of regression or classification tasks) of the multiple tasks live on a low-dimensional linear subspace. This model will form the building block of a more general model for Multitask Learning that will be presented in the next chapter.

5.1 Introduction

Many learning settings consist of multiple prediction problems that are related with each other in some way. A common instance is multivariate regression or multilabel classification where each example is associated with several response variables (real-valued for regression, and discrete-valued for classification). For example, given a document, one may be interested in predicting its topic category as well as its author. Clearly, such tasks are expected to be related. A simple way to learn such multiple prediction problems would be to simply treat them as separate problems and learn separate models for each of them, essentially ignoring any correlation that might exist among them. Such an approach, however, fails to exploit any correlations there may be among these tasks, and it is desirable to share information across tasks if they are related.

Motivated by this idea, a number of techniques have been proposed to exploit task relatedness in order to better learn a set of related tasks, rather than learning them individually. This is commonly known as Multitask Learning (Caruana, 1997), "learning to learn" (Heskes, 2000), inductive bias (Baxter, 2000), or predicting multivariate responses

(Breiman and Friedman, 1997), where multiple tasks are pooled together with the goal of improving the generalization performance of all the tasks. The idea is to use some aspect that can be shared across all the tasks in order to share their individual statistical strengths, compensating for the paucity of labeled examples.

In this chapter, we consider one such aspect, namely a *shared* predictor subspace. The assumption here is that all the task parameters share an underlying *basis space*, which accounts for the task relatedness. Each individual task can then be represented as a linear combination of the set of basis tasks. Our predictor subspace model is similar in spirit to (Zhang et al., 2006, 2008). In this work, we propose a nonparametric, fully Bayesian framework that can learn this subspace without making any parametric assumptions (e.g., the framework does not assume the intrinsic dimensionality of the subspace to be known *a priori*). We present two models to learn such a subspace, with a special emphasis on cases when the number of tasks and/or the number of examples per task is small. In this chapter, we concentrate on Bayesian linear regression (for regression) and Bayesian logistic regression (for classification). The framework, however, is general enough and can accommodate a variety of different probabilistic discriminative models. In addition, being a hierarchical Bayesian model, the model can easily be extended to a *mixture* of subspaces setting (described in the next chapter) which allows the task parameters to share a *nonlinear* manifold.

In Section 5.2, we describe the problem setup and our basic framework to model task relatedness. Section 5.3 describes both our models. Section 5.4 talks about inference in our model, Section 8.4 reports experimental results, and Section 8.5 discusses related work. We finally discuss the mixture extension of our work and conclude with Section 5.9.

5.2 Latent Subspace Model for Task Parameters

To model task relatedness, we assume that the tasks have an underlying basis space and each actual task is a linear combination of the basis vectors (which act as "source" tasks). More specifically, suppose we have M tasks (regression/classification) represented by task parameters $\theta_1, \ldots, \theta_M$ where $\theta_m \in \mathbb{R}^D$ is the task parameter for the m-th task. We assume the following generative model for each task parameter:

$$\theta_m = \mathbf{Z}\mathbf{A}_m + \epsilon_m$$

Here, $\mathbf{Z} \in \mathbb{R}^{D \times K}$ is a matrix in which each column is a D dimensional basis vector, $\mathbf{A}_m \in \mathbb{R}^{K \times 1}$ is the the set of coefficients for the mth task parameter, and ϵ_m is task-specific noise. The matrix $\mathbf{Z}$ under this model defines the latent space underlying the set of predictors, and is shared across all tasks, justifying the task relatedness. The same generative model, with all task parameters grouped together in a matrix $\Theta = [\theta_1 \ldots \theta_M] \in \mathbb{R}^{D \times M}$, can be written in a matrix form as $\Theta = \mathbf{Z}\mathbf{A}_\theta + \mathbf{E}$, where $\mathbf{A}_\theta = [\mathbf{A}_1 \ldots \mathbf{A}_M]$.

Together, the matrix $\mathbf{Z}$ of basis tasks, and the coefficients $[\mathbf{A}_1 \ldots \mathbf{A}_M]$ give the task parameters a parsimonious representation where each $D \times 1$ task parameter vector is represented by a vector of size $K \times 1$, with $K \ll D$. Finally, each row e_m of the $D \times M$ matrix $\mathbf{E}$ explains the task-specific idiosyncrasies and is assumed to be drawn from a multivariate Gaussian with a diagonal covariance matrix $\Psi = diag(\psi_{11}, \ldots, \psi_{DD})$.

At first blush, such a setup may seem like Factor Analysis (Bartholomew and Knott, 1999, Rai and Daumé III, 2008). However, unlike Factor Analysis, e.g., $\mathbf{X} = \mathbf{Z}\mathbf{A} + \mathbf{E}$ type of set-up where the data $\mathbf{X}$ is observed, in this case, the matrix Θ of task parameters is *not* observed. So the "data" Θ itself is a latent variable in this model (others being $\mathbf{Z}, \mathbf{A}, \mathbf{E}$, and the associated hyperparameters). The goal is to learn Θ along with all the other latent variables, harnessing the data available from all the tasks. Also note that it is a *supervised* setting unlike standard Factor Analysis.

A crucial issue in the model is determining the intrinsic dimensionality and sparsity of the underlying predictor subspace defined by $\mathbf{Z}$. We propose a nonparametric Bayesian model based on the recently proposed Indian Buffet Process (Section 2.3) (Ghahramani et al., 2007) to deal with both these issues. The dimensionality K of the latent space and the degree of sparsity of the basis space defined by $\mathbf{Z}$ is automatically determined by the IBP prior. Note that the sparsity of $\mathbf{Z}$ is akin to imposing an ℓ_1-type regularization on $\mathbf{Z}$ as in the Lasso framework, or assuming a Laplace prior on the columns of $\mathbf{Z}$: $Z_k \sim \prod_{d=1}^{D} \textsc{Laplace}(0, \eta)$.

5.3 Infinite Latent Subspace Models for Multitask Learning

Our goal is to simultaneously learn several prediction tasks. In the rest of the exposition and our experiments, we consider the special case of multilabel prediction where each

input $\mathbf{x}$ is associated with multiple labels. Therefore, predicting each label is a task. Our framework is, however, more general and can also be applied for cases where each prediction problem has its own source of input.

In the multilabel setting, learning the prediction task for the mth label amounts to learning the task parameter θ_m. Formally, given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1^m), \ldots, (\mathbf{x}_N, y_N^m)\}$ for task m where $\mathbf{x}_i \in \mathbb{R}^D$ and y_i^m is a real (for regression) or a binary valued (for classification) response, a learning task parameterized by θ_m, can be defined as:

$$\text{Regression: } y_i^m \sim \mathcal{N}or(\theta_m^T \mathbf{x}_i, \rho^2)$$
$$\text{Classification: } y_i^m \sim \mathcal{B}in(1/(1 + e^{-\theta_m^T \mathbf{x}_i}))$$

To follow a more compact notation, we shall denote the inputs $[\mathbf{x}, \ldots, \mathbf{x}_N]$ by an $N \times D$ matrix $\mathbf{X}$, the responses for all the M tasks by an $N \times M$ matrix $\mathbf{Y}$, and the M task parameters as a $D \times M$ matrix $\Theta = [\theta_1 \ldots \theta_M] \in \mathbb{R}^{D \times M}$. With this notation, we can define the prediction setting as a probabilistic model $\mathbf{Y}|\Theta, \mathbf{X} \sim \mathcal{N}or(\mathbf{Y}|\mathbf{X}^T\Theta, \rho^2 I)$ for regression (Bayesian linear regression), and $\mathbf{Y}|\Theta, \mathbf{X} \sim \mathcal{B}in(1/(1+e^{-\mathbf{X}^T\Theta}))$ for classification (Bayesian logistic regression).

Recall our original setup $\Theta = \mathbf{Z}\mathbf{A}_\theta + \mathbf{E}$. We wish to model the matrix $\mathbf{Z}$ using the Indian Buffet Process (IBP), thereby automatically choosing the intrinsic dimensionality of the task basis space defined by $\mathbf{Z}$. However, since IBP defines a distribution over binary matrices and $\mathbf{Z}$ needs to be a real-valued matrix, we model $\mathbf{Z}$ as $\mathbf{B} \odot \mathbf{V}$, the element-wise product of a binary matrix $\mathbf{B}$ and a real-valued matrix $\mathbf{V}$, both of size $D \times K$. We place an IBP prior over the binary matrix $\mathbf{B}$ and a Gaussian prior over the real-valued matrix $\mathbf{V}$. Our complete hierarchical model is the following (the corresponding graphical model shown in Figure 5.1: Top; error term not shown for the sake of brevity):

$$\begin{aligned}
\mathbf{Y} &\sim \mathcal{N}or(\mathbf{X}^T\Theta, \rho^2 I)(regression) \\
\mathbf{Y} &\sim \mathcal{B}in(1/(1 + e^{-\mathbf{X}^T\Theta})(classification) \\
\Theta &= (\mathbf{B} \odot \mathbf{V})\mathbf{A}_\theta + \mathbf{E} \\
\mathbf{B} &\sim \mathcal{IBP}(\alpha) \\
\mathbf{V} &\sim \mathcal{N}or(0, \sigma_v^2 \mathbf{I}), \quad \sigma_v \sim IG(a, b) \\
\mathbf{A}_\theta &\sim \mathcal{N}or(0, \sigma_\theta^2 \mathbf{I}), \quad \sigma_\theta \sim IG(c, d)
\end{aligned}$$

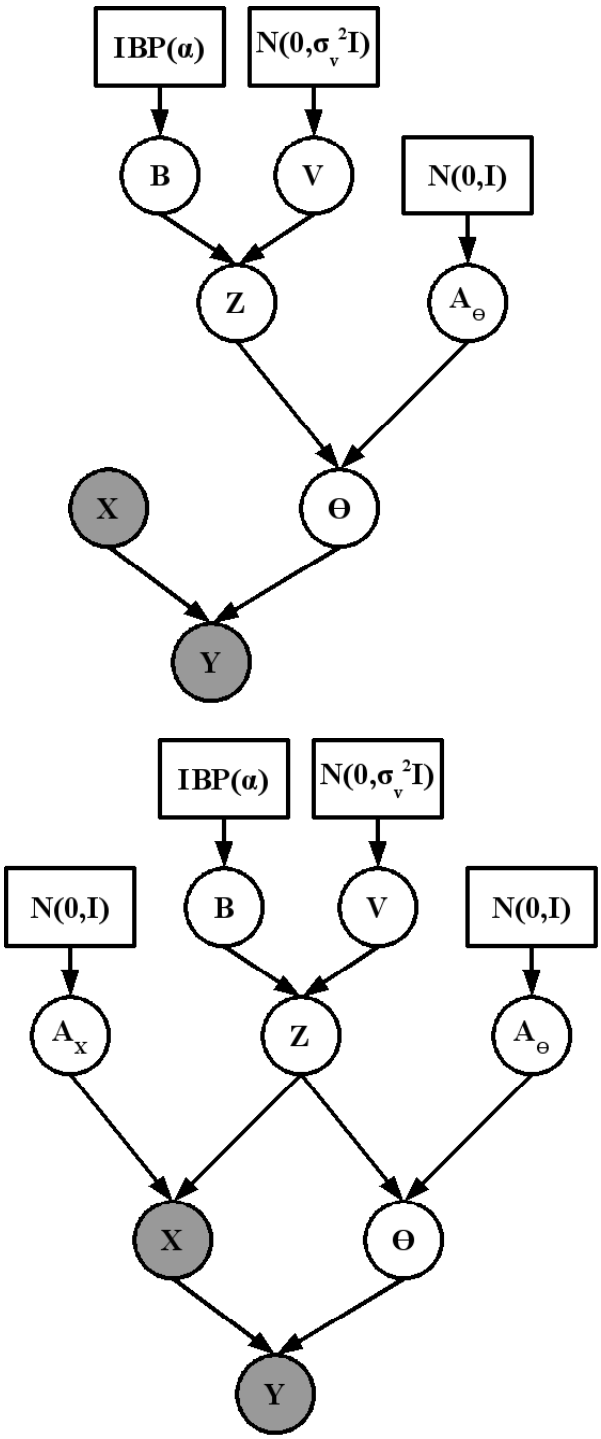


Figure 5.1. Predictor subspace model. Top: our basic model. Bottom: the augmented model using both task parameters *and* input data. **X** in the augmented model can additionally also consist of unlabeled data. Noise hyperparameters not shown for the sake of brevity. In both the models, the shaded nodes are observed, and the remaining ones (including the matrix Θ consisting of task parameters, and the noise hyperparameters) are latent variables to be learned.

$$\mathbf{E} \sim \mathcal{N}or(0, \Psi), \quad \Psi_D \sim IG(e, f)$$

Here Θ, which is itself a latent variable, acts as the "data" in the model and depends on other latent variables in the model, and the data from actual tasks ($\mathbf{B}$,$\mathbf{V}$,$\mathbf{A}_\theta$,$\mathbf{E}$,$\mathbf{X}$,$\mathbf{Y}$). Our proposed model learns Θ (along with learning the latent subspace underlying Θ) by sharing information across all the tasks.

5.3.1 An Augmented Model for Learning Task Basis

Learning the task subspace $\mathbf{Z}$ ($= \mathbf{B} \odot \mathbf{V}$) reliably would require a reasonable amount of data. In the basic model, the only available "data" for learning $\mathbf{Z}$ is Θ (which, under our probabilistic model, is actually itself a latent variable to be learned). Given related but only a small number of tasks M, the $D \times M$ matrix Θ may not be enough to reliably learn the basis $\mathbf{Z}$. This motivates our second model that allows also using the inputs $\mathbf{X}$ from each task to improve the learning of $\mathbf{Z}$. Under this model (shown in Figure 5.1: Bottom), it is assumed that the task parameters Θ and the inputs $\mathbf{X}$ both share the same basis space $\mathbf{Z}$, with different mixing matrices $\mathbf{A}_\theta$ and $\mathbf{A}_\mathbf{x}$, respectively. This model can be thought of as simultaneously discovering both the task parameter basis, as well as the latent features underlying the data $\mathbf{X}$. Furthermore, under this model, the data matrix $\mathbf{X}$ need not only consist of examples for which labels are known. So, the matrix $\mathbf{X}$ shown in Figure 5.1 (bottom) can *additionally* also consist of unlabeled examples, which are relatively easier to obtain.

The reason for having the input share the same subspace as the task parameters can be explained using a Representer theorem (Schölkopf et al., 2001) argument: write the solution of a regularized loss function as: $\theta = \sum_i \alpha_i \mathbf{x}_i$ (assume a linear kernel). Now, if we write each input vector $\mathbf{x}_i$ as a combination of basis vectors ($\mathbf{Z}\mathbf{a}_i + \epsilon_i$), then (after rearranging the coefficients) one can also write the task parameter θ as a combination of the same basis vectors defined by $\mathbf{Z}$. Therefore, it makes sense to have both $\mathbf{X}$ and Θ share the same subspace.

5.4 Inference

We take a fully Bayesian approach for inference in this model. Inference is akin to the Gibbs sampler for the IBP (Ghahramani et al., 2007), except for the following differences:

- The matrix **Z** is no longer a binary matrix but is expressed as an element-wise product of the binary matrix **B** and the real-valued matrix **V**. So both **B** and **V** need to be sampled in conjunction in our model.

- The latent variable Θ acts as the "data" and therefore needs to be sampled from its posterior $P(\Theta|\mathcal{D}, \mathbf{B}, \mathbf{V}, \mathbf{A}_\theta)$ where $\mathcal{D} = \{(\mathbf{x}_1, y_1^m), \ldots, (\mathbf{x}_N, y_N^m)\}, (m = [1, \ldots, M])$ denotes the actual data the model has access to.

Inference in our model is done using Gibbs sampling with a few Metropolis-Hastings steps. The sampler draws posterior samples of **B**, **V**, $\mathbf{A}_\theta$, Θ, and the remaining hyperparameters of the model. Here, we describe the sampling equations for all latent variables in our basic model. Sampling the hyperparameters (α, σ_v, etc.) is straightforward and we skip it due to the space limitation.

5.4.1 Sampling B

Sampling the binary IBP matrix **B** consists of sampling existing dishes, proposing new dishes and accepting or rejecting them based on the acceptance ratio in the associated M-H step. For sampling existing dishes, an entry in $\mathbf{B}$ is set as 1 according to $p(B_{ik} = 1|\Theta, B_{-ik}, \mathbf{V}, \mathbf{A}_\theta, \mathbf{\Psi}) \propto \frac{m_{-i,k}}{D} p(\Theta|\mathbf{B}, \mathbf{V}, \mathbf{A}_\theta, \mathbf{\Psi})$ whereas it is set as 0 according to $p(B_{ik} = 0|\Theta, B_{-ik}, \mathbf{V}, \mathbf{A}_\theta, \mathbf{\Psi}) \propto \frac{D-m_{-i,k}}{D} p(\Theta|\mathbf{B}, \mathbf{V}, \mathbf{A}_\theta, \mathbf{\Psi})$. $m_{-i,k} = \sum_{j \neq i} B_{jk}$ is how many other customers chose dish k.

For sampling new dishes, we use an M-H step where we simultaneously propose $\mathbf{J} = (K^{new}, \mathbf{V}^{new}, \mathbf{A}_\theta^{new})$ where $K^{new} \sim Poisson(\alpha/D)$. We accept the proposal with an acceptance probability given by $a = \min\{1, \frac{p(rest|\mathbf{J}^*)}{p(rest|\mathbf{J})}\}$. Here, $p(rest|\eta)$ is the probability of the data given parameters η. We propose $\mathbf{V}^{new}$ from its prior (Gaussian) but, for faster mixing, we propose $\mathbf{A}_\theta^{new}$ from its posterior (a Gaussian).

5.4.2 Sampling V

We sample the real-valued matrix **V** from its posterior:

$$p(V_{i,k}|\Theta, \mathbf{B}, \mathbf{A}_\theta, \mathbf{\Psi}) \sim \mathcal{N}or(V_{i,k}|\mu_{i,k}, \Sigma_{i,k})$$

where $\Sigma_{i,k} = (\sum_{n=1}^{N} \frac{A_{\theta k,n}^2}{\Psi_i} + \frac{1}{\sigma_v^2})^{-1}$ and $\mu_{i,k} = \Sigma_{i,k}(\sum_{n=1}^{N} A_{\theta k,n}\Theta_{i,k}^*)\Psi_g^{-1}$. We define $\Theta_{i,k}^* = \Theta_{i,n} - \sum_{l=1,l\neq k}^{K}(B_{i,l}V_{i,l})A_{\theta l,n}$. The hyperparameter σ_v on **V** has an inverse-gamma prior and the posterior also has the same form.

5.4.3 Sampling $\mathbf{A}_\theta$

We sample for $\mathbf{A}_\theta$ from its posterior $p(\mathbf{A}_\theta|\Theta, \mathbf{B}, \mathbf{V}, \mathbf{\Psi}) \sim \mathcal{N}or(\mathbf{A}_\theta|^-, \mathbf{\Sigma})$ where $^- = \mathbf{Z^T}(\mathbf{ZZ^T} + \mathbf{\Psi})^{-1}\mathbf{\Theta}$ and $\mathbf{\Sigma} = \mathbf{I} - \mathbf{Z^T}(\mathbf{ZZ^T} + \mathbf{\Psi})^{-1}\mathbf{Z}$, where $\mathbf{Z} = \mathbf{B} \odot \mathbf{V}$

5.4.4 Sampling Θ

The posterior for Θ can be written as $P(\Theta|\mathcal{D}, \mathbf{B}, \mathbf{V}, \mathbf{A}_\theta) \propto P(\mathbf{Y}|\mathbf{X}^T\Theta)P(\Theta)$. The prior on Θ is a Gaussian $\mathcal{N}or((\mathbf{B} \odot \mathbf{V})\mathbf{A}_\theta, \Psi)$. For the likelihood term, there are 2 cases. For regression, the likelihood $P(Y|\mathbf{X}^T\Theta)$ is Gaussian, so the posterior is available in closed form and is easy to sample from. Specifically, the posterior $P(\Theta|\mathcal{D}, \mathbf{B}, \mathbf{V}, \mathbf{A}_\theta)$ is a Gaussian $\mathcal{N}or(\mu_\theta, \Sigma_\theta)$ where

$$\begin{aligned} \mu_\theta &= \Sigma_\theta(\Psi^{-1}(\mathbf{B} \odot \mathbf{V})\mathbf{A}_\theta + \beta\mathbf{X}^T\mathbf{Y}) \\ \Sigma_\theta^{-1} &= \Psi^{-1} + \beta\mathbf{X}^T\mathbf{X} \end{aligned}$$

where β is the precision (inverse variance) of the Gaussian likelihood term $P(Y|\mathbf{X}^T\Theta)$.

For classification however, the likelihood is no longer Gaussian, so we lose conjugacy. There are several ways to deal with this. One way is to use Laplace approximation to the posterior (Bishop, 2006). Another possibility is to use the variational method proposed in (Jaakkola and Jordan, 1996) to approximate a non-Gaussian likelihood by a Gaussian one. We instead use another approach based on the auxiliary-variable-based Gibbs sampler for logistic regression (Holmes and Held, 2006), which is more appropriate in the Gibbs sampling scheme we employ.

The auxiliary variable sampler (Holmes and Held, 2006) for logistic regression associates with each response $y_i \in \{0, 1\}$ an auxiliary variable $\tilde{y}_i = \mathbf{x}_i^T\theta + \epsilon_i$ with $\epsilon \sim \mathcal{N}or(0, \lambda_i)$, such that $y_i = 1$ if $\tilde{y}_i > 0$, and 0 otherwise. λ_i is assigned a Kolmogorov-Smirnov distribution. With a normal prior $\mathcal{N}or(b, v)$ on θ, the posterior on θ is still a Gaussian:

$$\begin{aligned} \theta|\tilde{\mathbf{y}}, \lambda &\sim \mathcal{N}or(\mu_\theta, \Sigma_\theta) \\ \mu_\theta &= \Sigma_\theta(v^{-1}b + \beta\mathbf{X}^T\mathbf{W}\tilde{\mathbf{y}}) \\ \Sigma_\theta^{-1} &= (v^{-1} + \beta\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1} \\ \mathbf{W} &= diag(\lambda_1, \ldots, \lambda_N), \quad \tilde{\mathbf{y}} = [\tilde{y}_1, \ldots, \tilde{y}_N]' \end{aligned}$$

where the posterior over the auxiliary variables $\tilde{y}_i$ is a truncated normal, which can be sampled from using standard techniques.

$$\tilde{y}_i|\theta, \mathbf{x}_i, y_i, \lambda_i \sim \begin{cases} \mathcal{N}or(\mathbf{x}_i^T\theta, \lambda_i)\mathbb{I}(\tilde{y}_i > 0) & \text{if } y_i = 0 \\ \mathcal{N}or(\mathbf{x}_i^T\theta, \lambda_i)\mathbb{I}(\tilde{y}_i \leq 0) & \text{if } y_i \neq 0 \end{cases}$$

and in our case, the mean and covariance on the normal prior over Θ are given by $b = (\mathbf{B} \odot \mathbf{V})\mathbf{A}_\theta$ and $v = \mathbf{\Psi}$, respectively.

5.4.5 Sampling in the Augmented Model

The sampling steps in our augmented model are essentially the same as in the basic model, except that we replace the $D \times M$ matrix Θ by the $D \times (M + N)$ matrix $[\Theta\ \mathbf{X}]$. As in the basic model, Θ still needs to be sampled as above, whereas $\mathbf{X}$ stays fixed and does not have to be sampled.

We note here that although a fully Bayesian solution can be slow with data having a large number of features (since each feature corresponds to a customer in the IBP model), one may address this by using a number of recently proposed alternatives to Gibbs sampling (Doshi and Ghahramani, 2009a) for IBP that can be as much as an order of magnitude faster.

5.5 Prediction

Having learned the task parameters Θ, we use these to make predictions on the test data. For the test data $\mathbf{x}$ of the mth task, the prediction can be written as

$$p(y|\mathbf{x}) = \int p(y|\mathbf{x}, \theta_m)p(\theta_m|\mu_m, \Sigma_m)d\theta_m$$

which is essentially averaging over the predictions made by each of the posterior samples of θ_m, where μ_m and Σ_m are the mean and covariance parameters of the mth task. Since the posterior averaging can be computationally expensive, it can also be replaced by $\hat{\theta}_m$, the MAP estimate of θ_m. Prediction for $\mathbf{x}$ then simply requires plugging in the MAP estimate: $p(y|\mathbf{x}) = p(y|\mathbf{x}, \hat{\theta}_m)$.

5.6 Experiments

We present our experimental results on two real-world multilabel classification datasets (Yeast and Scene) from the UCI repository, comparing our models against independently trained Bayesian logistic regression, the pooling-based approach, and another state-of-the-

art Multitask Learning baseline. The Yeast dataset consists of 1500 training and 917 test examples, each having 103 features. The number of labels (or tasks) per example is 14. The Scene dataset consists of 1211 training and 1196 test examples, each having 294 features. The number of labels per example for this dataset is 6. We use the following baselines:

- **LR**: Independent Bayesian logistic regression
- **pool**: Pooling all data and learning a single model
- **yaxue**: The matrix stick-breaking-process-based Multitask Learning model proposed in (Xue et al., 2007a)

In the experimental results (Figure 5.2 and Table 5.1) , we refer to our basic model as *model-1*, and the augmented model with input data as *model-2*. Note that all the multitask approaches compared here use Logistic Regression as the base classifier. We use overall accuracy, F1-Macro and F1-Micro (Yang, 1997), and AUC (Area Under ROC Curve) as the performance metrics. The Gibbs samplers used in Bayesian logistic regression, the method of (Xue et al., 2007a), and both of our models were run for 1000 iterations. Results on both datasets, with full training dataset used, are shown in Table 5.1.

As the results show, both our models perform better than independently trained Bayesian logistic regression, which completely ignores the task relatedness. When compared across all the baselines, we obtain consistent improvements for almost all of the scores. Also, the pooling-based approach, surprisingly, ends up hurting the overall performance here, suggesting that a simple pooling may not always be a good idea. Furthermore, our augmented model (model-2) does best overall, suggesting that incorporating the input data in learning the predictor subspace defined by $\mathbf{Z}$ indeed helps in learning the task parameters even better, especially when the number of tasks is small (which is indeed the case with Yeast and Scene datasets). We also investigated the effect of varying the dataset size starting with a small number of training examples and incrementing slowly. The results on the Scene data are shown in Figure 5.2. We see that both our models do considerably better than Bayesian logistic regression learned separately for each task, especially when the training set size is small. Moreover, the augmented model does best, implying that the including the input data while learning the predictor subspace indeed helps. We also observe that even with a

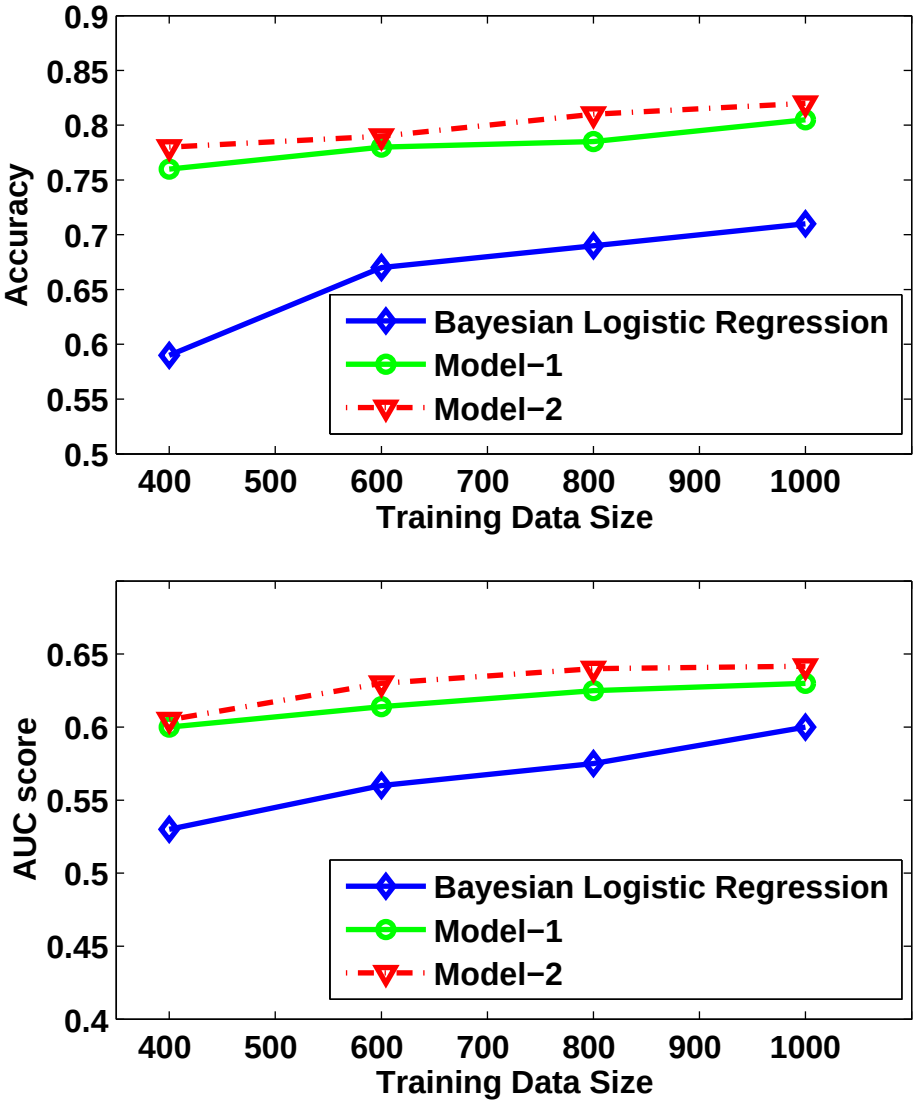


Figure 5.2. Performance comparison between both our Multitask Learning models, and Bayesian logistic regression trained separately for each task. Top: Accuracy with varying training data size. Bottom: AUC score with varying training data size.

Table 5.1. Comparison of Bayesian logistic regression, pooling approach, kernel stick--breaking approach (**yaxue**), our basic model (model-1), and our augmented model (model-2), for two multilabel datasets. Bold face implies the best performance. Results are averaged over 10 runs with different initializations.

Model	**Yeast**				**Scene**			
	Acc	F1-macro	F1-micro	AUC	Acc	F1-macro	F1-micro	AUC
LR	0.5047	0.3415	0.3828	0.5049	0.7362	0.3132	0.3173	0.6153
pool	0.4983	0.3497	0.3910	0.5112	0.7862	0.2842	0.3012	0.5433
yaxue	0.5106	0.3897	0.4022	0.5105	0.7765	0.2669	0.2816	0.5603
Model-1	0.5212	0.3631	0.3901	0.5244	0.7756	0.3153	**0.3242**	0.6325
Model-2	**0.5424**	**0.3946**	**0.4112**	**0.5406**	**0.7911**	**0.3214**	0.3226	**0.6416**

very small training dataset, performance of both our models is reasonably close to optimal, suggesting that it is possible to learn reliably even with a small amount of data. Logistic regression, on the other hand, falls behind by quite a lot when the amount of training data is small. It begins to catch up with our models but they still do better, even with the full data. This evidence supports the model assumption that an underlying task space is shared across all tasks and learning the task parameters with this assumption indeed improves performance of all the tasks.

5.7 Related Work

The recent interest in learning a set of related tasks has spurred a range of work in Multitask Learning with different notions of task relatedness being proposed with varying degrees of success. One of the earliest works on Multitask Learning includes sharing of the hidden layers in neural networks to share information across tasks (Caruana, 1997). Other prominent approaches include tasks based on the assumption of being generated from an IID space (Baxter, 2000), learning multiple tasks in a Bayesian setting using a hierarchical prior over the task space (Daumé III, 2009, Xue et al., 2007b), sharing parameters of Gaussian processes (Lawrence and Platt, 2004), sharing a common structure on the predictor space (Ando and Zhang, 2005), and structured regularization in kernel methods (Evgeniou et al., 2006), among others. Extending the task-clustering model of (Xue et al., 2007b), the matrix stick-breaking process (MSBP) model proposed in (Xue et al., 2007a) (the **yaxue** model used as one of our baselines) allows separate clustering and borrowing of information for the different feature components. This can be important if we

expect the tasks to be more closely related for some features than for others.

Another notion of task relatedness assumes that the *data* from related tasks share an underlying low-dimensional feature space (Ji et al., 2008) that essentially captures the task relatedness. This is in contrast with our proposed approach where we assume that the task-parameters share a latent low-dimensional subspace. Note, however, that our model-2 additionally also performs dimensionality reduction of the input data, sharing information across tasks. Thus, one may as well use this alternate feature representation of data to learn the multiple tasks

Structurally, our basic model (model-1) is most similar to the one proposed in (Zhang et al., 2006). Their model, however, fixes the number of task basis vectors to the number of tasks, whereas our model automatically infers this. In addition to automatically determining the intrinsic task dimensionality, an IBP prior on $\mathbf{Z}$ (via the binary matrix $\mathbf{B}$) also allows us to discover any underlying sparsity of the task basis space. Furthermore, the model proposed in (Zhang et al., 2006) uses EM for inference whereas we propose a fully Bayesian solution for our proposed models.

Another closely related work similar in spirit to our model is the *semiparametric* latent factor model (Teh et al., 2005) for regression. This model makes use of a set of Gaussian Processes (GP), linearly mixed to capture the possible dependencies among the response variables. The difference between this model and ours is that the former assumes a linear mixing process in the instance space whereas we assume it to hold in the predictor space.

Finally, the idea of encouraging sparsity of the task basis space is also in line with recent work on taking advantage of sparsity in Multitask Learning. (Lounici et al., 2009) recently proposed a model based on grouped LASSO, which enforces sparsity directly on regression vectors. Our proposed model addresses the issue of sparsity in a somewhat different but complementary manner as our model assumes that the task basis vectors are sparse.

5.8 A Mixture of Subspaces Model for Multitask Learning

Our Factor-Analysis-based predictor subspace model also admits natural extensions to more complex settings. In this section, we briefly outline how a nonlinear manifold underlying the task parameters can be learned by extending our basic linear subspace model. Note that a *single* shared *linear* subspace can be somewhat restrictive due to

two reasons: a) when there are outlier tasks for which it is unreasonable to assume the same shared subspace as the other relevant tasks, and b) when underlying task parameter subspace is a *nonlinear* manifold. Our predictor subspace model can be easily generalized to deal with such issues by assuming a mixture of subspaces model. We describe this generalization in more detail in the next chapter.

5.9 Conclusion

In this chapter, we proposed a nonparametric, fully Bayesian, probabilistic framework to learn the latent shared subspace of a set of related tasks. The shared subspace captures the task relatedness in a manner that each task parameter (i.e., the weight vector of a classification/regression model) can be treated as a linear combination of a set of basis tasks constituting this subspace. More importantly, we do not restrict the intrinsic dimensionality of this subspace to an *a priori* fixed number, but discover it automatically. An additional advantage of our proposed model is that our prior promotes sparsity of the basis space, leading to LASSO style notion of model sparsity. Furthermore, we also propose an extension to the model, which can incorporate inputs from labeled data (and, potentially, also inputs from *additional* unlabeled data), to more reliably learn the model when the number of tasks is small. Our model is also easily extendable to a mixture of subspaces setting as described in Section 5.8, which can be appropriate for cases where the task parameters lie on a nonlinear manifold, and/or if there are outlier tasks. We believe that similar flexible models lead to effective capturing of task relatedness, and can result in improved model performance in Multitask Learning problems.

CHAPTER 6

NONPARAMETRIC MIXTURE OF SUBSPACES FOR MULTITASK LEARNING

In this chapter, we generalize the model presented in the previous chapter and show that this generalization leads to a very flexible Multitask Learning model that can adapt its task relatedness assumptions on-the-fly based on the data. This is especially desirable because an incorrect assumption of how the tasks relate may even hurt Multitask Learning performance. We propose a probabilistic framework for grouping tasks based on their similarities. We further assume that, within each group, a task can be expressed as a sparse linear combination of a set of *basis tasks* (i.e., we have a sparse-coding-based representation of tasks within each group).

6.1 Introduction

Motivated by the desire of flexible modeling of task relatedness, we propose a nonparametric Bayesian MTL model by representing the task parameters (e.g., the weight vectors for logistic regression models) as being generated from a nonparametric mixture of nonparametric factor analyzers. Parameters are shared only between tasks in the same cluster and, within each cluster, across a linear subspace that regularizes what is shared. Moreover, by virtue of this being a nonparametric model, various existing MTL models result as special cases of our model; for example, the weight vectors are drawn from a single shared Gaussian prior, or form clusters (equivalently, generated from a mixture of Gaussians), or live close to a subspace, etc. Our model can automatically interpolate between these assumptions as needed, providing the best fit to the given MTL problem.

In addition to offering a general framework for Multitask Learning, our proposed model also addresses several shortcomings of commonly used MTL models. For example, task

clustering (Xue et al., 2007b), which fits a full-covariance Gaussian mixture model over the weight vectors, is prone to overfitting on high-dimensional problems as the number of learning tasks is usually much smaller than the dimensionality, making it difficult to estimate the covariance matrix. A model based on mixtures of factor analyzers, like ours, can deal with this issue by adaptively estimating the dimensionality of each component, using less parameters than in the full rank case. Likewise, models based on task subspaces (Agarwal et al., 2010, Rai and Daumé III, 2010, Zhang et al., 2006) assume that the weight vectors of all the tasks live on or close to a *single* shared subspace, which is known to lead to negative transfer in the presence of outlier tasks. Our model, based on a mixture of subspaces, circumvents these issues by allowing different groups of weight vectors to live in different subspaces when grouping all together them would not fit the data well. One can also view our model as allowing the sharing of statistical strengths at two levels: (1) by exploiting the cluster structure, and (2) by additionally exploiting the subspace structure *within* each cluster.

In the context of MTL, since the task relatedness structure is usually unknown, the standard solution is to try many different models, covering many similarity assumptions, with many settings of complexity for each model, and choose the one according to some model selection criteria. In this work, we take a nonparametric Bayesian approach to this problem (using the Dirichlet Process and the Indian Buffet Process as building blocks) such that the appropriate MTL model capturing the correct task relatedness structure and the model complexity for *that* model will be learned in a data-driven manner side-stepping the model selection issues.

6.2 Mixture of Factor Analyzers-based Generative Model for MTL

Our proposed model assumes that the parameters (i.e., the weight vector) of each task are sampled from a mixture of factor analyzers (Ghahramani and Beal, 2000). Note that our model is defined over *latent* weight vectors whereas the standard mixture of factor analyzers is commonly defined to model *observed data*.

We assume that we are learning T related tasks, where each task is represented by a weight vector $\theta_t \in \mathbb{R}^D$ that is assumed to be sampled from a mixture of F factor analyzers

where each factor analyzer consists of $K \leq \min\{T, D\}$ factors (note: our model also allows each factor analyzer to have a different number of factors). Here, D denotes the number of features in the data. Each task is a set of X and Y values, and each Y is assumed to be generated from the corresponding X value and task weight vector. In our model, the weight vector θ_t for task t is generated by first sampling a factor analyzer (defined by a mean task parameter $\mu_t \in \mathbb{R}^D$ and a factor loading matrix $\Lambda_t \in \mathbb{R}^{D \times K}$) using the DP, and then generating θ_t using *that* factor analyzer. In equations, this can be written as $\theta_t = \mu_t + \Lambda_t f_t + \varepsilon_t$.

The weight vector θ_t is a *sparse* linear combination of K *basis vectors* represented by the columns of Λ_t (each column is a "basis task"). The combination weights are given by $f_t \in \mathbb{R}^K$, which we represent as $s_t \odot b_t$, where s_t is a real-valued vector and b_t is a binary valued vector, both of size K. Our model uses a Beta-Bernoulli/IBP prior on b_t to determine K, the number of factors in each factor analyzer. The $\{\mu_t, \Lambda_t\}$ pair for each task is drawn from a DP, also giving the tasks a clustering property, and there will be a finite number $F \leq T$ of *distinct* factor analyzers. Finally, $\varepsilon_t \sim \mathcal{N}or(0, \frac{1}{\sigma^2}\mathbf{I})$ represents task-specific noise.

Figure 6.1 shows a graphical depiction of our model and Figure 6.2 shows the generative story for the linear regression case . The DP base measure G_0 is a product of two Gaussian priors for μ_t, Λ_t. In our nonparametric Bayesian model, F and K need not be known *a priori*; these are inferred from the data.

For classification, the only change is that the first line in the generative model becomes $Y_{t,i} \sim \mathcal{B}er(sig(\theta_t \cdot X_{t,i}))$, where $sig(x) = \frac{1}{1+\exp(-x)}$ is the logistic function and $\mathcal{B}er$ is the Bernoulli distribution.

A number of existing Multitask Learning models arise as special cases of our model as it nicely interpolates between some different and useful scenarios, depending on the actual inferred values of F and K, for a given Multitask Learning dataset:

- **Shared Gaussian Prior** (F=1, K=0): (Chelba and Acero, 2006). This corresponds to a single factor analyzer modeling either a diagonal or full-rank Gaussian as the prior.
- **Cluster-based Assumption** ($F > 1, K$=0): (Jacob and Bach, 2008, Xue et al., 2007b). This corresponds to a mixture of identity-covariance or full-rank Gaussians

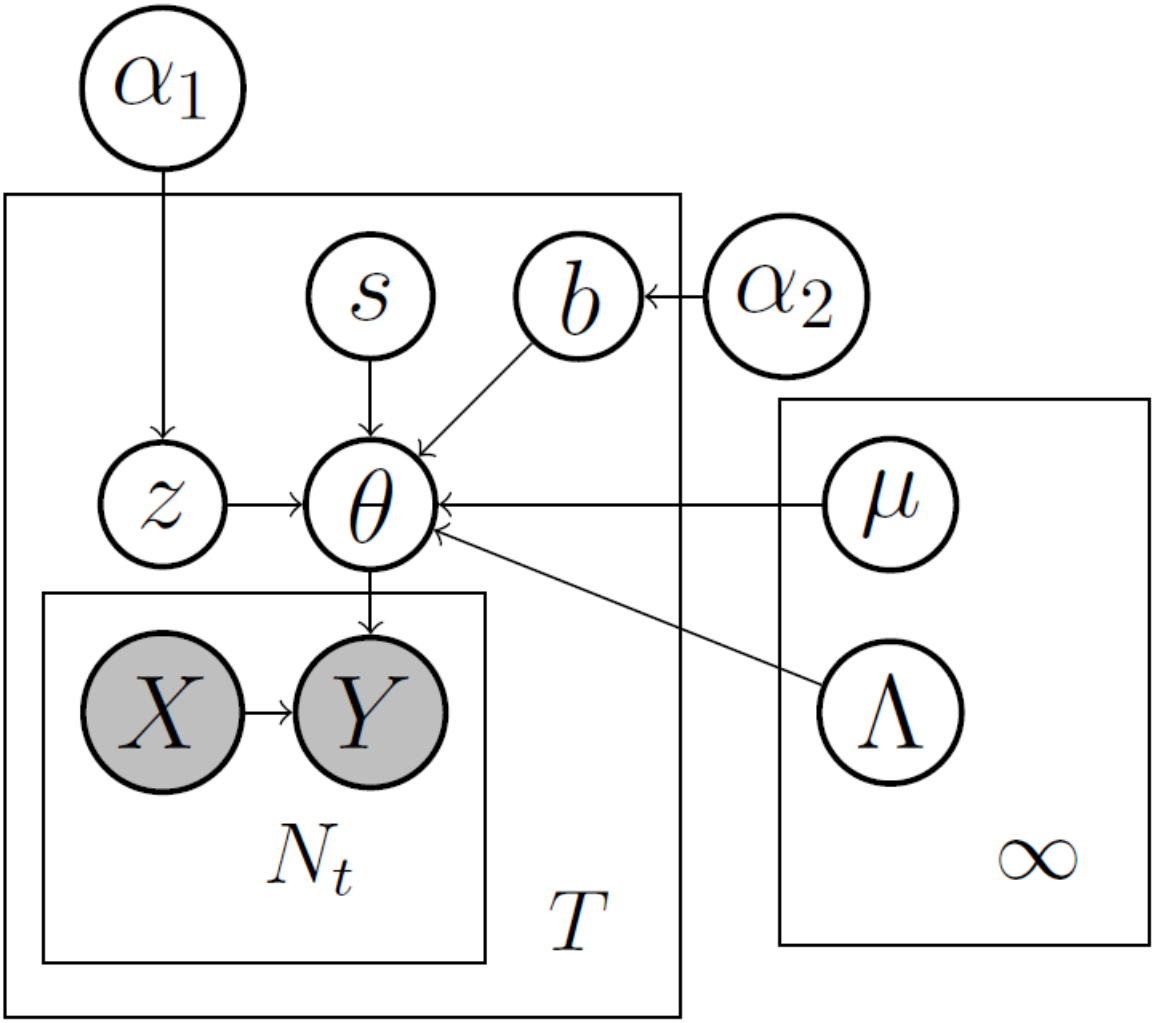


Figure 6.1. A graphical depiction of our model. The task parameters θ are sampled from a DP-IBP mixture and used to generate the Y values.

$$
\begin{aligned}
Y_{t,i} &\sim \mathcal{N}or(\theta_t^T X_{t,i}, \mathbf{I}) \\
\theta_t &\sim \mathcal{N}or(\mu_t + \Lambda_t \cdot (s_t \odot b_t), \frac{1}{\sigma^2}\mathbf{I})) \\
\mu_t, \Lambda_t &\sim G \quad s_t \sim \mathcal{N}or(0, \mathbf{I}) \quad b_{kt} \sim \mathcal{B}er(\pi_k) \\
G &\sim \mathcal{DP}(\alpha_1, G_0) \quad \pi_k \sim \mathcal{B}et(\alpha_2/K, 1)
\end{aligned}
$$

Figure 6.2. The hierarchical model. The cluster indicator variable z is implicit in the draw from the DP. The Beta-Bernoulli draw for b_{kt} approximates the IBP for large K (actual K will be inferred from the data).

as the prior.

- **Linear Subspace Assumption** ($F{=}1, K < D$): (Rai and Daumé III, 2010, Zhang et al., 2006). This corresponds to a single factor analyzer with less than full rank. Note that this is also equivalent to the matrix $\Theta = \{\theta_1, \ldots, \theta_T\}$ being a rank-K matrix (Argyriou et al., 2007).
- **Nonlinear Manifold Assumption**: A mixture of linear subspaces allows modeling a nonlinear subspace (Chen et al., 2010) and can capture the case when the weight vectors live on a nonlinear manifold (Agarwal et al., 2010, Ghosn and Bengio, 2003). Moreover, in our model, the manifold's intrinsic dimensionality can be different in different parts of the ambient space (since we do not restrict K to be the same for each factor analyzer).

Our nonparametric Bayesian model can interpolate between these cases as appropriate for a given dataset, without changing the model structure or hyperparameters. From a nonprobabilistic analogy, our model can be seen as doing dictionary learning/sparse coding (Aharon et al., 2010) over the *latent* weight vectors (albeit, using an *undercomplete* dictionary setting since we assume $K \leq \min\{T, D\}$). The model learns M dictionaries of basis tasks (one dictionary per group/cluster of tasks, and M inferred from the data) and tasks within each cluster are expressed as a sparse linear combination of elements from *that* dictionary. Our model can also be generalized further; e.g., by replacing the Gaussian prior on the low-dimensional latent task representations $s_t \in \mathbb{R}^K$ by a prior of the form $P(s_{t+1}|s_t)$, one can even relax the exchangeability assumption of tasks within each group, and have tasks that are evolving with time.

6.2.1 Variational Inference

As this model is infinite and combinatorial in nature, exact inference is intractable and sampling-based inference may take too long to converge (Blei and Jordan, 2006, Doshi-Velez et al., 2009b). Hence, we employ a variational mean-field algorithm to perform inference in this model. To do so, we lower-bound the marginal log-probability of Y given X using a fully factored approximating distribution Q over the model parameters $\theta, \mu, \Lambda, z, b, s$:

$$\begin{aligned}\log P(Y|X) &= \log E_P[P(Y|X,\theta,\mu,\Lambda,z,b,s)] \\ &\geq E_Q[\log P(Y|X)] \\ &\quad -E_Q[\log Q(Y|X)].\end{aligned}$$

To do so, we approximate the DP and the IBP with a tractable distribution Q. For the DP, we use a finite stick-breaking distribution, based on the infinite stick-breaking representation of the DP (Blei and Jordan, 2006). In this representation, we introduce, for each θ_t, a multinomial random variable z_t that indexes the infinite set of possible mixture parameters μ and Λ. The z_t vector is nonzero on its i-th component with probability $\phi_i \prod_{j<i}(1-\phi_j)$, where ϕ is an infinite set of independent $\mathcal{B}et(1,\alpha_1)$ random variables ($\mathcal{B}et$ is the Beta distribution). A finite approximation to the DP is obtained by setting a given ϕ_i to 1, which sets the probability of z_j for $j > i$ necessarily to 0. While there is a similar stick-breaking construction to the IBP (Teh et al., 2007a), it is not in the exponential family and requires complicated approximations, so we represent the IBP by its finite Beta-Bernoulli approximation (Doshi-Velez et al., 2009b).

The distribution we are approximating then (for the linear regression case) is shown in Figure 6.3 (top). The stick-breaking distribution SBP, which is the prior for z_t, is such that $P(z_t{=}i) = \phi_i \prod_{j<i}(1-\phi_j)$.

In our variational distribution, we set the number of factor analyzers in the truncated stick-breaking representation to a hyperparameter F and the number of factors in each such analyzer to a truncation level hyperparameter K. After inference, if the truncation levels are set high enough, most factor analyzers (and factors within each factor analyzer) will not be used, effectively approximating the property of the infinite model that only a small finite number of components is ever used to model a finite data set. It is worthwhile to note that while the solution found by the variational approximation is necessarily finite and with complexity bounded by the truncation parameters, it will still implicitly perform model selection. Therefore, more often than not, it will concentrate most of its posterior mass on models with less complexity than the truncation parameters suggest. (Ishwaran and James, 2001) present two theorems to help choose these truncation levels, as using smaller values of F and K (particularly K, as the update equations are quadratic in K) can lead to significant savings of computing time (in our experiments, we simply set these to $\min\{D,T\}$, which we found to be sufficient).

$$
\begin{aligned}
Y_{t,i} &\sim \mathcal{N}or(\theta_t^T X_{t,i}, \mathbf{I}). \\
\theta_t &\sim \mathcal{N}or(\mu_{z_t} + \Lambda_{z_t}(s_{t,z_t} \odot b_{t,z_t}), \frac{1}{\sigma^2}\mathbf{I}) \\
\mu_f &\sim \mathcal{N}or(0, \mathbf{I}), \quad \Lambda_{f,k} \sim \mathcal{N}or(0, \mathbf{I}) \\
s_{t,f} &\sim \mathcal{N}or(0, \mathbf{I}), \quad b_{t,f,k} \sim \mathcal{B}er(\beta_{f,k}) \\
z_t &\sim SBP(\phi), \quad \beta_{f,k} \sim \mathcal{B}et(\alpha_2/K, 1) \\
\phi_f &\sim \mathcal{B}et(1, \alpha_1)
\end{aligned}
$$

$$
\begin{aligned}
Q(\theta_t) &= \mathcal{N}or(\nu_{\theta_t}, \mathbf{I}) \\
Q(\mu_f) &= \mathcal{N}or(\nu_{\mu_f}, \mathbf{I}), \quad Q(\Lambda_f) = \mathcal{N}or(\nu_{\Lambda_f}, \mathbf{I}) \\
Q(s_{t,f}) &= \mathcal{N}or(\nu_{s_{t,f}}, \mathbf{I}), \quad Q(b) = \mathcal{B}er(\nu_b) \\
Q(z_t = i) &= \nu_{z_{t,i}}, \quad Q(\beta) = \mathcal{B}et(\rho_1, \rho_2) \\
Q(\phi) &= \mathcal{B}et(\gamma_1, \gamma_2)
\end{aligned}
$$

Figure 6.3. Variational approximation. Top: the distribution being approximated. Bottom: Our approximating Q distribution (note: $P(Y|\theta)$ is lower-bounded directly)

Our approximating Q distribution is shown in Figure 6.3 (bottom). For the linear regression case, we treat $P(Y|\theta)$ by lower-bounding it directly, without introducing an approximating distribution for Y. In the case of logistic regression, we use the lower bound by (Jaakkola and Jordan, 1996) that allows us to integrate out the logistic function.

Apart from approximating the DP with the truncated stick-breaking prior, approximating the IBP with a set of symmetric, finite Beta distributed variables, and lower-bounding the logistic function with a quadratic, all the computations involved in deriving the variational lower bound are straightforward exponential-family computations. Note that for Q, we could use more general covariances instead of the identity matrices. In practice, we found that this did not improve classification performance, and it would imply a significantly higher computational cost. Another less expensive option, however, would be to use the same hyperparameter for each feature, i.e., a spherical (instead of diagonal) covariance $\tau^2\mathbf{I}$, which would require optimizing w.r.t. a single hyperparameter τ. The variational parameter updates are[1]:

[1]The complete derivations are provided in the Appendix.

$$
\begin{aligned}
\gamma_{f,1} &= 1 + \sum_t \nu_{z_{t,f}} \\
\gamma_{f,2} &= \alpha_1 + \sum_t \sum_{j>f} \nu_{z_{t,j}} \\
\nu_{z_{t,f}} &\propto \exp\Big(\digamma(\gamma_{f,1}) - \digamma(\gamma_{f,1} + \gamma_{f,2}) \\
&\quad + \sum_{j<f} (\digamma(\gamma_{j,2}) - \digamma(\gamma_{j,1} + \gamma_{j,2})) \\
&\quad + E_Q[\log P(\theta_t | z_t = f)] \Big) \\
\rho_{f,k,1} &= \frac{\alpha_2}{K} + \sum_t \nu_{b_{t,f,k}}, \quad \rho_{f,k,2} = 1 + \sum_t (1 - \nu_{b_{t,f,k}}) \\
\nu_{b_{t,f,k}} &= sig\Big(\digamma(\rho_{f,k,1}) - \digamma(\rho_{f,k,2}) \\
&\quad + \sigma \nu_{z_{t,f}} \Big(\Big[\nu_{\theta_t} - \nu_{\mu_f} - (\nu_{s_{t,i}} + 1) \nu_{\Lambda_{f,i}} \\
&\quad - \sum_{j \neq i} \nu_{s_{t,j}} \nu_{b_{t,f,j}} \nu_{\Lambda_{f,j}} \Big]^T \nu_{\Lambda_{f,i}} \nu_{s_{t,i}} - \frac{D}{2} \nu_{s_{t,i}}^2 - \frac{DF}{2} \Big) \Big) \\
\nu_{s_{t,i}} &= (1 + \sigma \nu_{z_{t,f}} \nu_{b_{t,f,i}} (D + ||\nu_{\Lambda_{f,i}}||^2))^{-1} \\
&\quad \nu_{z_{t,f}} \sigma \Big(\Big(\nu_{\theta_t} - \nu_{\mu_f} 0.5 \sum_{j \neq i} \nu_{s_{t,f,j}} \nu_{b_{t,f,j}} \nu_{\Lambda_{f,j}} \Big)^T \nu_{\Lambda_{f,i}} \nu_{b_{t,f,i}} \Big) \\
\nu_{\mu_f} &= \frac{\sum_t \nu_{z_{t,f}} \sigma (\nu_{\theta_t} - \nu_{\Lambda_f} (\nu_{s_{t,f}} \odot \nu_{b_{t,f}}))}{1 + \sigma \sum_t \nu_{z_{t,f}}} \\
\nu_{\Lambda_{f,i}} &= \Big(1 + \sigma \sum_t \nu_{z_{t,f}} \nu_{b_{t,f,i}} (1 + \nu_{s_{t,f,i}}^2) \Big)^{-1} \\
&\quad \sigma \sum_t \nu_{z_{t,f}} \nu_{s_{t,f,i}} \nu_{b_{t,f,i}} \Big(\nu_{\theta_t} - \nu_{\mu_f} - \frac{1}{2} \sum_{j \neq i} \nu_{s_{t,f,j}} \nu_{b_{t,f,j}} \nu_{\Lambda_{f,j}} \Big)
\end{aligned}
$$

In the above, $\digamma$ denotes the digamma function. While it is possible to update ν_{θ_t} analytically, the update requires inverting a matrix, and in our experiment, this matrix was often ill-conditioned, so we updated ν_{θ_t} by optimizing the lower bound with the L-BFGS-B optimizer (Zhu et al., 1997). The optimizer is run until convergence at each iteration, warm-started with the previous value. We note that it could be replaced by any other optimizer, including gradient methods, with no changes in the above equations.

For regression, the gradient of the lower bound with respect to ν_{θ_t} is

$$
\nabla L(\nu_{\theta_t}) = \sigma \sum_f \nu_{z_{t,f}} \left(\nu_{\theta_t} - \nu_{\mu_f} - \nu_{\Lambda_f} (\nu_{s_{t,f}} \odot \nu_{b_{t,f}}) \right) + \sum_i^{N_t} \left(Y_{t,i} X_{t,i} - X_{t,i} X_{t,i}^T \nu_{\theta_t} \right).
$$

For classification, the gradient is similar, the main difference being that there is an extra factor in the $X_{t,i}X_{t,i}^T\nu_{\theta_t}$ term involving the variational parameter for the lower bound of the logistic function.

We also optimize the lower bound w.r.t the precision parameter σ to obtain an empirical Bayes estimate for $\frac{1}{\sigma}$:

$$\sum_t \sum_f \nu_{z_{t,f}} \left(\frac{||\nu_{\theta_t} - \nu_{\mu_f} - \nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}})||^2}{KDF} + \frac{\sum_i \nu_{b_{t,f,i}}(\nu^2_{s_{t,f,i}} + ||\nu_{\Lambda_{f,i}}||^2)}{KF} + \frac{1}{K} \right).$$

The hyperparameters α_1 and α_2 are held fixed and can be optimized by cross-validation. We initialize the inference process with ν_{θ_t} set to the maximum likelihood solution to each task's regression or classification problem. Then, we alternate updating all other parameters to convergence and updating ν_{θ_t} given the other parameters. The value of ν_{θ_t}, and hence the regression or classification accuracy, usually stabilizes after the first couple of iterations, and the only changes observed are further improvements to the lower bound. This matches behavior observed in (Ando and Zhang, 2005). All our experiments were run on three iterations.

6.3 Experiments

We present results on both synthetic and real-world datasets, and on linear regression and classification settings. As a sanity check to show that our model can learn the underlying latent task structures correctly, we generated a synthetic data consisting of 5 clusters of tasks. Each cluster consists of 10 binary classification tasks, having 100 examples each. We used a 50/50 split for train/test data. Each task is represented by a weight vector of length $D = 20$. Figure 6.4 (top) shows the true correlation structure of the tasks and Figure 6.4 (bottom) shows the recovered structure by our model: it correctly infers the correct number (5) of clusters. Our model resulted in a classification accuracy of 83.2%, whereas independently learned tasks resulted in an accuracy of 79.2%.

Our next set of experiments compare our model with a number of baseline methods on several synthetic and real-world multitask regression and multitask classification problems. Our baselines include:

- Independently Learned Tasks - **STL**: assumes the tasks are independent (no information sharing).

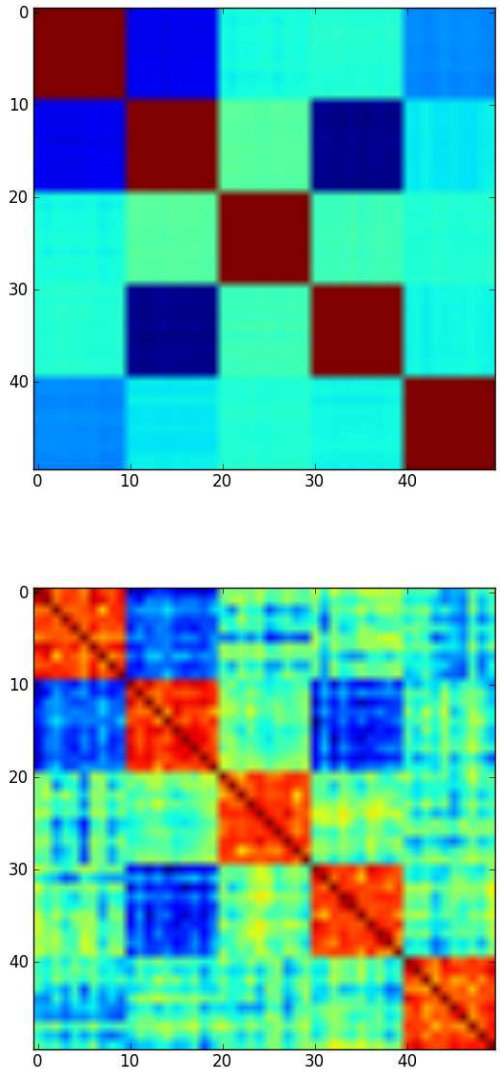


Figure 6.4. Synthetic Data. Top: Plot of the correlation matrix of the ground-truth weight vectors of the 50 tasks. Bottom: Inferred correlation matrix

- Multitask Feature Learning - **MTFL**: assumes the tasks share a common set of features (Argyriou et al., 2007).

- Shared Gaussian prior over the weight vectors - **PRIOR** (Chelba and Acero, 2006): assumes the tasks are drawn from a shared Gaussian prior with a unknown but fixed mean and covariance.

- Single shared subspace - **RANK** (Rai and Daumé III, 2010, Zhang et al., 2006): assumes the tasks live close to a linear subspace (also equivalent to the matrix of the weight vector being low-rank).

- DP mixture model-based task clustering - **DP-MTL** (Xue et al., 2007b): assumes the weight vectors are generated from a mixture model, each component being a full-rank Gaussian.

- **L**earning with **W**hom to **S**hare - **LWS** (Kang et al., 2011). It is an integer-programming-based method that learn the task grouping structure (with prespecified number of groups) and encourages the tasks within each group to share features.

Of these baselines, MTFL and LWS were used for regression problems only since the publicly available implementations are for regression. In the experiments, we would refer to our model as **MFA-MTL** (**M**ixture of **F**actor **A**nalyzers for **M**ulti**T**ask **L**earning). In all our experiments, we set the hyperparameters $\alpha_1 = 1$ and $\alpha_2 = 5$, as these values performed reasonably in preliminary experiments. The truncation level for the DP can be chosen to be equal to the number of tasks T, and for the IBP, to be the minimum of T and the number of features D in the data. This is often more than necessary and in most of our experiments, much smaller truncation levels were found to be sufficient.

For our multitask regression experiments, we compared MFA-MTL with STL, MTFL, and LWS (we skip the other baselines as they performed comparably or worse than MTFL and LWS). For this experiment, we used three datasets - one synthetic dataset used in (Kang et al., 2011), and two real-world datasets used commonly in the Multitask Learning literature: (1) **School**: This dataset consists of the examination scores of 15362 students from 139 schools in London. Each school is a task so there are a total of 139 tasks for this dataset. (2) **Computer**: This dataset consists of a survey of 190 students about the

chances of purchasing 20 different personal computers. There are a total of 190 tasks, 20 examples per task, and 13 features per example. For the synthetic data, we followed the similar procedure for train/test split as used by (Kang et al., 2011). For School and Computer datasets, we split the data equally into training and test set and further only used 20% of the training data (training set deliberately kept small as is often the case with Multitask Learning problems in practice). The average mean squared errors (i.e., across tasks) in predicting the responses by each method are shown in Table 6.1. As shown in Table 6.1, MFA-MTL outperforms the other baselines on all the datasets. Moreover, for the synthetic data, we found that it also inferred the number of task groups (3) correctly (the LWS baseline needs this number to be specified - we ran it with the ground truth). On the school and computer datasets, MFA-MTL outperforms STL and LWS and does slightly better than MTFL. For LWS on these two datasets, we report the best results as obtained by varying the number of groups from 1 to 20.

We next experiment with the classification setting. For this, we chose two datasets: (1) **Landmine**: The landmine detection dataset is a subset of the dataset used in the symmetric Multitask Learning experiment by (Xue et al., 2007b). It contains 19 classification tasks and the tasks are known to be clustered for this data. (2) **20ng:** We did the standard training/test split of 20 Newsgroups for Multitask Learning, following (Raina et al., 2006), and used a 50/50 split for the landmine data. The classification accuracies reported by our model and the various baselines on landmine and 20 Newsgroups datasets are shown in Table 6.2. As shown in Table 6.2, our method outperforms the various baselines. We note that 3 of them (PRIOR, RANK, and DP-MTL), which are methods proposed in prior work, are special cases of our model (as discussed in Section 6.2). In particular, RANK performs worse than our method, potentially because all weight vectors share the same subspace, which may not be desirable if not all the tasks are related with each other. DP-MTL performs worse than our method, potentially because it fits a *full-rank* Gaussian for each mixture component and

Table 6.1. Mean squared error (MSE) of various methods on multitask regression problems

	Synthetic	**School**	**Computer**
STL	1.35	468.7	153.3
MTFL	0.36	376.1	30.4
LWS	0.37	430.9	30.2
MFA-MTL	**0.18**	**374.5**	**29.8**

Table 6.2. Multitask classification accuracies of various methods on the **Landmine** and **20ng** datasets

	Landmine	**20ng**
STL	52.9%	69.3%
PRIOR	52.9%	75.8%
RANK	53.8%	75.8%
DP-MTL	53.8%	75.7%
MFA-MTL	**62.4**%	**76.9**%

is especially prone to overfit if the number of tasks is smaller than the number of features.

Finally, we investigated the behavior of different algorithms in the small training data regimes. For this, we varied the amount of training examples per task (for landmine data, we varied the fraction from 20% to 100%; for 20 Newsgroup, we varied the number of examples from 20 to 100). Results are shown in Figure 6.5. To uncrowd the figure, we compare only with STL and DP-MTL (the best performing baseline). In the small data regimes, our algorithm performs better as compared to both STL and DP-MTL. Another important aspect of an MTL algorithm is its asymptotic behavior in the limit of large training data per task. For this experiment, we compared MFA-MTL with STL on the school multitask regression dataset by providing each algorithm the complete training data. MFA-MTL resulted in an MSE of 261.4 as compared to STL, which gave an MSE of 271.1. Therefore, our algorithm tends to do comparably (in fact, marginally better) to independently learned tasks even when the amount of training data per task is sufficiently large.

6.4 Related Work

Apart from the prior work on Multitask Learning discussed in Section 8.1, our model is based on a somewhat similar motivation as the model proposed in (Argyriou et al., 2008). Their model assumes that tasks can be partitioned into groups and tasks within each group share a kernel. Their assumption is an extension of the earlier work on Multitask Feature Learning (Argyriou et al., 2007) (one of the baselines we used in our experiments) that assumes all tasks share the common kernel. In (Kumar and Daumé III, 2012), the authors assume that there is a *single* set of task *basis vectors* (i.e., a task dictionary) and each task is a *sparse* combination of these basis vectors. In their model, the number of basis vectors

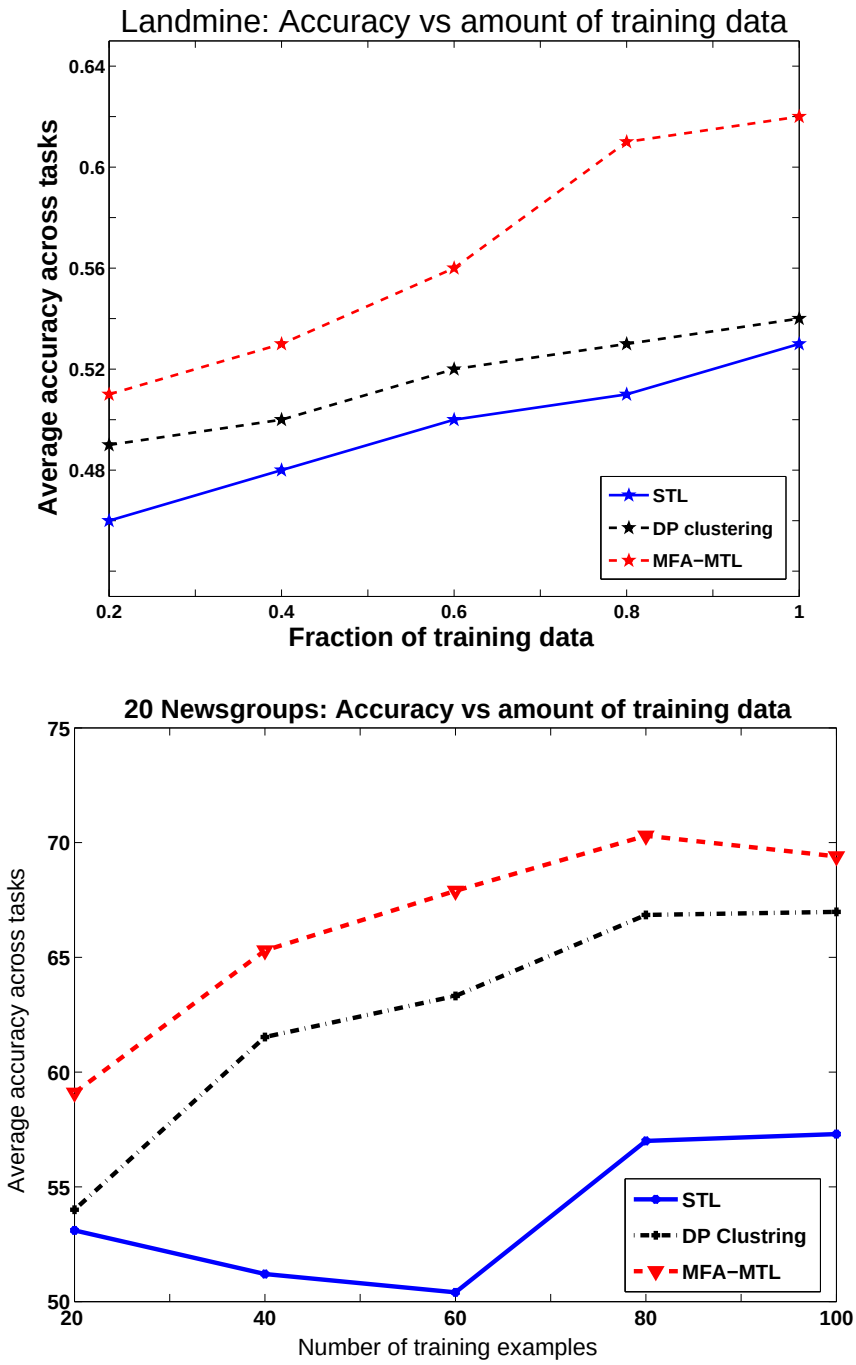


Figure 6.5. Average accuracies w.r.t. varying amount of training data (top: landmine data, bottom: 20ng data).

shared between two tasks can be seen as the pairwise task similarity. In (Kang et al., 2011), the authors proposed a model based on the assumption that the tasks exist in groups and the tasks within each group share features, which is again similar in spirit to our work (this model was one of our baselines in the experiments). In contrast, the generative model we presented in this chapter offers a number of advantages over these models, such as the ability to deal with missing data in a principled manner, doing automatic model complexity control in a nonparametric Bayesian setting, and being flexible enough to subsume these and many other notions as task relatedness used in Multitask Learning.

Among other related work, (Canini et al., 2010) propose Hierarchical Dirichlet Process models as good models for human categorical learning. The idea is that one can model transfer learning by assuming that people unsupervisedly learn subgroups of known classes and use these groups to refine the knowledge of new classes by sharing subgroups via a Hierarchical Dirichlet Process. Our model can be seen as a discriminative analog of their generative model, where aspects of the task parameter—instead of the distribution of the test examples—are shared among similar tasks and the sharing structure is discovered automatically.

6.5 Future Work and Discussion

We proposed and evaluated a nonparametric Bayesian Multitask Learning model that usefully interpolates between many different previously proposed models for estimating task parameters of multiple related learning problems, such as a shared Gaussian prior (Chelba and Acero, 2006), a clustering structure (Xue et al., 2007b), reduced dimensionality (Argyriou et al., 2007, Zhang et al., 2006), manifold structure (Agarwal et al., 2010, Ghosn and Bengio, 2003), etc. We presented a variational mean-field algorithm for this model that exhibits competitive results on a set of synthetic as well as real-world Multitask Learning datasets. The proposed model, by using the flexibility afforded by nonparametric Bayesian techniques, requires only minimal assumptions to be applied to any given Multitask Learning problem. A possible future work is studying a Hierarchical Dirichlet Process variant of this model where different tasks are allowed to share exactly the *same* θ parameters, which might be beneficial in cases where training data are especially sparse or the tasks are more strongly clustered.

CHAPTER 7

BEAM SEARCH-BASED MAP INFERENCE FOR THE INDIAN BUFFET PROCESS

This chapter describes our beam-search algorithm for the Indian Buffet Process.

7.1 Introduction

Although the Indian Buffet Process offers a flexible way to learn the correct number of latent features in the data, this flexibility comes at a price (as is true for most interesting/useful Bayesian models!). The combinatorially complex nature of the IBP (search over all possible binary feature assignment matrices) poses significant challenges during inference in the IBP-based models. MCMC-based approaches such as Gibbs sampling (Ghahramani et al., 2007) are traditionally used in these models, which tend to be computationally expensive and may take long to converge. Another alternative is to use variational methods (Doshi-Velez et al., 2009c). Although faster than the sampling-based methods, these can be difficult to design and implement, and can potentially run into local optima issues.

Sampling-based methods such as MCMC produce samples from the posterior distribution. However, in many applications, we only require the *maximum a posteriori* (MAP) sample, discarding all other samples. This naturally leads to the following question: *If all we care about is a single MAP assignment, why not find one directly?* Furthermore, note that although sampling and variational methods *aim* to explore the full posterior over the latent feature matrix, they may not be well-suited for searching a posterior mode: Sampling may take too long to mix and get close to the maxima; variational methods may not be able to find the true maxima due to their inherent local maxima problem. In this chapter, we propose search algorithms such as A^* and beam search (Russell and Norvig,

2003) for finding an *approximate* MAP estimate of the latent feature assignment matrix. Our approach can be a viable and more efficient alternative to sampling or variational approaches if only the MAP estimate is required. If samples from the true posterior are desired, then the search-based MAP estimate can serve as a sensible initializer for MCMC, resulting in faster convergence.

7.2 Infinite Latent Feature Model

Given an $N \times D$ matrix X of N observations having D dimensions each, the latent feature model represents X as $ZA + E$. Here, Z is an $N \times K$ binary matrix (with $K \ll D$) denoting which latent features are present in each observation, A is a $K \times D$ matrix consisting of feature scores, and E consists of observation specific noise. A crucial issue in these models is the choice of K, the number of latent features. The Indian Buffet Process (Section 2.3) (Ghahramani et al., 2007) defines a prior distribution on the binary matrix Z such that it can have a potentially unbounded (i.e., infinite) number of columns, and offers a principled way to select K automatically from the data.

The IBP defines the following probability distribution over the *left-ordered-form* of Z (invariant to latent feature ordering; see (Ghahramani et al., 2007) for details):

$$P([Z]) = \frac{\alpha^K}{\prod_{h=1}^{2^N-1} K_h!} e^{(-\alpha H_N)} \prod_{k=1}^{K} \frac{(N-m_k)!\,(m_k-1)!}{N!}$$

where H_N is the Nth harmonic number, K_h is the number of columns in Z with binary representation h, and $m_k = \sum_i Z_{ik}$. K is the number of nonzero columns in Z.

In this chapter, we consider models of the form $X = ZA + E$ (e.g., the linear-Gaussian model (Ghahramani et al., 2007)) where A can be integrated out and thus $P(X|Z) = \int P(X|Z,A)P(A)dA$ can be represented in closed form, or can be approximated efficiently. Here, we do not describe computing A but, given Z, it is easy to compute in these models.

7.3 Search-based MAP Estimate for IBP

Our beam-search algorithm (Figure 7.1) for IBP takes as input the set of observations, a scoring function g, and a maximum beam size b. The algorithm maintains a max-queue of candidate latent feature assignment matrices. Each of these matrices on the queue is

```
function IBPSearch
input: a scoring function g, beam size b, data X_{1:N}
output: IBP matrix Z
initialize max-queue: Q ← [⟨⟩]
while Q is not empty do
    remove the best scoring candidate Z from Q
    if |Z| = N then return Z
    for all possible assignments Z_{N^0} for the next (say N^0-th) customer (i.e., each
    of the 2^K possibilities from existing dishes, and for each possibility 0 and
    max{1, ⌈α/N^0⌉ − 1} new dishes) do
        let Z^0 = [Z; Z_{N^0}]
        compute the score s = g(Z^0, X)
        update queue: Q ← Enqueue(Q, Z^0, s)
    end for
    if b < ∞ and |Q| > b then
        Shrink queue: Q ← Q_{1:b}
            (drop lowest-scoring elements)
    end if
end while
```

Figure 7.1. The generic IBP search algorithm (takes the scoring function as input).

associated with a score on the basis of how likely it is to maximize the posterior probability of the *complete* Z given X. This essentially means how likely it is to being the eventual MAP estimate once we have seen all the observations. The maximum beam size specifies the maximum number of candidates allowed on the queue at any time. At each iteration, the highest scoring candidate Z is removed from the queue, and is expanded with the set of all possible feature assignments for the next (say N^0-th) observation. For the possible expansions, we consider 2^K possibilities for assigning the existing dishes and, for each such possibility, 0 and $\max\{1, \lceil \alpha/N^0 \rceil - 1\}$ new dishes (note: $\lceil \alpha/N^0 \rceil - 1$ is the *mode* of the number of new dishes chosen by the N^0-th customer in the IBP culinary analogy). Our algorithm therefore explores matrices Z of sizes *up to* $N \times \sum_{n=1}^{N} \max\{1, \lceil \alpha/N^0 \rceil - 1\}$, but this is a reasonable approximation since the number of latent features is typically much smaller than N or D. Scores are computed for each of the new candidates and these candidates are placed in the queue. If the beam size is not infinite, then we also drop the lowest scoring elements so as to maintain the maximum queue size. We stop at the point when the number of rows in the matrix removed from the queue equals the total number of

observations.

Scoring of the candidate latent feature assignment matrices constitutes an important aspect of our search algorithms. Recall that finding the MAP estimate requires finding Z that maximizes the posterior probability of Z given X, $P(Z|X)$, which is proportional to the joint probability $P(Z, X)$. However, since our algorithm processes one observation at a time (in an online fashion), at any point having seen N^0 observations, we can only have an upper bound on the joint probability of all N observations. Since the joint probability $P(Z, X)$ can be again factored as $P(Z)P(X|Z)$, an upper bound on $P(Z, X)$ can thus be obtained by independently upper-bounding the prior probability:

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

where $m_k = \sum_i Z_{ik}$, and the likelihood $P(X|Z)$, both given the first N^0 observations. In fact, as we shall show (Section 7.4), it is possible to even explicitly upper bound the prior term. Unfortunately, the same is not true for the likelihood term (as it also involves the future observations and their latent feature assignments), and we therefore propose several heuristics for upper bounding the likelihood term (Section 7.5). The sum (assuming probabilities are expressed on log scale) of these two terms is the scoring function.

The search algorithm is guaranteed to find the optimal MAP feature assignment matrix if the beam size is infinite and the scoring function g is *admissible*. Being admissible means that it should *over-estimate* the posterior probability of *best possible* feature assignment Z that agrees with Z^0 on the first N^0 observations. Denoting the condition as $Z|N^0 = Z^0$ as the *restriction* of Z to the first N^0 elements, admissibility can be written formally as:

$$g(Z^0, X) \geq \max_{Z:Z|N^0=Z^0} P(Z, X)$$

Although the admissible scoring functions provably lead to optimal MAP estimates, the NP-hardness of the MAP problem implies that these can be inefficient (in terms of enqueue/dequeue operations on the queue; a large gap between these two numbers would mean that it takes too long to search for the optimal candidate). For efficiency reasons, it is often useful to have scoring functions that occasionally *under-estimate* the true posterior probability, and are therefore *inadmissible*. In fact, as described in Section 7.5, our proposed scoring functions are not guaranteed to be admissible in general, but they lead

to efficient approximate MAP estimates for the Z matrix (see the Experiments section for evidence supporting this).

Our search algorithm is akin to the A^* search (Russell and Norvig, 2003) where we optimize a *path-cost-so-far* function plus a *cost-to-goal* function. In our case, we rank a candidate feature assignment matrix by computing its score that is a summation of the joint probability $P(X, Z)$ up to first N^0 observations (similar to the path-cost-so-far), and an *upper bound* on the joint probability corresponding to the remaining observations (similar to the cost-to-goal). Since the joint probability can be factored into the prior and the likelihood terms, we next show in Section 7.4 and Section 7.5 how each of these can be upper bounded. In keeping with the culinary metaphor of IBP, in the rest of the exposition, we will occasionally refer to observations as customers, and features as dishes.

7.4 Upper Bounding the Prior

Given the customer-dish assignment Z^0 for the first N^0 customers, it is possible to explicitly compute the dish assignment for the remaining customers that maximizes the probability $P(Z)$. For this maximization, we need to consider two cases for the remaining customers: (a) maximization w.r.t. the already selected dishes, and (b) maximization w.r.t. the new dishes.

7.4.1 Upper Bounding w.r.t. Already Selected Dishes

Given an $N^0 \times K$ matrix Z^0 for the first N^0 customers, if one were to maximize the IBP prior $P(Z)$, then the $(N^0 + 1)^{th}$ customer would choose an already selected dish k only if it was chosen previously by more than half the customers (i.e., the *majority*). Let us denote this event by a random variable $x_k = \mathbb{I}_{(m_k > N^0/2)}$, where $\mathbb{I}$ is the indicator function and m_k is the number of previous customers who chose the kth dish. Now, to maximize $P(Z)$, all subsequent customers would also make the same choice as the $(N^0 + 1)^{th}$ customer (since the customers making that choice will continue to remain in the majority). To derive the *probability* of this event happening, we appeal to the exchangeability of the IBP and can assume that the $(N^0 + 1)^{th}$ customer comes at the end after the remaining $(N - N^0 - 1)$ customers (who either all select or all skip the dish k). Therefore the probability that the $(N^0 + 1)^{th}$ customer *selects* dish k is $p_k = (m_k + (N - N^0 - 1))/N$, and the probability that this dish is skipped $1 - p_k$. Since all the $(N - N^0)$ customers make the identical choice

in selecting/skipping this dish, the random variable $x_k \in \{0, 1\}$ and p_k take on the same values for each customer. This leads to a score w.r.t. dish k:

$$s_k = [p_k^{x_k}(1 - p_k)^{(1-x_k)}]^{(N-N^0)}$$

which is a product of $(N - N^0)$ binomials. The total score for the maximization w.r.t. the existing dishes is given by the *product* (or the log sum if using log probabilities) of individual scores for each of the existing dishes.

7.4.2 Upper Bounding w.r.t. the New Dishes

In the IBP culinary metaphor, the nth customer selects $Poisson(\alpha/n)$ number of new dishes so the prior would be maximized if customer n selects a number of dishes equal to the *mode* of this number, which is $\lfloor \alpha/n \rfloor$. The score contribution of this part for $P(Z)$ is given by:

$$\prod_{n=N^0+1:N} \frac{(\alpha/n)^{\lfloor \alpha/n \rfloor !} \exp(-\alpha/n)}{\lfloor \alpha/n \rfloor !}$$

The part of the above product involving the $\exp$ terms just requires computing a harmonic mean of $(N - N^0)$ numbers. For the terms involving $\lfloor \alpha/n \rfloor$, we only need to care about those for which $\lfloor \alpha/n \rfloor > 0$. This computation is inexpensive since α is usually small and therefore $\lfloor \alpha/n \rfloor$ quickly goes to zero .

7.5 Upper Bounding the Likelihood

Unlike the prior term, an explicit maximization is not possible for the likelihood because the future observations would not have been assigned any latent features yet, precluding the associated likelihood computation. We propose here several heuristics for approximating the likelihood of future observations.

7.5.1 A Trivial Function

Given the matrix Z^0 having N^0 many rows, a possible trivial upper bound on $P(X|Z)$ can be obtained by only considering the likelihood over the first N^0 observations. This function is given by:

$$g_{Trivial}(X|Z^0) = P(X_{1:N^0}|Z^0)$$

For discrete likelihood distributions (e.g., multinomial likelihood), the *true* likelihood of each future observation is upper bounded by 1. Therefore, the above function would

be a trivial upper bound on $P(X|Z)$, since it assigns a probability one to the likelihood term of each future observation. With an infinite beam size, this admissible function is guaranteed to find the optimal MAP estimate. Note that this would however not be true for continuous likelihood distributions, e.g., Gaussian likelihood, which is actually a density (not a probability) upper bounded by $(2\pi\sigma_X^2)^{-1/2}$. Unless the data variance σ_X is such that $(2\pi\sigma_X^2)^{-1/2} \leq 1$, admissibility is not guaranteed in such cases, and the search would not be guaranteed to find the global optimal solution. Moreover, as discussed earlier in Section 7.3, even though the trivial function is admissible in certain cases and may find the optimal solution, the bound tends to be quite loose, which can make the search inefficient (see empirical evidence in the Experiments section).

7.5.2 An Inadmissible Function

Another possibility is to use a function which is significantly tighter (i.e., better approximation to the true likelihood), but not admissible in any of the cases. Therefore, the search is no longer guaranteed to find the global optimal solution. However, since it is tighter, it is much more efficient to run, and can find approximate solutions much more quickly. This *inadmissible* function is given by:

$$g_{Inad}(X|Z^0) = P(X|[Z^0; Z_{N^0+1:N}])$$

where $Z_{N^0+1:N}$ is a matrix of size $(N - N^0) \times (K + N - N^0)$ such that each future customer $n \in [N^0 + 1, \ldots, N]$ gets assigned a single (owned by himself) new dish. Here, $[Z^0; Z_{N^0+1:N}]$ denotes row-wise concatenation with appropriate padding of Z^0 and $Z_{N^0+1:N}$ with zeros. This is an inadmissible heuristic since it is always preferable to instead assign the same set of dishes to two customers if both are identical, a fact which this function does not take into account.

7.5.3 A Clustering-Based Function

Even though the trivial function discussed above is admissible in certain cases (i.e., discrete likelihood distributions), the upper bound is very loose since it does not take into account the feature assignments of any of the future observations, and the search would therefore be inefficient. The inadmissible function, on the other hand, assigns a single new dish to each future customer which may not mirror the likelihood of future observations

that closely. Our next proposal aims to find a middle ground by trying to account for the probable dish selection by the remaining customers.

One way to incorporate the dish assignment of future customers in the likelihood term is to first do a *coarse level* of feature assignment. Given the set of observations $X = [X_1, \ldots, X_N]$, we first run a clustering algorithm with a small number of clusters. Having obtained a clustered representation of the data, we pick one representative point from each cluster and run the IBP search algorithm (using the trivial scoring function described above) on these cluster representative observations. This gives us a *coarse* feature assignment for the representative points. We then run the IBP search on the full data and, while computing the likelihood (heuristic) of a future observation n, we use the same set of latent features for this observation as assigned to the representative data point of the cluster to which it belongs.

7.6 Experiments

We report experimental results on a variety of datasets (both synthetic and real), and compare the search-based approaches against a number of baselines. Our results are on two types of tasks: (1) latent Factor Analysis (Rai and Daumé III, 2008), and (2) factor regression (Rai and Daumé III, 2008, West, 2003), which uses the factors for making predictions in classification or regression settings (we experiment with classification setting). For the Factor Analysis task, we report the joint log probability scores and the time taken, and for the factor regression task, we report the predictive accuracies on a held-out test data.

7.6.1 Baselines and Experimental Setup

The baselines we compare against are uncollapsed Gibbs sampling (Ghahramani et al., 2007), infinite variational inference (Doshi-Velez et al., 2009c), and particle filtering (Wood and Griffiths, 2007) for the IBP. In addition, we also briefly discuss a comparison with a greedy search-based approach (Section 7.6.6). The variational inference was given 5 random restarts to avoid the issue of local optima (the reported time is the average time taken for a *single run*). The particle filter was run with a varying number of particles (500-5000) and the reported results are the best achieved with a minimum possible number of particles. We would like to note here that we also compared with the semicollapsed

Gibbs sampler for IBP (Doshi-Velez and Ghahramani, 2009), but the results and the running times were very similar to the uncollapsed Gibbs, so we included only the uncollapsed version in our experiments. The uncollapsed version has the same time complexity as the semicollapsed version (linear in the number of observations). Although the uncollapsed version is sometimes known to mix slowly, we did not observe this in our experiments. For our search-based approaches, we used small beam sizes (10-20), which seemed to be enough for our experiments. In our first experiment, we applied our search-based approach to the block-image dataset with known ground truth, generated in a manner akin to (Ghahramani et al., 2007) using a linear-Gaussian model of the data: $X = ZA + E$. The feature score matrix A has a zero mean Gaussian prior: $A \sim \mathcal{N}or(0, \sigma_A^2)$, and the noise as well is Gaussian: $E \sim \mathcal{N}or(0, \sigma_X^2)$. Our dataset consists of twenty 4×4 synthetic block-images generated by combining four different 4×4 latent images. The latent feature assignment matrix Z is 20×4. More importantly, we note that Z was *not* generated from an IBP prior. Each generated image had Gaussian noise with $\sigma_X = 0.1$ added to it. We then ran our search-based approaches and various baseline approaches on this dataset. The trivial, cluster-based, and the inadmissible approaches finish reasonably fast, taking a time of 1.02 seconds, 0.86 seconds, and 0.45 seconds, respectively, suggesting that the inadmissible search is the fastest among all (the number of enqueued/dequeued elements, though not reported to conserve space, were also the smallest for this method). In comparison, Gibbs sampling took 3.30 seconds, particle filter 0.98 seconds, and the infinite variational inference (Doshi-Velez et al., 2009c) took 3.73 seconds to finish (truncation level was set to 12). All approaches recovered the ground truth latent features.

7.6.2 E-Coli Data

The E-Coli dataset is a gene-expression dataset with known gene-pathway loadings, which is a sparse 50×8 binary matrix ($K = 8$) (Rai and Daumé III, 2008). This is a semireal dataset; the gene-factor connectivity network (binary Z matrix) is taken from a real dataset and the observations are simulated using this network using a linear-Gaussian model. We generated 50 observations with 100 dimensions each. The number of latent features, time taken, and log-joint probabilities reported by our search-based approaches and the other baselines are given in Table 7.1. As we see, our search-based approaches

Table 7.1. Results on the E-coli data

	K	Time (sec)	logP(X,Z)
Gibbs Sampling	6	49.8	**-4681**
Particle Filter	7	17.8	-5369
Infinite Variational	3	12.1	-6875
Trivial	**8**	72.5	-5887
Cluster-Based	**8**	15.5	-5759
Inadmissible	**8**	**10.3**	-5865

successfully recover the correct number of latent features (8) in the data, and are reasonably faster (with the inadmissible approach being the fastest) than the other baselines. The variational inference, although comparable to search in terms of speed, severely underestimates the number of latent features, possibly due to getting trapped in a local optima. In our experiment, we set the beam size to 10 in all the search-based approaches. The IBP parameter α was set to 3 and the hyperparameters (the noise variance σ_X and latent feature variance σ_A) were set based on the data variance, for all the algorithms, akin to the method in (Doshi-Velez and Ghahramani, 2009, Doshi-Velez et al., 2009c).

7.6.3 Scalability

Next, we demonstrate the scalability of the search-based algorithms with the number of observations. We report experiments on one synthetic and one real-world dataset. The synthetic dataset was generated using the IBP Prior with $\alpha = 1$ and linear Gaussian model of the data with noise variance $\sigma_X = 0.1$. The generated dataset consists of 1000 data points, each with 100 dimensions, and the number of latent features K is 4. We varied the number of observations from 200 to 1000 with increments of 200. For the real-world dataset, we take the 50 $\times$ 100 E-coli data and vary the number of observations from 10 to 50. The timings and log-joint probabilities for the synthetic and E-coli datasets are shown in Figure 7.2 and 7.3. As the figures show, the search-based approaches are the fastest on both the datasets (except for the trivial heuristic on E-Coli data). On the synthetic data, all the search approaches actually recover the ground truth (the log-joint probabilities of all search-based approaches therefore look the same). Also, although the timings are roughly the same for all search-based approaches, the inadmissible search did the fewest number of enqueue/dequeue operations, and was therefore the fastest. Among the other baselines,

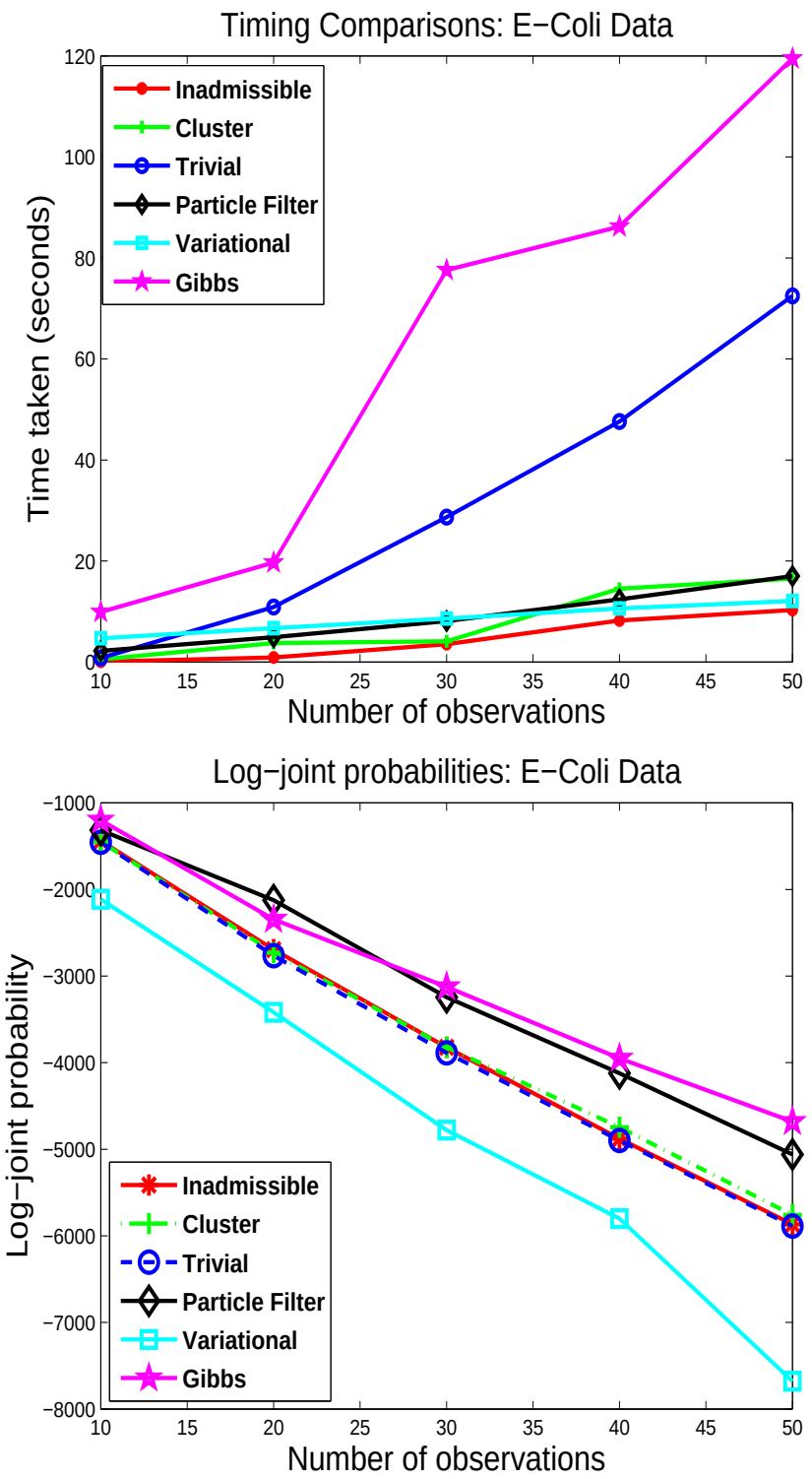


Figure 7.2. Scalability results of various algorithms for the E-Coli dataset

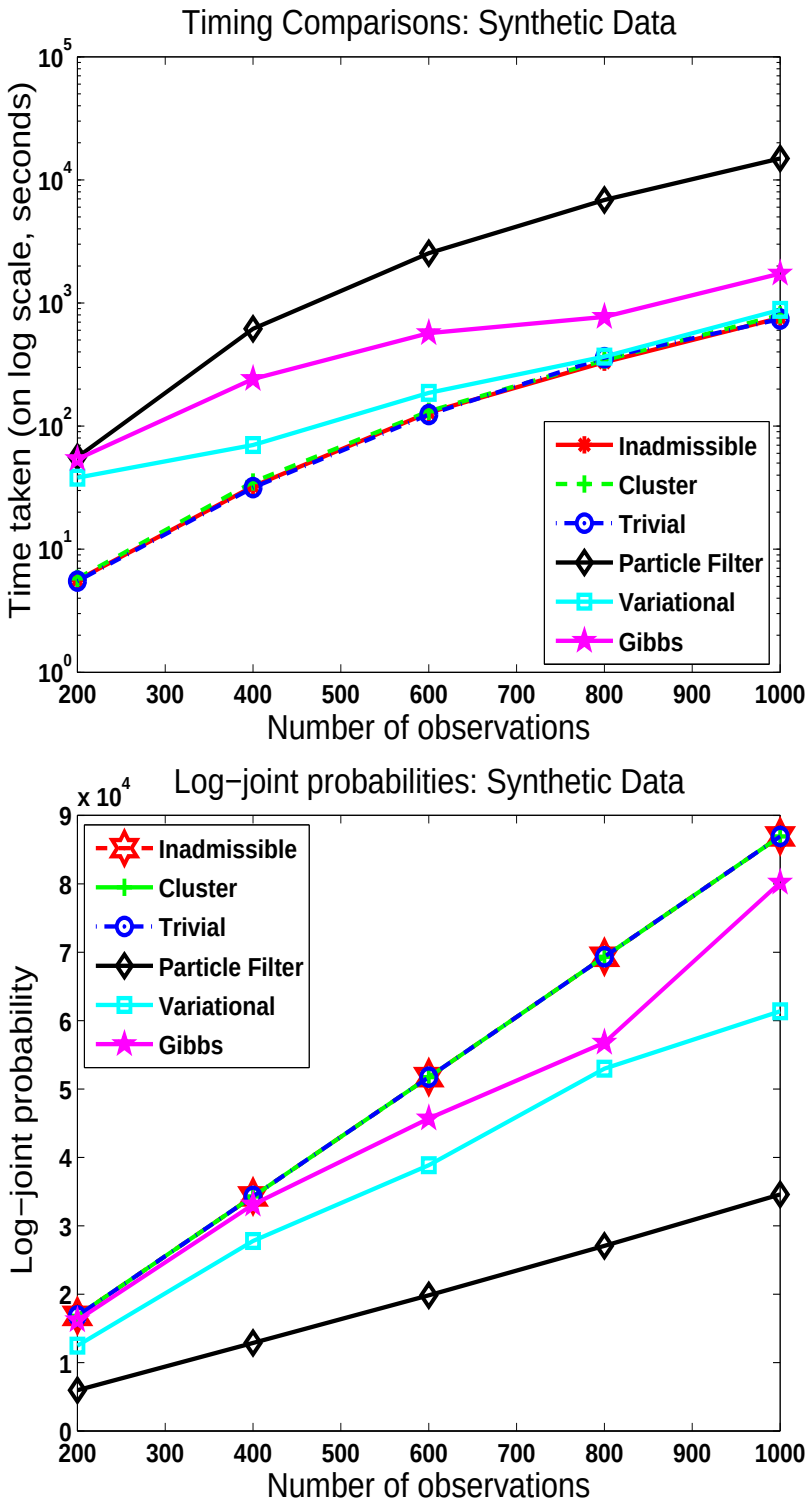


Figure 7.3. Scalability results of various algorithms for the Synthetic dataset

the variational inference is the fastest one but it fails to recover good solutions most of the time (as measured by the log-joint probability, and also the number of latent features discovered). The particle filter, although scaled well on small data regimes (E-Coli data), scaled poorly for large datasets, as can be seen by its (lack of) scalability on the synthetic data.

7.6.4 Factor Regression

Next, we apply the various methods on real-world binary classification datasets to extract latent factors and use them to train a classification model (akin to (Rai and Daumé III, 2008, West, 2003)). We use two real-world datasets for the classification tasks: the aspect-angle dependent sonar signals dataset and the scene classification dataset from the UCI Machine Learning Repository. The sonar signal dataset consists of 208 examples having 60 features each. The scene classification dataset is actually a multilabel dataset with 2407 examples having 294 features each; we chose the 7th label as a prediction task. Since the feature assignment matrix is binary and the latent factors we care about are real-valued, we applied all the algorithms on the transposed $D \times N$ data matrix. The matrix Z is $D \times K$ in this case, and we treat the $K \times N$ real-valued, feature score matrix A as the factor matrix (N examples with K real-valued features each) used to train the classification model. For the search-based algorithms, we compute A by drawing a sample from its posterior given Z.

After the feature extraction stage, we split the data into two equal parts (training and test), train an SVM classifier (with linear kernel), and then apply the learned classifier on the test data. We experiment with 200 random splits of training and test data and report the average and standard deviation of the accuracies achieved by various methods. As the results in Table 7.2 show, the search-based approaches achieve prediction performance that, in most cases, is competitive (or better) than Gibbs sampling. At the same time, search finished much faster than sampling in the latent Factor Analysis step of the task.

7.6.5 (Approximate) MAP as an Initializer

The search-based approach yields a MAP estimate. In many cases, however, we care about the full posterior. In such cases, the approximate MAP estimate found by our search-based algorithms can serve as a sensible initializer to the sampling-based approaches. As

Table 7.2. Latent factor-based classification results

	Sonar		**Scene**	
	Acc	K	Acc	K
Gibbs	70.9 ($\pm$4.8)	6	77.6 ($\pm$0.9)	6
Particle Filter	52.4 ($\pm$4.2)	6	77.8 ($\pm$1.3)	10
Infinite Variational	68.5 ($\pm$5.6)	10	74.3 ($\pm$2.1)	9
Trivial	72.4 ($\pm$3.9)	7	76.2 ($\pm$1.7)	7
Cluster Based	71.5 ($\pm$3.6)	7	77.8 ($\pm$2.1)	6
Inadmissible	67.1 ($\pm$4.9)	5	76.9 ($\pm$3.2)	6

an illustration, we ran an uncollapsed Gibbs sampler by using random initialization and the search-based MAP initialization, and monitored the joint likelihood over time. As we see in Figure 7.4, the MAP-initialized Gibbs sampler localizes itself in the high-probability region quite early on, as compared to the randomly initialized sampler, which takes much longer to attain similar values of the joint likelihood. The overhead of doing the search to get the MAP estimate is much smaller than the overall time taken by the Gibbs sampler.

7.6.6 Comparison with Greedy Search

We also compared our beam search-based approach with a greedy search heuristic, which works by selecting, for the $(N^0+1)^{th}$ observation, the feature assignment Z_{N^0+1} that maximizes the posterior probability up to this observation, i.e., $P([Z^0; Z_{N^0+1}]|X_{1:N^0+1})$. Note that this heuristic is similar to the one proposed in (Wang and Dunson, 2011) for the Dirichlet Process Mixture Model. Also, the greedy search approach is akin to beam search with the trivial heuristic, but without the explicit prior term maximization as we do in Section 7.4 (it only considers the prior $P([Z^0; Z_{N^0+1}])$ up to the $N^0 + 1$ observations) and a beam size of 1. Due to space limit, we do not report the full experimental results here, but we found that, on the block-images dataset, greedy search ran much slower than our inadmissible approach, ran almost as fast as the trivial heuristic, but inferred a much larger value of K than the ground truth (and lower log-likelihood scores). Moreover, the greedy search that only considers the posterior probability up to the current observation (ignoring the future observations) is not expected to do well if the number of observations is very large.

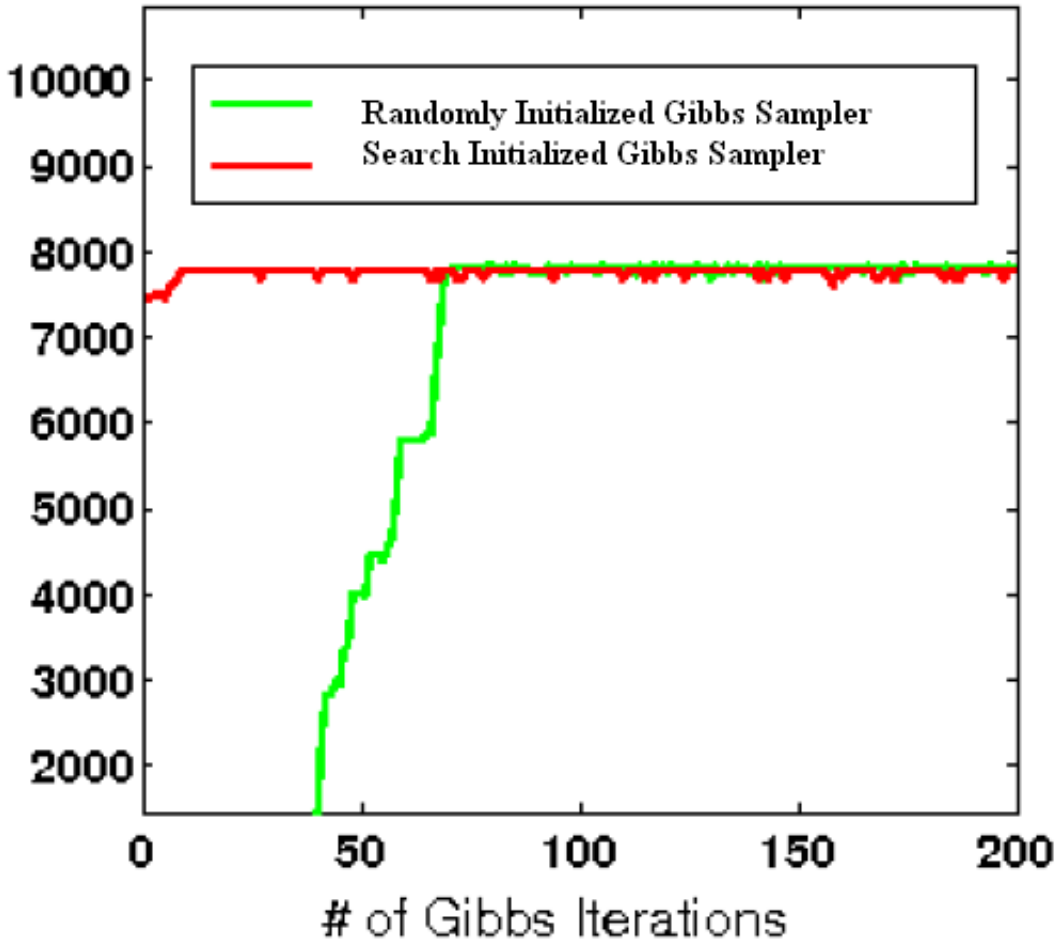


Figure 7.4. Log-likelihood scores for random vs search-based MAP initialized Gibbs Sampler

7.7 Related Work

In this section, we review previous work on inference in IBP-based models, some of which were used as baselines in our experiments. One of the first attempts to scale inference in IBP-based models to large datasets was the particle filter (Wood and Griffiths, 2007) for IBP. Particle filters are somewhat similar in spirit to our approach since a particle filter can be considered as doing a stochastic beam search. The particle filter can process one observation at a time. However, the particle filter samples each row of Z from the prior and the naïve sequential importance resampling scheme does not perform very well on datasets having a large number of observations (which is perhaps the reason behind the poor performance of particle filter in our experiments). Besides, particle filters are known

to suffer from the sample impoverishment problem and need to make multiple passes over the data to deal with this issue. Among the sampling-based approaches, (Doshi-Velez and Ghahramani, 2009) proposed a fast collapsed Gibbs sampler to address the slow mixing issue of the uncollapsed Gibbs sampler. Other sampling-based approaches include the Metropolis split-merge proposals (Meeds et al., 2006), and slice sampling (Teh et al., 2007b). Parallelization of the sampling-based inference for the IBP has also been attempted (Doshi-Velez et al., 2009a).

Deterministic variational inference can be an efficient alternative to sampling in IBP-based models. One such approach was proposed in (Doshi-Velez et al., 2009c), who proposed a variational inference algorithm for IBP which is based on the truncated stick-breaking approximation. Our search-based approach for inference is also deterministic and is similar in spirit to (Daumé III, 2007), who applied beam search algorithms for finding MAP estimates in Dirichlet Process mixture models. However, we note that the combinatorial problem posed by the IBP is even more challenging than the DP since the former looks at the space of $\mathcal{O}(2^{NK})$ possible feature assignments as opposed to the latter where this space is $\mathcal{O}(K^N)$ possible clusterings of the data.

7.8 Discussion and Conclusion

In this chapter, we have presented a general, search-based framework for MAP estimates in the nonparametric latent feature models. There are several aspects of the proposed algorithm that can be improved even further. Note that when a candidate is removed from the queue and expanded with the possible feature assignments for the next observation, we need to consider all 2^K possible candidates, compute their scores, and place them on the queue. This can be expensive for cases where K is expected to be large. An alternative to this would be to modify the proposed beam search by expanding along the *columns* of the Z matrix for a given row, considering one dish at a time (this would amount to a *search-within-search* procedure). Such a modification is expected to make search even faster. Besides, the heuristics used for likelihood maximization are critical to getting tighter bounds for the posterior and it would be interesting to consider other possible heuristics that result in even tighter even bounds. Another possibility is to estimate the hyperparameters (IBP hyperparameter α and the variance hyperparameters σ_X and σ_A, which are currently

set of a fixed value), for examples, as is done in (Wang and Dunson, 2011). Finally, although in the chapter we showed the conjugate case as an example (where we do not care about A), conjugacy is not necessary for our approach to be applicable. If the A matrix cannot be integrated out due to the nonconjugate prior, we can explicitly represent it at each step of the search algorithm by also computing the MAP assignment for A, given Z (for example, by running a few steps of some gradient-based optimizer), or by running a few Metropolis-Hastings steps for A, given Z.

CHAPTER 8

SPACE-EFFICIENT SEQUENTIAL INFERENCE FOR THE INDIAN BUFFET PROCESS

The previous chapter presented a search-based inference algorithm to obtain an approximate MAP solution for the latent feature assignment matrix in the Indian Buffet Process-based models. This chapter presents an online inference algorithm for the IBP, which is capable of processing one observation at a time. This is desirable both for scalability purposes as well as for the cases where the data naturally arrive in a sequential manner and batch methods such as Gibbs sampling and standard variational inference are no longer an option.

8.1 Introduction

Gibbs sampling (Doshi-Velez and Ghahramani, 2009, Ghahramani et al., 2007) and variational inference (Doshi-Velez et al., 2009c) are typically employed for doing inference in the Indian Buffet Process-based models (please refer to Section 8.5 for other related work). Both are, however, *batch* inference methods requiring all the observations at each step of the inference. This can make inference slow when dealing with datasets with large number of observations and/or high data dimensionality. Moreover, in an online setting where observations arrive one-at-a-time, batch methods are no longer an option. Besides, even in the batch setting, if new observations become available at a later point of time, inference needs to be re-run on the entire data. Sequential Monte Carlo (SMC) methods (Doucet et al., 2001) such as the particle filter offer an alternative by naturally allowing observations to be processed one-at-a-time. At the same time, the inherently sequential nature of inference also makes them amenable to be applied for large datasets. The SMC methods approximate the target posterior distribution using a discrete distribution defined

by a weighted set of "particles". Each particle is a sample from some (problem-specific) proposal distribution, and the associated weight denotes how much this particle is supported by the observations seen thus far.

For the IBP, in (Wood and Griffiths, 2007) the authors designed a sequential importance resampling (SIR) based particle filter and demonstrated better scalability than Gibbs sampling. However, as the number of observations grows, the particle filters are known to suffer from issues such as sample impoverishment, also known as the weight degeneracy problem (Doucet et al., 2001). This is the case when a small number of particles dominate the entire ensemble (i.e., their weights dominate the overall set of weights). Therefore, as the number of observations grow, the inference quality tends to deteriorate. In this chapter, we present a particle filtering method for the IBP designed to address these problems. We accomplish this by using an improved proposal distribution that takes into account the current observation, and additionally representing the particle filtering distribution as a mixture distribution with its mixture weights being exact (in the sense that we marginalize over the latent feature assignments of the current observation). Our method is in contrast with the particle filter for the IBP proposed in (Wood and Griffiths, 2007) in which the importance sampling proposal distribution ignores the current observation, and the importance weights depend on the "proposed" latent features of the current observation. These improvements lead to our method achieving better or comparable inference quality as compared to the standard particle filter for the IBP while requiring far fewer number of particles (giving the posterior a parsimonious representation (Snelson and Ghahramani, 2005)), and being comparable in terms of computational efficiency.

Just like the previous chapter, we consider the linear-Gaussian model (Griffiths and Ghahramani, 2011) for the data $\boldsymbol{X}$ with an IBP prior on the $\boldsymbol{Z}$ matrix. The model can be written as: $\boldsymbol{X} = \boldsymbol{Z}\boldsymbol{A} + \boldsymbol{E}$. Here, $\boldsymbol{A}$ is a $K \times D$ matrix consisting of latent feature scores, and $\boldsymbol{E}$ consists of observation-specific noise. In the linear-Gaussian model, the feature scores are Gaussian distributed with variance σ_a^2 and the noise is Gaussian with variance σ_x^2. Given these, the distribution of $\boldsymbol{X}$ is given by: $p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{A}) = \mathcal{N}or(\boldsymbol{X}|\boldsymbol{Z}\boldsymbol{A}, \sigma_x^2)$. For the rest of the exposition, we would be interested with cases where we want to infer only the latent feature matrix $\boldsymbol{Z}$, and not the $\boldsymbol{A}$ matrix. For the linear-Gaussian model, we would be using the collapsed likelihood $P(\boldsymbol{X}|\boldsymbol{Z}) = \int P(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{A})P(\boldsymbol{A})d\boldsymbol{A}$, which can

be represented in closed form by a Gaussian (Griffiths and Ghahramani, 2011) when $\boldsymbol{A}$ has a conjugate prior and can be integrated out. However, the particle filtering algorithms we describe in this chapter are applicable even in the nonconjugate settings where $\boldsymbol{A}$ cannot be integrated out. In such cases, we can explicitly also maintain a particle representation for $\boldsymbol{A}$ (note that if we want to additionally also infer $\boldsymbol{A}$ in the conjugate case, we can do the same).

8.2 Particle Filtering for IBP

We first introduce some notations. In what follows, small-case $\boldsymbol{x}_t$ denotes the tth observation and large-case $\boldsymbol{X}_t$ denotes the data matrix consisting of all the observations up to and including the tth observation. Likewise, small-case $\boldsymbol{z}_t$ denotes the latent feature assignment of the tth observation and large-case $\boldsymbol{Z}_t$ denotes the matrix consisting of the latent feature assignments of all the observations up to and including the tth observation.

Having processed the first t observations, in the next step of the particle filtering algorithm, the target posterior distribution for the latent feature assignment of up to the $(t+1)^{th}$ observations is expressed as:

$$p(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_{t+1}) \propto p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}, \boldsymbol{X}_t)p(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_t) \tag{8.1}$$

where

$$p(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_t) = \sum_{\boldsymbol{Z}_t} p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t)p(\boldsymbol{Z}_t|\boldsymbol{X}_t) \tag{8.2}$$

The particle filter approximates $p(\boldsymbol{Z}_t|\boldsymbol{X}_t)$ as a discrete distribution, which is defined by a particle representation based on a weighted set of N particles $\{w_t^{(i)}, \boldsymbol{Z}_t^{(i)}\}_{i=1}^N$ as follows: $p^N(\boldsymbol{Z}_t|\boldsymbol{X}_t) = \sum_{i=1}^N w_t^{(i)}\delta_{\boldsymbol{Z}_t^{(i)}}$. The particle approximation $\{w_t^{(i)}, \boldsymbol{Z}_t^{(i)}\}_{i=1}^N$ can be turned into an equally weighted random sample from $p(\boldsymbol{Z}_t|\boldsymbol{X}_t)$ by sampling with replacement from the discrete distribution $\{w_t^{(i)}, \boldsymbol{Z}_t^{(i)}\}_{i=1}^N$. This produces a new sample with uniform weights $w_t^{(i)} = 1/N$. Using this uniformly distributed sample, we can approximate the combinatorial summation over $\boldsymbol{Z}_t$ in Equation 8.2 by a more tractable summation: $p(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_t) \approx \frac{1}{N}\sum_{i=1}^N p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t^{(i)})$, which in turn can be used to approximate Equation 8.1:

$$p^N(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_{t+1}) \propto \frac{1}{N}\sum_{i=1}^{N} p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}, \boldsymbol{X}_t)p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t^{(i)}) \tag{8.3}$$

The above equation shows how the particle approximation of $p^N(\boldsymbol{Z}_t|\boldsymbol{X}_t)$ can be updated to a particle approximation to $p^N(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_{t+1})$. Note that Equation 8.3 expresses the target posterior in form of a *mixture distribution*. The mixture components are given by the distributions $\{p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t^{(i)}\}_{i=1}^N$ and the weights are given by the corresponding likelihood term $p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}, \boldsymbol{X}_t)$.

The following algorithm produces the particle approximation of the target distribution $p(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_{t+1})$ given samples from $p^N(\boldsymbol{Z}_t|\boldsymbol{X}_t)$:

1. Draw $\boldsymbol{Z}_{t+1}^{(i)} \sim p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t^{(i)})$ for $i = 1, \ldots, N$

2. Compute particle weights (and normalize)

$$w_t^{(i)} \propto p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}^{(i)}, \boldsymbol{X}_t) \tag{8.4}$$

3. Resample $\boldsymbol{Z}_{t+1}^{(i)} \sim \mathcal{M}ult(\{w_t^{(i)}\}_{i=1}^N)$ for $i = 1, \ldots, N$

This summarizes the particle filtering algorithm for the IBP proposed in (Wood and Griffiths, 2007). This is basically a sequential importance resampling (SIR) algorithm where the proposal distribution used in step-1 (in the context of the IBP) is given by the transition prior for $\boldsymbol{Z}_{t+1}$ given the latent feature assignments of the previously seen observations, and each sample $\boldsymbol{Z}_{t+1}^{(i)}$ is weighted by the the *conditional* probability $p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}^{(i)}, \boldsymbol{X}_t)$ of the most recent observation $\boldsymbol{x}_{t+1}$ given all the previous observations $\boldsymbol{X}_t$ and the latent feature assignment matrix $\boldsymbol{Z}_{t+1}$.

8.3 Improved Particle Filtering for IBP

Although the SIR-based particle filtering approach for the IBP described in Section 8.2 offers a nice way to sequentially update the target posterior distribution as new observations arrive, it has some inherent limitations. The method uses the transition prior $p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t)$ as the proposal distribution, and therefore ignores the current observation $\boldsymbol{x}_{t+1}$. This is problematic because the drawn sample $\boldsymbol{Z}_{t+1}^{(i)}$ may not lie in the important, high-likelihood region. Although SIR weights each particle, the weight computation involves the likelihood conditioned on the "proposal" $\boldsymbol{Z}_{t+1}^{(i)}$.

To circumvent these issues, we present an improved particle filtering algorithm for doing inference in the IBP-based models. Our algorithm makes use of a proposal distribution that takes into account the current observation, and can compute the mixture weights *without* having them depend on the proposal $\boldsymbol{Z}_{t+1}^{(i)}$ (by marginalizing out $\boldsymbol{Z}_{t+1}^{(i)}$). We note that similar ideas have been proposed recently for doing particle filtering in models such as mixture regression models, conditional dynamic linear models, and nonparametric mixture models (Lopes et al., 2011).

We now describe the idea more formally. The idea is based on expressing the target distribution as:

$$p(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_{t+1}) \propto \int p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_t, \boldsymbol{X}_t)p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t, \boldsymbol{X}_{t+1})p(\boldsymbol{Z}_t|\boldsymbol{X}_t)d\boldsymbol{Z}_t \tag{8.5}$$

This representation of $p(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_{t+1})$ is different from Equation 8.1. In Equation 8.5, $p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_t, \boldsymbol{X}_t)$ denotes the predictive likelihood and $p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t, \boldsymbol{X}_{t+1})$ is the updated state posterior. Using this alternate representation of $p(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_{t+1})$, we obtain the following mixture representation for its particle approximation $p^N(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_{t+1})$ given samples from the particle approximation of $p^N(\boldsymbol{Z}_t|\boldsymbol{X}_t)$:

$$\begin{aligned} p^N(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_{t+1}) &\propto \sum_{i=1}^{N} p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_t^{(i)}, \boldsymbol{X}_t)p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t^{(i)}, \boldsymbol{X}_{t+1}) \\ &= \sum_{i=1}^{N} w_t^{(i)} p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t^{(i)}, \boldsymbol{X}_{t+1}) \end{aligned} \tag{8.6}$$

The particle weights $w_t^{(i)}$ are given by:

$$w_t^{(i)} = \frac{p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_t^{(i)}, \boldsymbol{X}_t)}{\sum_{i=1}^{N} p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_t^{(i)}, \boldsymbol{X}_t)} \tag{8.7}$$

Note that, unlike the SIR-based particle filtering for the IBP (Wood and Griffiths, 2007), the weight computation marginalizes out the "proposed" latent features $\boldsymbol{z}_{t+1}$ of the current observation $\boldsymbol{x}_{t+1}$ (cf, Equation 8.4).

Given the mixture representation of the posterior as in Equation 8.6, here are the sampling equations:

1. Resample $\boldsymbol{Z}_t^{(i)} \sim \mathcal{M}ult(\{w_t^{(i)}\}_{i=1}^N)$ for $i = 1, \ldots, N$

2. Draw $\boldsymbol{Z}_{t+1}^{(i)} \sim p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t^{(i)}, \boldsymbol{X}_{t+1})$ for $i = 1, \ldots, N$

To actually apply this algorithm in practice, we need to be able to compute the weights $w_t^{(i)} \propto p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_t^{(i)}, \boldsymbol{X}_t)$ which requires evaluating the predictive likelihood given by

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_t, \boldsymbol{X}_t) = \sum_{\boldsymbol{Z}_{t+1}} p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}, \boldsymbol{X}_t) p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t) \tag{8.8}$$

and sampling the latent feature assignments $\boldsymbol{Z}_{t+1}$ from the proposal distribution (step 2) given by

$$p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t, \boldsymbol{X}_{t+1}) \propto p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}, \boldsymbol{X}_t) p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t) \tag{8.9}$$

8.3.1 Computing the Mixture Weights

To evaluate the expression in Equation 8.8, we can perform an explicit summation over all possibilities of the latent feature assignments of $\boldsymbol{X}_{t+1}$. This can, however, be expensive due to the combinatorially many possibilities of $\boldsymbol{Z}_{t+1}$ (both for existing and newly proposed dishes). To avoid that, we use Monte-Carlo sampling to generate a set of S samples $\{\boldsymbol{Z}_{t+1}^s\}_{s=1}^S$ from the distribution $p(\boldsymbol{Z}_{t+1}|\boldsymbol{Z}_t)$, which is easy to sample from for the IBP (following the culinary analogy described in Section 2.3). Given these samples, we can approximate the integral in Equation 8.8 by an empirical average:

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_t, \boldsymbol{X}_t) \approx \frac{1}{S} \sum_{s=1}^{S} p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}^s, \boldsymbol{X}_t) \tag{8.10}$$

and use these empirical averages in Equation 8.7 for computing the mixture weights.

8.3.2 Sampling from the Proposal

To sample from the proposal distribution given in Equation 8.9, we first select the highest probablity sample from $\{\boldsymbol{Z}_{t+1}^s\}_{s=1}^S$ given by:

$$\hat{\boldsymbol{Z}}_{t+1}^{(i)} \quad = \quad \arg\max_{\boldsymbol{Z}_{t+1}^s} p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}^s, \boldsymbol{X}_t) p(\boldsymbol{Z}_{t+1}^s|\boldsymbol{Z}_t)$$

and then run a Gibbs sampling step initialized with that sample.

Note that $p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}^s, \boldsymbol{X}_t)$ need not be computed again since it was already computed while computing the mixture weights. Also, given the Monte-Carlo samples $\boldsymbol{Z}_{t+1}^s$, computing the probability $p(\boldsymbol{Z}_{t+1}^s|\boldsymbol{Z}_t)$ is simple for the IBP prior - it is just a product of probabilities of each cell of $\boldsymbol{z}_{t+1}$. For a cell corresponding to an existing dish k, the

Algorithm 1 IBP-PF-CP

Input: Data $\boldsymbol{X}_{1:T}$, α, σ_x, σ_a, N : number of particles, S : number of Monte-Carlo samples to be used for computing the particle weights and sampling from the proposal

Output: Particle representation of the IBP matrix $\{\boldsymbol{Z}^{(i)}\}$ for $i = 1, \ldots, N$

```
for t = 0 to T − 1 do
    if t = 0 then
        for i = 1 to N do
            Try k = 0 to k_new dishes for z_1 (k_new = α, or some fixed number)
            Compute p(z_1|x_1) ∝ p(x_1|z_1) × Poisson(k; α) for each possibility of z_1
            Set z_1^(i) to z_1 that maximizes p(z_1|x_1)
            Z_1^(i) ← z_1^(i)
        end for
    else
        Compute weight w_t^(i) for i = 1, ..., N (using Equation 8.7, Equation 8.10, and
        Equation 8.11)
        Resample Z_t^(i) ∼ Mult({w_t^(i)}_{i=1}^N) for i = 1, ..., N
        Draw Z_{t+1}^(i) ∼ p(Z_{t+1}|Z_t^(i), X_{t+1}) for i = 1, ..., N (as described in Sec-
        tion 8.3.2)
    end if
end for
```

probability is computed using the Bernoulli distribution with parameter $m_k/(t+1)$ where m_k is the sum of the k-th column of the matrix $\boldsymbol{Z}_t$. For the cells corresponding to the newly sampled dishes, probabilities are evaluated using the Poisson distribution with parameter $\alpha/(t+1)$..

8.3.3 Computing the Conditional Probabilities

Note that both computing the mixture weights using Equation 8.10 and sampling from the proposal involve computing the *conditional* probability $p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}^s_{t+1}, \boldsymbol{X}_t)$ of the most recent observation $\boldsymbol{x}_{t+1}$ given all the previous observations $\boldsymbol{X}_t$ and the latent feature assignment matrix $\boldsymbol{Z}^s_{t+1}$. For the linear-Gaussian observation model with an IBP prior on the latent feature matrix $\boldsymbol{Z}$, $p(\boldsymbol{X}_{t+1}|\boldsymbol{Z}_{t+1})$ is Gaussian. Therefore, using the conditioning rule for Gaussians, the conditional probability $p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}, \boldsymbol{X}_t)$ will be a Gaussian as well. In the linear-Gaussian model, the distribution $p(\boldsymbol{X}_{t+1}|\boldsymbol{Z}_{t+1})$ has its covariance matrix $\boldsymbol{\Sigma}^{-1}$ given by: $\boldsymbol{\Sigma}^{-1} = \boldsymbol{I} - \boldsymbol{Z}_{t+1}(\boldsymbol{Z}^\top_{t+1}\boldsymbol{Z}_{t+1} + \frac{\sigma_x^2}{\sigma_a^2}\boldsymbol{I})^{-1}\boldsymbol{Z}^\top_{t+1}$, where σ_x is the noise variance and σ_a is the feature score variance. The covariance matrix $\boldsymbol{\Sigma}^{-1}$ can be partitioned as: $\boldsymbol{\Sigma}^{-1} = \left[\begin{array}{c|c} \boldsymbol{C}_1 & \boldsymbol{c}_2 \\ \hline \boldsymbol{c}_2^T & c_3 \end{array}\right]$ where $\boldsymbol{C}_1$ is a matrix, $\boldsymbol{c}_2$ is a vector, and c_3 is a scalar. With this

decomposition structure of $\boldsymbol{\Sigma}^{-1}$, the *conditional* distribution is given by:

$$\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}, \boldsymbol{X}_t \sim \mathcal{N}or(\boldsymbol{c}_2^\top \boldsymbol{C}_1^{-1}\boldsymbol{X}_t, c_3 - \boldsymbol{c}_2^\top \boldsymbol{C}_1^{-1}\boldsymbol{c}_2) \tag{8.11}$$

Evaluation of this probability can be made more efficient by exploiting the structure of $\boldsymbol{C}_1$, which can make the matrix inversion faster (Barnett, 1979).

8.3.4 The Full Algorithm

The complete algorithm for the linear-Gaussian model is given in Algorithm 1. We call our algorithm IBP-PF-CP (for IBP Particle Filtering with Compact Posterior). The algorithm processes one observation at a time. Note that the weight calculation for the very first observation is not required. For this observation, we enumerate the number of latent features to assign (up to a fixed number), and for each possibility, compute the posterior $p(\boldsymbol{z}_1|\boldsymbol{x}_1)$. The vector $\boldsymbol{z}_1$ corresponding to the largest value of the posterior is chosen as the assignment for the first observation. For each subsequent observation $\boldsymbol{x}_{t+1}$, we follow the 3 steps of weight computation, resampling particles using these weights, and finally drawing the latent feature assignment from the proposal distribution.

8.4 Experiments

We provide experimental results on both synthetic and real datasets. In our experiments, we first compare our method IBP-PF-CP with the particle filtering method proposed in (Wood and Griffiths, 2007) (referred to as IBP-PF) on all the datasets. Then, in Section 8.4.5, we also compare our method with batch inference methods for the IBP based on standard Gibbs sampling (Griffiths and Ghahramani, 2011) and infinite variational inference method proposed in (Doshi-Velez et al., 2009c) on all the datasets.

For the synthetic datasets with ground truth $\boldsymbol{Z}$ known, we use the difference between the true $\boldsymbol{Z}\boldsymbol{Z}^\top$ and the inferred $\mathbb{E}[\boldsymbol{Z}\boldsymbol{Z}^\top]$ (i.e., averaged over all particles, or samples) to measure the quality of inference. Note that $\boldsymbol{Z}\boldsymbol{Z}^\top$ represents the pair-wise similarities between the observations in terms of the latent features they possess. This error metric (referred to as ERROR) is computed following (Wood and Griffiths, 2007) by taking the expectation of the matrix $\boldsymbol{Z}\boldsymbol{Z}^\top$ over the posterior samples/particles produced by each method, followed by computing the summed *absolute* difference between the upper triangular portion of $\mathbb{E}[\boldsymbol{Z}\boldsymbol{Z}^\top]$ computed over the samples/particles and the upper triangular portion of the *true*

$\boldsymbol{Z}\boldsymbol{Z}^\top$ (including the diagonal). On other datasets, where the ground truth $\boldsymbol{Z}$ is not known, we report the log-joint probabilities achieved by each method.

8.4.1 Synthetic Data

The first dataset is a synthetic dataset generated using the linear-Gaussian model (Griffiths and Ghahramani, 2011). The dataset consists of 150 observations, each of dimensionality 150. The latent feature matrix $\boldsymbol{Z}$ was generated using the IBP prior with $\alpha = 2$, which resulted in $\boldsymbol{Z}$ being 150×5. Using this $\boldsymbol{Z}$, noise variance $\sigma_x = 0.1$, and feature score variance $\sigma_a = 1$, we generated the 150×150 data matrix $\boldsymbol{X}$. We then ran both particle filter methods IBP-PF-CP and IBP-PF on this data by varying the number of particles from 50 to 250 with increments of 50. The number of Monte-Carlo samples in our method is set to 10 in all cases. For both methods, we average the results over 10 different initializations.

On synthetic data, as Figure 8.1 (top) shows, our method achieves considerably lower error as compared to the standard particle filter for the IBP. Moreover, even with very small number of particles, our method results in very small error (and with the number of particles set to 200 or 250, the error goes to zero - so we recover the ground truth *exactly*). It shows that the particle representation of our method is more parsimonious as compared to the standard particle filter. For IBP-PF, although the error goes down with increasing number of particles, it always stays higher than IBP-PF-CP. Another remarkable thing is the stability of IBP-PF-CP as measured by the standard deviation of the error across the multiple runs. In contrast, the standard deviations of the IBP-PF are much larger.

8.4.2 Block-Images Data

The second dataset is the block-images dataset also used in (Wood and Griffiths, 2007). This dataset consists of a set of 100 images with each consisting of a subset of four shared latent images of size 6×6. A 100×4 binary matrix $\boldsymbol{Z}$ is used to generate the 100 images from these four latent images, using a noise variance $\sigma_x = 0.1$. On this dataset, we compare both IBP-PF-EXACT and IBP-PF by varying the number of particles from 50 to 500 with increments of 50. As Figure 8.1 (middle) shows, even with as few as 50 or 100 particles, the mean accuracy of IBP-PF-CP is close to the mean accuracy of IBP-PF with 500 particles. This shows that the particle-based posterior representation learned by our method is more accurate and at the same time more succinct. Moreover, as was the case with the synthetic

data experiment in Section 8.4.1, the accuracy of our method is fairly stable across multiple runs as is evident by the extremely small standard deviations. In contrast, for the standard particle filter, although the mean accuracies improve as the number of particles increase, the standard deviations still remain quite high.

8.4.3 Breast-Cancer Data

The third dataset is a breast-cancer dataset consisting of the 226 gene-expression values from 251 samples (Knowles and Ghahramani, 2011, Rai and Daumé III, 2008). For this dataset, the ground truth is not known and, therefore, we compare the log-joint probabilities $P(\boldsymbol{X}, \boldsymbol{Z})$ of IBP-PF-CP and IBP-PF. As Figure 8.1 (bottom) shows, our method achieves better log-joint probabilities as compared to the standard particle filter for the IBP, as was the case with the previous two datasets.

8.4.4 Computation vs Storage Trade-off

We would like to mention here that although our method would require more computation per particle as compared to the standard particle filter, the individual particles in our method are much better representatives of the target posterior (because of the improved proposal distribution and improved particle weights). Therefore, our method needs far fewer particles as compared to the standard particle filter to achieve better (or comparable) inference quality, as our experiments suggest. Parsimonius representations of the posterior distribution (Snelson and Ghahramani, 2005) are appealing since they require small storage cost and can be faster when evaluating predictive quantities or doing averaging over samples.

8.4.5 Comparison with Batch Methods

Finally, we compare our sequential inference method with Gibbs sampling (Griffiths and Ghahramani, 2011) and and infinite variational inference (Doshi-Velez et al., 2009c) for the IBP. Note that these are batch methods and make use of all the data at each step of the inference. The Gibbs sampler was run until there was no improvement in the log-joint probabilities. The variational inference was given 5 random restarts to avoid the issue of local optima (the reported time is the average time taken for a *single run*). We also averaged the results over 10 such runs of the variational inference method. The truncation level for

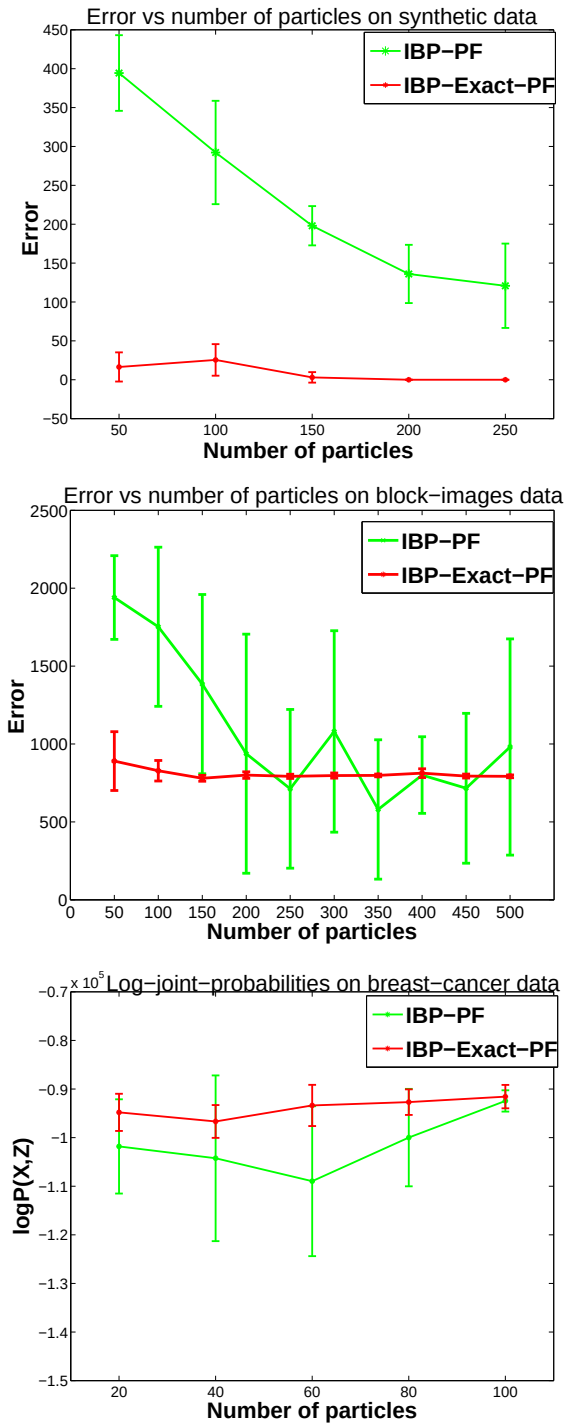


Figure 8.1. Inference quality vs number of particles. (Top) Error vs number of particles on synthetic data. (Middle) Error vs number of particles on block-images data. (Bottom) Log-joint-probability vs number of particles on breast-cancer data. Results for each sampler are averaged over 10 runs with random initializations.

variational inference was set at twice the number of latent features in cases where this number is known. For the breast-cancer data, we set it to 30. Our method was run using 50 particles (with 10 Monte-Carlo samples) and results are averaged across 10 runs.

The results on the block-images dataset and the breast-cancer dataset are shown in Table 8.1 . As the results show, our method runs much faster than the uncollapsed Gibbs sampler while achieving comparable inference quality. Our method also achieves better inference quality than variational inference.

Finally, we would like to mention that since the batch methods have access to all the data at each step of the inference, the better inference quality of Gibbs sampling as compared to our method is to be expected.

8.5 Related Work

In this section, we review prior work on inference in the IBP-based models. Since MCMC methods are widely used, a lot of effort has gone into improving the standard Gibbs sampling used for the IBP (Griffiths and Ghahramani, 2011). Among the sampling-based approaches, (Doshi-Velez and Ghahramani, 2009) proposed a fast collapsed Gibbs sampler to address the slow mixing issue of the uncollapsed Gibbs sampler. Other sampling-based approaches include the Metropolis split-merge proposals (Meeds et al., 2006), slice sampling (Teh et al., 2007b), and sampling based on the stick-breaking representation of the Beta process (Paisley et al., 2010). Parallelization of the sampling-based inference for the IBP has also been attempted (Doshi-Velez et al., 2009a).

Deterministic variational inference can be an efficient alternative to sampling in IBP-based models. One such approach was proposed in (Doshi-Velez et al., 2009c), who proposed a variational inference algorithm for IBP, which is based on the truncated stick-breaking approximation. In subsequent work (Paisley et al., 2011a), a variational inference algorithm was proposed in using the stick-breaking construction of the Beta Process. Expectation Propagation (Minka, 2001) combined with variational inference was used in (D. et al., 2010) for IBP-based nonnegative matrix factorization. Among other deterministic inference methods for the IBP, beam-search was proposed in (Rai and Daumé III, 2011) for the special case when only a *maximum-a-posteriori* (MAP) estimate of the latent feature assignment is needed.

Table 8.1. Comparison with batch methods (first and second column: block-images data; third and fourth column: breast-cancer data)

	Error	Avg. time	$\log P(X, Z)$	Avg. time
Uncoll. Gibbs	0($\pm$ 0)	124	-6.12×10^4	2236
Infinite Variational	2542($\pm$ 246)	96	-7.32×10^4	384
IBP-PF-CP	814($\pm$ 54))	92	-6.46×10^4	462

In the context of nonparametric Bayesian methods, SMC inference has been applied in the past for doing inference in Dirichlet Process mixture models (Fearnhead, 2004, Lopes et al., 2011, MacEachern et al., 1999, Ulker et al., 2010), and has shown to achieve better scalability than batch inference methods such as Gibbs sampling. For the Indian Buffet Process, the only known particle filtering algorithm is by (Wood and Griffiths, 2007), which we have compared against in this chapter.

8.6 Future Work and Extensions

There are several directions along which our proposed method can be improved. Note that although our proposal distribution is exact by construction, computing the weights requires evaluating the predictive likelihood $p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_t)$ of the next observation $\boldsymbol{x}_{t+1}$ given the latent feature assignments of all the observations up to the previous step. This required a combinatorial summation over the possible latent feature assignments of $\boldsymbol{x}_{t+1}$. To circumvent this issue, we used Monte-Carlo simulation (Section 8.3.1) and it tends to work well in practice. Coming up with better (and more efficient) ways of doing this remains an open question. Moreover, computing the weights also involves computing the conditional probabilities given by the collapsed likelihood expression $p(\boldsymbol{x}_{t+1}|\boldsymbol{Z}_{t+1}, \boldsymbol{X})$, for which the cost of evaluation grows with the number of observations. If we additionally maintain a particle representation of the feature score matrix $\boldsymbol{A}$, then we can use the uncollapsed likelihood $p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{A})$, which will be much more efficient. Moreover, using tricks such as rank-1 updates for Gibbs sampling in the IBP-based models (Doshi-Velez and Ghahramani, 2009) could potentially lead to further speed-ups.

Another possible extension would be to also sample the hyperparameters α, σ_x, and σ_a. This can be accomplished by following the similar framework as used in (Lopes et al., 2011) by also maintaining a particle representation of the hyperparameters.

8.7 Discussion and Conclusion

In this chapter, we have presented a sequential Monte Carlo (SMC) method for inference in the infinite latent feature models based on the Indian Buffet Process. Our method improves upon the previously proposed particle filter for the IBP by making use of a better, mixture representation-based proposal distribution, which can be sampled from exactly, and does away with importance sampling-based methods traditionally used in particle filtering. Our results demonstrate that our method significantly improves the quality of inference over the standard particle filter while still being computationally efficient. In particular, our results showed that, even with a very small number of particles, the method can learn reasonably well approximations of the target posterior distribution. In contrast, the standard particle filter requires considerably higher number of particles to achieve the similar inference quality. This was evident from the final inference accuracies, and also from the variance of the particles at each step of our inference method.

We believe that the potential of SMC methods for doing inference in nonparametric Bayesian models has remained largely unexplored. One of the main reasons for this has been the problems that plague these methods, especially with large data sizes and high data dimensionality, which leads to issues such as poor representation of the target posterior (e.g., due to the sample impoverishment problem). However, as we have shown in this chapter, with carefully constructed SMC samplers, such problems can be alleviated and SMC methods can be successfully applied in real-world settings requiring online inference for nonparametric Bayesian models. At the same time, the computational efficiency of these methods also makes them viable alternative to batch inference methods such as MCMC and variational inference.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

The primary contributions of this thesis lie in designing flexible models for discovering latent structures from data. The types of latent structures considered in this thesis include latent features underlying high-dimensional data, latent relationships (i.e., dependency structures) among the latent features, and latent *task structures* among a set of related learning tasks. The thesis accomplishes these by leveraging the flexibility of nonparametric Bayesian models, and by designing efficient approximate inference methods for such models (in particular, the nonparametric latent feature model). To summarize, the contributions of the thesis include:

- Designing nonparametric Bayesian latent feature models for high-dimensional data, while allowing the latent features to be have relatioships that we *simultaneously* want to infer.

- Designing nonparametric Bayesian models for learning *shared predictive structures* to better solve multiple related prediction tasks jointly (the problem of Multitask Learning).

- Designing efficient approximate inference algorithm for nonparametric Bayesian models, particularly for the nonparametric latent feature model - the Indian Buffet Process.

9.1 Future Directions

The work in this thesis can be extended along several directions. Some of the possible future works include:

- **New methods for latent feature modeling:** In the model we proposed in Chapter 3, we used a combination of the IBP and the Kingman's Coalescent to introduce interdependencies along the latent features. It would be interesting to design ways of accomplishing this in a more direct manner. Some recent works have explored this direction (Doshi and Ghahramani, 2009b, Paisley et al., 2011b, Zhang et al., 2011) and we consider this to be a promising direction to go forward with.

- **Richer models for capturing task relatedness in Multitask Learning:** Our model proposed in Chapter 6 provides considerable flexibility in terms of the latent task structures that can be exploited in Multitask Learning. It would be interesting to extend this work to allow more general structures such as time-varying tasks.

- **Efficient inference for nonparametric latent feature models:** Another interesting future direction would be to design new online inference methods for the nonparametric latent feature models, along the lines of recently proposed online variational inference methods for the Dirichlet Process and Hierarchical Dirichlet Process (Wang et al., 2011). In addition, it would also be interesting and useful to have the ability to perform hyperparameter estimation in the beam-search and the SMC-based inference for the IBP.

Another interesting direction that is currently emerging is about designing nonprobabilistic counterparts of nonparametric Bayesian models, which can be useful for scaling up nonparametric Bayesian methods to larger datasets. Some recent work has explored this direction for the Dirichlet Process mixture models (Kulis and Jordan, 2012) and we believe that similar developments for other nonparametric Bayesian models would be of interest for the general machine learning community.

APPENDIX

APPENDIX: NONPARAMETRIC MIXTURE OF SUBSPACES FOR MULTITASK LEARNING: INFERENCE

In this supplementary material, we derive the variational lower bound for our model presented in Chapter 8 and derive the update equation for all the parameters of our model.

A.1 The Model

The model for the nonparametric mixture of nonparametric factor analyzers over the latent weight vectors in our multitask learning framework is as described in the paper. For the variational approximation, we work with the following distribution:

$$
\begin{aligned}
\phi_f &\sim \mathcal{B}et(1, \alpha_1) \\
z_t &\sim \mathcal{M}ult(\phi_f \prod_{i<f}(1-\phi_i))^1 \\
\beta_{f,k} &\sim \mathcal{B}et(\alpha_2/K, 1) \\
b_{t,f,k} &\sim \mathcal{B}er(\beta_{f,k}) \\
\mu_f &\sim \mathcal{N}or(0, \mathbf{I}) \\
\Lambda_{f,k} &\sim \mathcal{N}or(0, \mathbf{I}) \\
s_{t,f} &\sim \mathcal{N}or(0, \mathbf{I}) \\
\theta_t &\sim \mathcal{N}or(\mu_{z_t} + \Lambda_{z_t}(s_{t,z_t} \odot b_{t,z_t}), \frac{1}{\sigma^2}\mathbf{I}) \\
Y_{t,i} &\sim \mathcal{N}or(\theta_t^T X_{t,i}, \mathbf{I}).
\end{aligned}
$$

We approximate this distribution the usual way with an approximating distribution Q. Since we are only interested in the predictive performance of the model, we do not model

the covariances of the gaussian variables of the approximating distribution explicitly.

A.2 The Variational Lower Bound

The variational lower bound, following (Jordan et al., 1999), is the following sum:

$$\begin{aligned}
\log P(Y|X) \quad \geq \quad & E_q[\log P(\phi)] - E_q[\log Q(\phi)] \\
& + E_q[\log P(\mu)] - E_q[\log Q(\mu)] \\
& + E_q[\log P(\Lambda)] - E_q[\log Q(\Lambda)] \\
& + E_q[\log P(z)] - E_q[\log Q(z)] \\
& + E_q[\log P(s)] - E_q[\log Q(s)] \\
& + E_q[\log P(b)] - E_q[\log Q(b)] \\
& + E_q[\log P(\beta)] - E_q[\log Q(\beta)] \\
& + E_q[\log P(\theta)] - E_q[\log Q(\theta)] \\
& + E_q[\log P(Y)]
\end{aligned}$$

Computing each term is a simple exponential family calculation, which we do in the following sections explicitly for the sake of clarity. Unless stated otherwise, the mean field parameter for the variable v is ν_v, so, for example, the mean of the variational distribution for θ_t is ν_{θ_t}. Note that, as we do not approximate the distribution of Y, there is no term for the entropy of $Q(Y)$.

A.2.1 The Bound for ϕ

ϕ are beta stick-breaking priors for the DP.

$$\begin{aligned}
E_q[\log Beta(1, \alpha_1)] - E[\log Beta(\gamma_{i,1}, \gamma_{i,2})] \quad = \quad & \log \Gamma(1 + \alpha_1) - \log \Gamma(\alpha_1) \\
& + (\alpha_1 - 1)(\digamma(\gamma_{i,2}) - \digamma(\gamma_{i,1} + \gamma_{i,2})) \\
& - \log \Gamma(\gamma_{i,1} + \gamma_{i,2}) \\
& + \log \Gamma(\gamma_{i,1}) + \log \Gamma(\gamma_{i,2}) \\
& - (\gamma_{i,1} - 1)(\digamma(\gamma_{i,1}) - \digamma(\gamma_{i,1} + \gamma_{i,2})) \\
& - (\gamma_{i,2} - 1)(\digamma(\gamma_{i,2}) - \digamma(\gamma_{i,1} + \gamma_{i,2}))
\end{aligned}$$

[1] This is a shortcut to the truncatead stick-breaking distribution, where the probability of z_t being equal to f is proportional to that value

A.2.2 The Bound for β

β are the symmetric dirichlet priors for the finite IBP variational inference; hence, there is one β for each component of each factor, and each $\beta_{f,k} \sim Beta(\alpha_2, 1)$. As we use a beta distribution as a mean-field for β, $q(\beta_{f,k}) = Beta(\rho_{f,k,1}, \rho_{f,k,2})$. The bound then is

$$
\begin{aligned}
= \quad & \log\Gamma(\alpha_2+1) - \log\Gamma(\alpha_2) \\
& +(\alpha_2-1)(\digamma(\rho_{f,k,1}) - \digamma(\rho_{i,1}+\rho_{f,k,2})) \\
& -\log\Gamma(\rho_{f,k,1}+\rho_{i,2}) + \log\Gamma(\rho_{f,k,1}) + \log\Gamma(\rho_{f,k,2}) \\
& -(\rho_{f,k,1}-1)(\digamma(\rho_{f,k,1}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2})) \\
& -(\rho_{f,k,2}-1)(\digamma(\rho_{f,k,2}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2})) \\
= \quad & \log\alpha_2 \\
& +(\alpha_2-1)(\digamma(\rho_{f,k,1}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2})) \\
& -\log\Gamma(\rho_{f,k,1}+\rho_{i,2}) + \log\Gamma(\rho_{f,k,1}) + \log\Gamma(\rho_{f,k,2}) \\
& -(\rho_{f,k,1}-1)(\digamma(\rho_{f,k,1}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2})) \\
& -(\rho_{f,k,2}-1)(\digamma(\rho_{f,k,2}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2}))
\end{aligned}
$$

A.2.3 The Bound for b

The b variables are the binary decision variables for the IBP-based latent factor analyzer; hence, we have a b for each task for each mixture component. The bound for the b variables is:

$$
\begin{aligned}
& \nu_{b_{t,f,k}}(\digamma(\rho_{f,k,1}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2})) \\
& +(1-\nu_{b_{t,f,k}})(\digamma(\rho_{f,k,2}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2})) \\
& -\nu_{b_{t,f,k}}\log\nu_{b_{t,f,k}} - (1-\nu_{b_{t,f,k}})\log(1-\nu_{b_{t,f,k}})
\end{aligned}
$$

A.2.4 The Bound for μ

$$
\begin{aligned}
& E_q[\log P(\mu)] - E_q[\log Q(\mu)] \\
= \quad & \sum_f \left(\int d\mu_f q(\mu_f)\log P(\mu_f) - \int d\mu_f q(\mu_f)\log Q(\mu_f) \right)
\end{aligned}
$$

Hence, we can work with each μ_f separately,

$$\begin{aligned} & \int d\mu_f q(\mu_f) \log P(\mu_f) - \int d\mu_f q(\mu_f) \log Q(\mu_f) \\ = & -\frac{D}{2}\log 2\pi - \frac{1}{2}||\nu_{\mu_f}||^2 - \frac{D}{2} + \frac{D}{2}\log 2\pi e \end{aligned}$$

A.2.5 The Bound for $\Lambda_{f,k}$

$$\begin{aligned} & E_q[\log P(\Lambda_{f,k})] - E_q[\log Q(\Lambda_{f,k})] \\ = & -\frac{D}{2}\log 2\pi + -\frac{1}{2}(||\nu_{\Lambda_{f,k}}||^2 + D) + \frac{D}{2}\log 2\pi e \end{aligned}$$

A.2.6 The Bound for z

$$\begin{aligned} & E_q[\log P(z)] - E_q[\log Q(z)] \\ = & \sum_{f=1}^{F}\left(\left(\sum_{j=f+1}^{F}\nu_{z_{t,j}}\right)(\digamma(\gamma_{f,2}) - \digamma(\gamma_{f,1}+\gamma_{f,2})) + \nu_{z_{t,f}}(\digamma(\gamma_{f,1}) - \digamma(\phi_{f,1}+\gamma_{f,2}))\right) \\ & -\sum_f \nu_{z_{t,f}}\log\nu_{z_{t,f}} \end{aligned}$$

A.2.7 The Bound for s

$$\begin{aligned} & E_q[\log P(s_{t,f})] - E_q[\log Q(s_{t,f})] \\ = & \int ds_{t,f} q(s_{t,f}) \log P(s_{t,f}) - \int ds_{t,f} q(s_{t,f}) \log Q(s_{t,f}) \\ = & -\frac{D}{2}\log 2\pi - \frac{1}{2}||\nu_{s_{t,f}}||^2 - \frac{D}{2} + \frac{D}{2}\log 2\pi e \end{aligned}$$

A.2.8 The Bound for θ

$$\begin{aligned} & E_q[\log P(\theta_t)] - E_q[\log Q(\theta_t)] \\ = & \sum_f \nu_{z_{t,f}}\left(\int d\theta_t\, d\Lambda\, d\mu\, ds\, q(\theta_t, \Lambda, \mu, s)\log P(\theta_t)\right) + \frac{D}{2}\log 2\pi e \end{aligned}$$

$$
\begin{aligned}
= \quad & \sum_f \nu_{z_{t,f}}(-\frac{D}{2}\log 2\pi + \frac{D}{2}\log\sigma \\
& -\frac{\sigma}{2}\int d\theta_t d\Lambda d\mu ds\; q(\theta_t,\Lambda,\mu,s)||\theta_t - \mu_f - \Lambda_f(s_{t,f}\odot b_{t,f})||^2) \\
& +\frac{D}{2}\log 2\pi e
\end{aligned}
$$

The main problem then is computing the expectation of $||\theta_t - \mu_f - \Lambda_f(s_{t,f}\odot d_{t,f})||^2$. This can be split in the following terms:

$$
\begin{aligned}
E_q[||\theta_t - \mu_f - \Lambda_f(s_{t,f}\odot b_{t,f})||^2] \quad = \quad & E_q[||\theta_t||^2] \\
& -2E_q[\theta_t^T\mu_f] \\
& -2E_q[\theta_t^T\Lambda_f(s_{t,f}\odot b_{t,f})] \\
& +2E_q[\mu_f^T\Lambda_f(s_{t,f}\odot b_{t,f})] \\
& +E_q[\mu_f^T\mu_f] \\
& +E_q[(\Lambda_f(s_{t,f}\odot b_{t,f}))^T\Lambda_f(s_{t,f}\odot b_{t,f})]
\end{aligned}
$$

and all terms except for the last one are trivial as they are either linear or the expectation of the norm of a normally distributed variable. The last term can be solved as follows,

$$
E_q[(\Lambda_f(s_{t,f}\odot b_{t,f}))^T\Lambda_f(s_{t,f}\odot b_{t,f})] \quad = \quad E_q[(s_{t,f}\odot b_{t,f})\Lambda_f^T\Lambda_f(s_{t,f}\odot b_{t,f})]
$$

then we can split this expectation into two sums,

$$
\begin{aligned}
E_q[(s_{t,f}\odot b_{t,f})\Lambda_f^T\Lambda_f(s_{t,f}\odot b_{t,f})] \quad = \quad & E_q[\sum_i s_{t,f,i}^2 b_{t,f,i}^2\Lambda_{f,i}^T\Lambda_{f,i}] \\
& +E_q[\sum_i s_{t,f,i}b_{t,f,i}\sum_{j\neq i} s_{t,f,j}b_{t,f,j}\Lambda_{f,i}^T\Lambda_{f,j}]
\end{aligned}
$$

Now with the linearity of expectation, we can solve the second expectation, which is

$$
\sum_i \nu_{s_{t,f,i}}\nu_{b_{t,f,i}}\sum_{j\neq i}\nu_{s_{t,f,j}}\nu_{b_{t,f,j}}\nu_{\Lambda_{f,i}}^T\nu_{\Lambda_{f,j}}
$$

and the first expectation, after summing over b and s is

$$
\sum_i(\nu_{s_{t,f,i}}^2 + 1)\nu_{b_{t,f,i}}E_q[\Lambda_{f,i}^T\Lambda_{f,i}]
$$

and after solving the last expectation, we get

$$\sum_i (\nu^2_{s_{t,f,i}} + 1)\nu_{b_{t,f,i}}(\nu^T_{\Lambda_{f,i}}\nu_{\Lambda_{f,i}} + D)$$

which we can expand to

$$\sum_i \nu^2_{s_{t,f,i}}\nu_{b_{t,f,i}}\nu^T_{\Lambda_{f,i}}\nu_{\Lambda_{f,i}} + \sum_i \nu^2_{s_{t,f,i}}\nu_{b_{t,f,i}}D + \sum_i \nu_{b_{t,f,i}}\nu^T_{\Lambda_{f,i}}\nu_{\Lambda_{f,i}} + \nu_{b_{t,f,i}}DF$$

The full lower bound for θ then is

$$\begin{aligned}
E_q[\log P(\theta)] - E_q[\log Q(\theta)] \quad = \quad & -0.5\log 2\pi + 0.5D\log 2\pi e + 0.5D\log\sigma \\
& -0.5\sigma(\\
& \quad \nu^T_{\theta_t}\nu_{\theta_t} + D \\
& \quad - 2\nu^T_{\theta_t}\nu_{\mu_f} \\
& \quad - 2\nu^T_{\theta_t}\nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}}) \\
& \quad + 2\nu^T_{\mu_f}\nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}}) \\
& \quad + \nu^T_{\mu_f}\nu_{\mu_f} + D \\
& \quad + \sum_i \nu^2_{s_{t,f,i}}\nu_{b_{t,f,i}}\nu^T_{\Lambda_{f,i}}\nu_{\Lambda_{f,i}} \\
& \quad + \sum_i \nu^2_{s_{t,f,i}}\nu_{b_{t,f,i}}D \\
& \quad + \sum_i \nu_{b_{t,f,i}}\nu^T_{\Lambda_{f,i}}\nu_{\Lambda_{f,i}} \\
& \quad + \nu_{b_{t,f,i}}DF \\
& \quad + \sum_i \nu_{s_{t,f,i}}\nu_{b_{t,f,i}}\sum_{j\neq i}\nu_{s_{t,f,j}}\nu_{b_{t,f,j}}\nu^T_{\Lambda_{f,i}}\nu_{\Lambda_{f,j}})
\end{aligned}$$

A.2.9 The Lower Bound for Y

We compute

$$\begin{aligned}
E_q[\log P(Y)] \quad &= \quad \int d\theta_t q(\theta_t)\log P(Y|X,\theta_t) \\
&= \quad -\frac{D}{2}\log 2\pi - \frac{1}{2}Y^2 + Y\nu^T_{\theta_t}X - \frac{1}{2}X^TX - \frac{1}{2}X^T\nu_{\theta_t}\nu^T_{\theta_t}X
\end{aligned}$$

A.2.10 The Complete Lower Bound

which we can simplify to (omitting constant terms)

$$
\begin{aligned}
\log P(Y|X) \quad \geq \quad & \log\Gamma(1+\alpha_1) - \log\Gamma(\alpha_1) \\
& +(\alpha_1 - 1)(\digamma(\gamma_{i,2}) - \digamma(\gamma_{i,1}+\gamma_{i,2})) \\
& - \log\Gamma(\gamma_{i,1}+\gamma_{i,2}) + \log\Gamma(\gamma_{i,1}) + \log\Gamma(\gamma_{i,2}) \\
& -(\gamma_{i,1}-1)(\digamma(\gamma_{i,1}) - \digamma(\gamma_{i,1}+\gamma_{i,2})) \\
& -(\gamma_{i,2}-1)(\digamma(\gamma_{i,2}) - \digamma(\gamma_{i,1}+\gamma_{i,2})) \\
& -\frac{1}{2}\sum_f ||\nu_{\mu_f}||^2 \\
& +\sum_f(\sum_k(\log\Gamma(\alpha_2+1) - \log\Gamma(\alpha_2) \\
& +(\alpha_2-1)(\digamma(\rho_{f,k,1}) - \digamma(\rho_{i,1}+\rho_{f,k,2})) \\
& - \log\Gamma(\rho_{f,k,1}+\rho_{i,2}) + \log\Gamma(\rho_{f,k,1}) + \log\Gamma(\rho_{f,k,2}) \\
& -(\rho_{f,k,1}-1)(\digamma(\rho_{f,k,1}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2})) \\
& -(\rho_{f,k,2}-1)(\digamma(\rho_{f,k,2}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2})) \\
& -\frac{1}{2}(||\nu_{\Lambda_{f,k}}||^2 + D)) \\
& +\sum_t\sum_f\sum_k(\nu_{b_{t,f,k}}(\digamma(\rho_{f,k,1}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2})) \\
& \quad + (1-\nu_{b_{t,f,k}})(\digamma(\rho_{f,k,2}) - \digamma(\rho_{f,k,1}+\rho_{f,k,2})) \\
& \quad - \nu_{b_{t,f,k}}\log\nu_{b_{t,f,k}} - (1-\nu_{b_{t,f,k}})\log(1-\nu_{b_{t,f,k}})) \\
& +\sum_t(\sum_f(\sum_{j=f+1}^{F}\nu_{z_{t,j}})(\digamma(\gamma_{f,2}) - \digamma(\gamma_{f,1}+\gamma_{f,2})) \\
& \quad - 0.5||\nu_{s_{t,f}}||^2 \\
& \quad + \nu_{z_{t,f}}(\\
& \quad\quad \digamma(\gamma_{f,1}) - \digamma(\phi_{f,1}+\gamma_{f,2}) \\
& \quad\quad - \log\nu_{z_{t,f}} - 0.5D\log\sigma \\
& \quad\quad - 0.5\sigma(\\
& \quad\quad\quad \nu_{\theta_t}^T\nu_{\theta_t} + D - 2\nu_{\theta_t}^T\nu_{\mu_f} - 2\nu_{\theta_t}^T\nu_{\Lambda_f}(\nu_{s_{t,f}}\odot\nu_{b_{t,f}}) \\
& \quad\quad\quad + 2\nu_{\mu_f}^T\nu_{\Lambda_f}(\nu_{s_{t,f}}\odot\nu_{b_{t,f}}) + \nu_{\mu_f}^T\nu_{\mu_f} + D
\end{aligned}
$$

$$
\begin{aligned}
&+ \sum_i \nu^2_{s_{t,f,i}} \nu_{b_{t,f,i}} \nu^T_{\Lambda_{f,i}} \nu_{\Lambda_{f,i}} \\
&+ \sum_i \nu^2_{s_{t,f,i}} \nu_{b_{t,f,i}} D + \sum_i \nu_{b_{t,f,i}} \nu^T_{\Lambda_{f,i}} \nu_{\Lambda_{f,i}} \\
&+ \nu_{b_{t,f,i}} DF \\
&+ \sum_i \nu_{s_{t,f,i}} \nu_{b_{t,f,i}} \sum_{j \neq i} \nu_{s_{t,f,j}} \nu_{b_{t,f,j}} \nu^T_{\Lambda_{f,i}} \nu_{\Lambda_{f,j}}) \\
+ \sum_i (&-\frac{1}{2} Y^2 + Y \nu^T_{\theta_t} X \\
&- \frac{1}{2} X^T X - \frac{1}{2} X^T \nu_{\theta_t} \nu^T_{\theta_t} X))
\end{aligned}
$$

To optimize the lower bound with respect to the variational parameters, we can take the gradients of the lower bound w.r.t. each parameter and set it to zero. Alternating this for every parameter, we have the usual variational mean field optimization algorithm. We also compute empirical bayes estimates for σ in the same fashion. For ν_{θ_t}, however, we found numerical instabilities in inverting the matrices required to compute the update, so we resorted to numerical maximization by the L-BFGS algorithm (Zhu et al., 1997).

A.3 Update Equations for Specific Parameters

A.3.1 Updates for γ

The updates for γ, following (Blei and Jordan, 2006), are

$$
\begin{aligned}
\gamma_{i,1} &= 1 + \sum_t \nu_{z_{t,f}} \\
\gamma_{i,2} &= \alpha_1 + \sum_t \sum_{j>i} \nu_{z_{t,j}}.
\end{aligned}
$$

A.3.2 Updates for ν_{z_t}

Also following (Blei and Jordan, 2006), the update for $\nu_{z_{t,i}}$ is

$$
\log \nu_{z_{t,i}} \propto \Gamma(\gamma_{i,1}) - \Gamma(\gamma_{i,1} + \gamma_{i,2}) + E_Q[\log P(\theta_t | z_t = i)] + \sum_{j<i} \left(\Gamma(\gamma_{j,2}) - \Gamma(\gamma_{j,1} + \gamma_{j,2}) \right).
$$

A.3.3 Updates for ρ

Following (Doshi-Velez et al., 2009b), the update for ρ is

$$
\rho_{f,k,1} = \frac{\alpha}{K} + \sum_t \nu_{b_{t,f,k}}
$$

$$\rho_{f,k,2} = 1 + \sum_t (1 - \nu_{b_{t,f,k}}).$$

A.3.4 Updates for $\nu_{b_{t,f,k}}$

Also following (Doshi-Velez et al., 2009b), the update for $\nu_{b_{t,f,k}}$ is

$$\begin{aligned}
\log \frac{\nu_{b_{t,f,i}}}{1 - \nu_{b_{t,f,i}}} &= \digamma(\rho_{f,k,1}) - \digamma(\rho_{f,k,2}) \\
&+0.5\nu_{z_{t,f}}\sigma(2(\nu_{\theta_t} - \nu_{\mu_f} - (\nu_{s_{t,f,i}} + 1)\nu_{\Lambda_{f,i}} \\
&- \sum_f \nu_{s_{t,f,j}}\nu_{b_{t,f,j}}\nu_{\Lambda_{f,j}})^T \nu_{\Lambda_{f,i}}\nu_{s_{t,f,i}} \\
&-\nu^2_{s_{t,f,i}} D - DF)
\end{aligned}$$

A.3.5 Updates for $\nu_{s_{t,f}}$

Taking the gradient of the lower bound with respect to a single $\nu_{s_{t,i}}$ and setting it to zero, we find that

$$\begin{aligned}
0 &= -\nu_{s_{t,f,i}} + \nu_{z_{t,f}}(\sigma((\nu_{\theta_t} - \nu_{\mu_f})^T \nu_{\Lambda_{f,i}}\nu_{b_{t,f,i}} \\
&-\nu_{s_{t,f,i}}\nu_{b_{t,f,i}}(D + \nu^T_{\Lambda_{f,i}}\nu_{\Lambda_{f,i}}) \\
&-0.5\nu_{b_{t,f,i}} \sum_{j \neq i} \nu_{s_{t,f,j}}\nu_{b_{t,f,j}}\nu^T_{\Lambda_{f,i}}\nu_{\Lambda_{f,j}} \\
(1 + \sigma\nu_{z_{t,f}}\nu_{b_{t,f,i}}(D + ||\nu_{\Lambda_{f,i}}||^2))\nu_{s_{t,f,i}} &= \nu_{z_{t,f}}\sigma((\nu_{\theta_t} - \nu_{\mu_f})^T \nu_{\Lambda_{f,i}}\nu_{b_{t,f,i}} \\
&-0.5\nu_{b_{t,f,i}} \sum_{j \neq i} \nu_{s_{t,f,j}}\nu_{b_{t,f,j}}\nu^T_{\Lambda_{f,i}}\nu_{\Lambda_{f,j}}).
\end{aligned}$$

A.3.6 Updates for ν_{μ_f}

Doing similarly for ν_{μ_f}, we find that

$$\begin{aligned}
0 &= -\nu_{\mu_f} + \sum_t \nu_{z_{t,f}}(\sigma(\nu_{\theta_t} - \nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}}) - \nu_{\mu_f})) \\
(1 + \sigma \sum_t \nu_{z_{t,f}})\nu_{\mu_f} &= \sum_t \nu_{z_{t,f}}\sigma(\nu_{\theta_t} - \nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}})).
\end{aligned}$$

A.3.7 Updates for ν_{Λ_f}

Taking the gradient of the lower bound with respect to a single $\nu_{\Lambda_{f,i}}$ we find that

$$0 = -\nu_{\Lambda_{f,i}} + \sum_t (\nu_{z_{t,f}}(\sigma(+\nu_{s_{t,f,i}}\nu_{b_{t,f,i}}\nu_{\theta_t} - \nu_{s_{t,f,i}}\nu_{b_{t,f,i}}\nu_{\mu_f}$$

$$
\begin{aligned}
& & -\nu^2_{s_{t,f,i}}\nu_{b_{t,f,i}}\nu_{\Lambda_{f,i}} - \nu_{b_{t,f,i}}\nu_{\Lambda_{f,i}} \\
& & -0.5\nu_{s_{t,f,i}}\nu_{b_{t,f,i}}\sum_{j\neq i}\nu_{s_{t,f,j}}\nu_{b_{t,f,j}}\nu_{\Lambda_{f,j}}) \\
(1+\sigma\sum_t \nu_{z_{t,f}}\nu_{b_{t,f,i}}(1+\nu^2_{s_{t,f,i}}))\nu_{\Lambda_{f,i}} & = & \sigma\sum_t \nu_{z_{t,f}}\nu_{s_{t,f,i}}\nu_{b_{t,f,i}}(\nu_{\theta_t}-\nu_{\mu_f} \\
& & -0.5\sum_{j\neq i}\nu_{s_{t,f,j}}\nu_{b_{t,f,j}}\nu_{\Lambda_{f,j}}).
\end{aligned}
$$

A.3.8 Updates for ν_{θ_t}

The analytical update for ν_{θ_t} would be

$$
\begin{aligned}
0 & = & \sum_f \nu_{z_{t,f}}(-\sigma(\nu_{\theta_t}-\nu_{\mu_f}-\nu_{\Lambda_f}(\nu_{s_{t,f}}\odot\nu_{b_{t,f}}))) \\
& & +\sum_i(+YX-\frac{1}{2}XX^T\nu_{\theta_t}) \\
(\sigma\sum_f \nu_{z_{t,f}}I+0.5XX^T)\nu_{\theta_t} & = & \sigma\sum_f \nu_{z_{t,f}}(\nu_{\mu_f}+\nu_{\Lambda_f}(\nu_{s_{t,f}}\odot\nu_{b_{t,f}}))+\sum_i Y_iX_i.
\end{aligned}
$$

However, as mentioned above, we use a numerical maximization of the lower bound due to numerical instability when inverting the matrix

$$\sigma\sum_f \nu_{z_{t,f}}I+0.5XX^T.$$

The gradient of the lower bound with respect to ν_{θ_t} is

$$\nabla L(\nu_{\theta_t}) = \sigma\sum_f \nu_{z_{t,f}}\left(\nu_{\theta_t}-\nu_{\mu_f}-\nu_{\Lambda_f}(\nu_{s_{t,f}}\odot\nu_{b_{t,f}})\right)+\sum_i\left(Y_{t,i}X_{t,i}-X_{t,i}X^T_{t,i}\nu_{\theta_t}\right).$$

A.4 Logistic Regression

All that changes in the model when we switch from squared loss regression to logistic regression is the conditional distribution $P(Y|X,\theta)$. In logistic regression, this is normally a logistic distribution:

$$P(Y|X,\theta) = \frac{1}{1+\exp(-Y\theta^TX)} = sig(Y\theta^TX)$$

Unfortunately, it is not easy to compute the expectation of this distribution w.r.t. the mean-field of theta, since the logistic distribution is not conjugate to the normal distribution.

Here, we follow Jaakkola and Jordan (Jaakkola and Jordan, 1996) and use the following variational lower-bound of the logistic function:

$$P(Y|X,\theta) \geq sig(\epsilon)\exp\left(\frac{Y\theta^T X - \epsilon}{2} - \frac{1}{2\epsilon}\left[sig(\epsilon) - \frac{1}{2}\right]((\theta^T X)^2 - \epsilon^2)\right).$$

As this is a quatratic in terms of θ, we can compute the last integral in the lower bound as

$$\begin{aligned}
& \int d\theta N(\theta;\nu_{\theta_t},I)\log sig(\epsilon)\exp\left(\frac{Y\theta^T X - \epsilon}{2} - \frac{1}{2\epsilon}\left[sig(\epsilon) - \frac{1}{2}\right]((\theta^T X)^2 - \epsilon^2)\right) \\
= \; & \log sig(\epsilon) + \int d\theta N(\theta;\nu_{\theta_t},I)\left(\frac{Y\theta^T X - \epsilon}{2} - \frac{1}{2\epsilon}\left[sig(\epsilon) - \frac{1}{2}\right]((\theta^T X)^2 - \epsilon^2)\right) \\
= \; & \log sig(\epsilon) + \frac{\nu_{\theta_t}^T X - \epsilon}{2} - \frac{1}{2\epsilon}\left[sig(\epsilon) - \frac{1}{2}\right]\left(X^T X + (\nu_{\theta_t}^T X)^2 - \epsilon^2\right)
\end{aligned}$$

The bound is exact whenever $\epsilon^2 = (\nu_{\theta_t}^T X)^2 + X^T X$. To optimize ν_{θ_t}, the term involved in the gradient is then $\frac{X}{2} - \lambda(\epsilon)2X\nu_{\theta_t}^T X$, where $\lambda(\epsilon) = \frac{1}{2\epsilon}\left[\frac{1}{2} - sig(\epsilon)\right]$.

The gradient of the lower bound with respect to ν_{θ_t} in the case of logistic regression then is

$$\nabla L(\nu_{\theta_t}) = \sigma\sum_f \nu_{z_{t,f}}\left(\nu_{\theta_t} - \nu_{\mu_f} - \nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}})\right) + \sum_i \left(Y_{t,i}X_{t,i} - \lambda(\epsilon)X_{t,i}X_{t,i}^T\nu_{\theta_t}\right).$$

A.5 Optimizing the σ Hyperparameter

We can also optimize an empirical bayes estimate of the σ hyperparameter by optimizing the lower bound with respect to it. Setting the gradient of the lower bound w.r.t. σ to zero gives the following expression for $\frac{1}{\sigma}$

$$\frac{(\sum_t\sum_f \nu_{z_{t,f}}(||\nu_{\theta_t} - \nu_{\mu_f} - \nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}})||^2 + \sum_i(\nu_{s_{t,f,i}}^2\nu_{b_{t,f,i}}D + \nu_{b_{t,f}}||\nu_{\Lambda_{f,i}}||^2) + \nu_{b_{t,f,i}}DF))}{KDF}$$

REFERENCES

ABERDEEN, D., PACOVSKY, O., AND SLATER, A. 2011. The learning behind gmail priority inbox. In *NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*.

AGARWAL, A., GERBER, S., AND DAUMÉ III, H. 2010. Learning multiple tasks using manifold regularization. In *NIPS*.

AHARON, M., ELAD, M., AND BRUCKSTEIN, A. 2010. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *JMLR*.

ANDO, R. K. AND ZHANG, T. 2005. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *The Journal of Machine Learning Research 6*, 1817–1853.

ARCHAMBEAU, C. AND BACH, F. 2008. Sparse probabilistic projections. In *NIPS*.

ARENAS-GARCÍA, J., PETERSEN, K. B., AND HANSEN, L. K. 2006. Sparse kernel orthonormalized pls for feature extraction in large data sets. In *NIPS*.

ARGYRIOU, A., EVGENIOU, T., AND PONTIL, M. 2007. Multi-task feature learning. In *NIPS*.

ARGYRIOU, A., MAURER, A., AND PONTIL, M. 2008. An algorithm for transfer learning in a heterogeneous environment. In *ECML*.

BACH, F. R. AND JORDAN, M. I. 2003. Beyond independent components: trees and clusters. *Journal of Machine Learning Research*, 1205–1233.

BACH, F. R. AND JORDAN, M. I. 2005. A Probabilistic Interpretation of Canonical Correlation Analysis. In *Technical Report 688, Dept. of Statistics*. University of California.

BARNETT, S. 1979. *Matrix Methods for Engineers and Scientists*. McGraw-Hill.

BARTHOLOMEW, D. J. AND KNOTT, M. 1999. *Latent Variable Models and Factor Analysis (2nd ed.)*. Oxford University Press.

BAXTER, J. 2000. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research 12*, 149–198.

BEAL, M. J., FALCIANI, F., GHAHRAMANI, Z., RANGEL, C., AND WILD, D. L. 2005. A Bayesian Approach to Reconstructing Genetic Regulatory Networks with Hidden Factors. *Bioinformatics 21*, 3.

BISHOP, C. 2006. *Pattern recognition and machine learning*. Springer New York.

BISHOP, C. M. 1999. Bayesian PCA. In *NIPS*.

BLEI, D. AND JORDAN, M. 2006. Variational inference for Dirichlet process mixtures. *Bayesian Analysis 1*, 1.

BONILLA, E. V., CHAI, K. M. A., AND WILLIAMS, C. K. I. 2007. Multi-task gaussian process prediction. In *NIPS*.

BREIMAN, L. AND FRIEDMAN, J. 1997. Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society. Series B (Methodological)*, 3–54.

CANINI, K., SHASHKOV, M., AND GRIFFITHS, T. 2010. Modeling transfer learning in human categorization with the hierarchical Dirichlet process. In *ICML*.

CARUANA, R. 1997. Multitask Learning. *Machine Learning 28*, 1, 41–75.

CARVALHO, C., LUCAS, J., WANG, Q., CHANG, J., NEVINS, J., AND WEST, M. 2008. High-Dimensional Sparse Factor Modelling - Applications in Gene Expression Genomics. In *JASA*.

CHELBA, C. AND ACERO, A. 2006. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language 20*, 4, 382–399.

CHEN, M., SILVA, J., PAISLEY, J., WANG, C., DUNSON, D., AND CARIN, L. 2010. Compressive Sensing on Manifolds Using a Nonparametric Mixture of Factor Analyzers: Algorithm and Performance Bounds. *IEEE Transactions on Signal Processing*.

D., N., QI, Y., XIANG, R., MOLLOY, I., AND LI, N. 2010. Nonparametric Bayesian Matrix Factorization by Power-EP. In *Proceedings of the Conference on Artificial Intelligence and Statistics (AISTATS)*.

DAUMÉ III, H. 2007. Fast Search for Dirichlet Process Mixture Models. In *Proceedings of the Conference on Artificial Intelligence and Statistics (AISTATS)*.

DAUMÉ III, H. 2009. Bayesian Multitask Learning with Latent Hierarchies. In *Conference on Uncertainty in Artificial Intelligence*. Montreal, Canada.

DOSHI, F. AND GHAHRAMANI, Z. 2009a. Accelerated Gibbs Sampling for the Indian Buffet Process. In *ICML*.

DOSHI, F. AND GHAHRAMANI, Z. 2009b. Correlated Non-Parametric Latent Feature Models. In *UAI*.

DOSHI-VELEZ, F. AND GHAHRAMANI, Z. 2009. Accelerated Sampling for the Indian Buffet Process. In *Proceedings of the International Conference on Machine Learning (ICML)*.

DOSHI-VELEZ, F., KNOWLES, D., MOHAMED, S., AND GHAHRAMANI, Z. 2009a. Large Scale Nonparametric Bayesian Inference: Data Parallelisation in the Indian Buffet Process. In *NIPS*.

DOSHI-VELEZ, F., MILLER, K., VAN GAEL, J., TEH, Y., AND UNIT, G. 2009b. Variational inference for the Indian buffet process. In *AISTATS*.

DOSHI-VELEZ, F., MILLER, K. T., GAEL, J. V., AND TEH, Y. W. 2009c. Variational Inference for the Indian Buffet Process. In *Proceedings of the Conference on Artificial Intelligence and Statistics (AISTATS)*.

DOUCET, A., DE FREITAS, N., AND GORDON, N. 2001. *Sequential Monte Carlo Methods in Practice*. Springer.

EVGENIOU, T., MICCHELLI, C., AND PONTIL, M. 2006. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*.

FEARNHEAD, P. 2004. Particle filters for mixture models with an unknown number of components. In *Journal of Statistics and Computing*. 11–21.

FERGUSON, T. 1973. A Bayesian analysis of some nonparametric problems. *The annals of statistics*.

FUKUMIZU, K., BACH, F. R., AND JORDAN, M. I. 2004. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *J. Mach. Learn. Res. 5*, 73–99.

GERSHMAN, S. J. AND BLEI, D. M. 2012. A Tutorial on Bayesian Nonparametric Models. In *Journal of Mathematical Psychology*.

GHAHRAMANI, Z. AND BEAL, M. J. 2000. Variational inference for bayesian mixtures of factor analysers. In *NIPS*.

GHAHRAMANI, Z., GRIFFITHS, T., AND SOLLICH, P. 2007. Bayesian Nonparametric Latent Feature Models. In *Bayesian Statistics 8. Oxford University Press*.

GHAHRAMANI, Z. AND HINTON, G. E. 1997. The em algorithm for mixtures of factor analyzers. Tech. rep.

GHOSN, J. AND BENGIO, Y. 2003. Bias learning, knowledge sharing. *IJCNN*.

GLOBERSON, A. AND TISHBY, N. 2003. Sufficient dimensionality reduction. *J. Mach. Learn. Res. 3*, 1307–1331.

GREEN, P. 1995. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrica 82*.

GRIFFITHS, T. L. AND GHAHRAMANI, Z. 2011. The Indian Buffet Process: An Introduction and Review. In *Journal of Machine Learning Research 12*. 11851224.

HESKES, T. 2000. Empirical Bayes for learning to learn. *In Proc. of the 17th ICML*.

HOLMES, C. C. AND HELD, L. 2006. Bayesian Auxiliary Variable Models for Binary and Multinomial Regression. In *Bayesian Statistics 8. Oxford University Press*.

HOTELLING, H. 1936. Relations Between Two Sets of Variables. *Biometrika*, 321–377.

ISHWARAN, H. AND JAMES, L. 2001. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association 96*, 453, 161–173.

JAAKKOLA, T. S. AND JORDAN, M. I. 1996. A variational approach to bayesian logistic regression models and their extensions. In *AISTATS*.

JACOB, L. AND BACH, F. 2008. Clustered multi-task learning: a convex formulation. In *NIPS*.

JI, S., TANG, L., YU, S., AND YE, J. 2008. Extracting Shared Subspace for Multi-label Classification.

JI, S. AND YE, J. 2009. Linear dimensionality reduction for multi-label classification. In *Twenty-first International Joint Conference on Artificial Intelligence*.

JORDAN, M., GHAHRAMANI, Z., JAAKKOLA, T., AND SAUL, L. 1999. An introduction to variational methods for graphical models. *Machine learning 37*, 2, 183–233.

KANG, Z., GRAUMAN, K., AND SHA, F. 2011. Learning with whom to share in multi-task feature learning. In *ICML*.

KIM, M. AND PAVLOVIC, V. 2009. Covariance operator based dimensionality reduction with extension to semi-supervised settings. In *Twelfth International Conference on Artificial Intelligence and Statistics*. Florida USA.

KINGMAN, J. F. C. 1982. The coalescent. *Stochastic Processes and their Applications*.

KLAMI, A. AND KASKI, S. 2007. Local dependent components. In *ICML '07: Proceedings of the 24th international conference on Machine learning*.

KNOWLES, D. AND GHAHRAMANI, Z. 2007. Infinite Sparse Factor Analysis and Infinite Independent Components Analysis. In *ICA 2007*.

KNOWLES, D. AND GHAHRAMANI, Z. 2011. Nonparametric Bayesian Sparse Factor Models with application to Gene Expression modelling. In *Annals of Applied Statistics*.

KULIS, B. AND JORDAN, M. I. 2012. Revisiting k-means: New Algorithms via Bayesian Nonparametrics. In *ICML*.

KUMAR, A. AND DAUMÉ III, H. 2012. Learning task grouping and overlap in multi-task learning. In *ICML*.

LAWRENCE, N. AND PLATT, J. 2004. Learning to learn with the informative vector machine. In *Proceedings of the twenty-first international conference on Machine learning*. ACM New York, NY, USA.

LOPES, H. F., CARVALHO, C. M., JOHANNES, M. S., AND POLSON, N. G. 2011. Particle Learning for Sequential Bayesian Computation. In *Bayesian Statistics 9*. 317–360.

LOUNICI, K., PONTIL, M., TSYBAKOV, A. B., AND GEER, S. 2009. Taking Advantage of Sparsity in Multi-Task Learning. In *arXiv:0903.1468v1 [stat.ML]*.

MACEACHERN, S. N., CLYDE, M., AND LIU, J. 1999. Sequential importance sampling for nonparametric bayes models: the next generation. In *The Canadian Journal of Statistics (Vol 27)*. 251267.

MEEDS, E., GHAHRAMANI, Z., NEAL, R., AND ROWEIS, S. 2006. Modeling Dyadic Data with Binary Latent Factors. In *NIPS*.

MEEDS, E., GHAHRAMANI, Z., NEAL, R. M., AND ROWEIS, S. T. 2007. Modeling Dyadic Data with Binary Latent Factors. In *NIPS*.

MINKA, T. 2001. Expectation Propagation for approximate Bayesian inference. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

NING, X. AND KARYPIS, G. 2010. Multi-task learning for recommender systems. *JMLR*.

PAISLEY, J., CARIN, L., AND BLEI, D. M. 2011a. Variational Inference for Stick-Breaking Beta Process Priors. In *Proceedings of the International Conference on Machine Learning (ICML)*.

PAISLEY, J., WANG, C., AND BLEI, D. M. 2011b. The discrete infinite logistic normal distribution for mixed-membership modeling. In *AISTATS*.

PAISLEY, J., ZAAS, A., WOODS, C., GINSBURG, G., AND CARIN, L. 2010. A stick-breaking construction of the beta process. In *Proceedings of the International Conference on Machine Learning (ICML)*.

POURNARA, I. AND WERNISCH, L. 2007. Factor Analysis for Gene Regulatory Networks and Transcription Factor Activity Profiles. *BMC Bioinformatics*.

RAI, P. AND DAUMÉ III, H. 2008. The infinite hierarchical factor regression model. In *NIPS*.

RAI, P. AND DAUMÉ III, H. 2010. Infinite predictor subspace models for multitask learning. In *AISTATS*.

RAI, P. AND DAUMÉ III, H. 2011. Beam Search based MAP Estimates for the Indian Buffet Process. In *Proceedings of the International Conference on Machine Learning (ICML)*.

RAINA, R., NG, A., AND KOLLER, D. 2006. Constructing informative priors using transfer learning. In *ICML*.

RUSSELL, S. J. AND NORVIG, P. 2003. *Artificial Intelligence: a modern approach* 2nd international edition Ed. Prentice Hall.

SABATTI, C. AND JAMES, G. 2005. Bayesian Sparse Hidden Components Analysis for Transcription Regulation Networks,. *Bioinformatics 22*.

SANGUINETTI, G., LAWRENCE, N. D., AND RATTRAY, M. 2006. Probabilistic Inference of Transcription Factor Concentrations and Gene-specific Regulatory Activities. *Bioinformatics 22*, 22.

SCHÖLKOPF, B., HERBRICH, R., AND SMOLA, A. J. 2001. A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*.

SHAWE-TAYLOR, D. H. J. 2008. The Double-Barrelled LASSO (Sparse Canonical Correlation Analysis). In *Workshop on Learning from Multiple Sources (NIPS)*.

SNELSON, E. AND GHAHRAMANI, Z. 2005. Compact Approximations to Bayesian Predictive Distributions. In *ICML*.

SRIPERUMBUDUR, B., TORRES, D., AND LANCKRIET, G. 2009. The Sparse Eigenvalue Problem. In *arXiv:0901.1504v1*.

TEH, Y., SEEGER, M., AND JORDAN, M. 2005. Semiparametric latent factor models. In *Conference on Artificial Intelligence and Statistics*. Vol. 10.

TEH, Y. W., DAUMÉ III, H., AND ROY, D. M. 2008. Bayesian Agglomerative Clustering with Coalescents. In *NIPS*.

TEH, Y. W., GÖRÜR, D., AND GHAHRAMANI, Z. 2007a. Stick-breaking construction for the Indian buffet process. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. Vol. 11.

TEH, Y. W., GÖRÜR, D., AND GHAHRAMANI, Z. 2007b. Stick-breaking Construction for the Indian Buffet Process. In *AISTATS*.

TISHBY, N., PEREIRA, F. C., AND BIALEK, W. The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*. 368–377.

UEDA, N. AND SAITO, K. 2003. Parametric Mixture Models for Multi-labeled Text. *NIPS*.

ULKER, Y., GUNSEL, B., AND CEMGIL, T. 2010. Sequential monte carlo samplers for dirichlet process mixtures. In *AISTATS*.

VERBEEK, J. J., ROWEIS, S. T., AND VLASSIS, N. 2004. Non-linear CCA and PCA by Alignment of Local Models. In *NIPS*.

WANG, C. 2007. Variational Bayesian approach to Canonical Correlation Analysis. In *IEEE Transactions on Neural Networks*.

WANG, C., PAISLEY, J., AND BLEI, D. M. 2011. Online variational inference for the hierarchical Dirichlet process. In *AISTATS*.

WANG, L. AND DUNSON, D. B. 2011. Fast Bayesian Inference in Dirichlet Process Mixture Models. In *Journal of Computational and Graphical Statistics 20(1)*. 196–216.

WEST, M. 2003. Bayesian Factor Regression Models in the “Large p, Small n” Paradigm. In *Bayesian Statistics 7*.

WIESEL, A., KLIGER, M., AND HERO, A. 2008. A Greedy Approach to Sparse Canonical Correlation Analysis. In *arXiv:0801.2748*.

WOOD, F. AND GRIFFITHS, T. L. 2007. Particle Filtering for Nonparametric Bayesian Matrix Factorization. In *NIPS*.

XUE, Y., DUNSON, D., AND CARIN, L. 2007a. The matrix stick-breaking process for flexible multi-task learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*.

XUE, Y., LIAO, X., CARIN, L., AND KRISHNAPURAM, B. 2007b. Multi-task Learning for Classification with Dirichlet Process Priors. *The Journal of Machine Learning Research 8*, 35–63.

YANG, Y. 1997. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval 1*, 67–88.

YU, K., YU, S., AND TRESP, V. 2005. Multi-label Informed Latent Semantic Indexing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM New York, NY, USA, 258–265.

YU, S., YU, K., TRESP, V., KRIEGEL, H., AND WU, M. 2006. Supervised Probabilistic Principal Component Analysis. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*.

ZHANG, J., GHAHRAMANI, Z., AND YANG, Y. 2006. Learning multiple related tasks using latent independent component analysis. In *NIPS*.

ZHANG, J., GHAHRAMANI, Z., AND YANG, Y. 2008. Flexible latent variable models for multi-task learning. *Machine Learning Journal 73*, 3.

ZHANG, X., DUNSON, D., AND CARIN, L. 2011. Tree-Structured Infinite Sparse Factor Model. In *ICML*.

ZHANG, Y. AND YEUNG, D. 2010. A convex formulation for learning task relationships in multi-task learning. In *UAI*.

ZHOU, Y. Z. Z. H. 2008. Multi-Label Dimensionality Reduction via Dependence Maximization. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*. 1503–1505.

ZHU, C., BYRD, R., LU, P., AND NOCEDAL, J. 1997. L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. *ACM Transactions on Mathetmatical Software 23*, 4, 550–560.