Abstract of "Nonparametric Bayesian Models for Neural Data" by Frank Wood, Ph.D., Brown University, May 2007.

Many neural data analyses can be cast as latent variable modeling problems. Specific examples include spike sorting and neurological data analysis. Challenges in spike sorting include figuring out how many neurons generated a set of recorded action potentials and, further, which neuron generated each action potential. A challenge in analyzing neurological data is to infer both the number and the characteristics of lesions that may be causal with respect to clinical signs presented by stroke patients. A shared characteristic of both of these problems is that the true underlying generative process is unobservable and potentially quite complex, so care must be taken in not only choosing a family of models but also in selecting a model of appropriate complexity. In such cases it may be preferable to employ a model that allows model complexity to be inferred from the data.

Non-parametric Bayesian (NPB) modeling is a type of latent variable modeling in which model complexity can be estimated from data without making restrictive a priori assumptions. Our thesis is that using NPB modeling results in theoretical and practical improvements to neural data analysis.

In defense of this thesis we develop a NPB spike sorting approach and show how it allows experimentalists to utilize more data, to make assumptions explicit, and to express spike sorting uncertainty at the level of inference from a novel spike train model. We discuss the theoretical advantages of this approach and demonstrate novel and improved neural data analyses including neural decoding. We also develop a new NPB binary matrix factorization model and accompanying posterior estimation algorithms. We illustrate this NPB binary matrix factorization model by inferring a causal model for signs exhibited by stroke patients. Finally, a sequential posterior estimation algorithm for this model is developed and demonstrated.

Nonparametric Bayesian Models for Neural Data

by

Frank Wood

B. S. Computer Science, Cornell University, Ithaca, NY, USA, 1996

M. Sc. Computer Science, Brown University, Providence RI, 2004

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island

May 2007

This dissertation by Frank Wood is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____ _____

Michael J. Black, Director

Recommended to the Graduate Council

Date _____ _____

Thomas L. Griffiths (University of California at Berkeley), Reader

Date _____ _____

John F. Hughes, Reader

Date _____ _____

Zoubin Ghahramani (Cambridge University), Reader

Approved by the Graduate Council

Date _____ _____

Sheila Bonde
Dean of the Graduate School

Acknowledgements

I would like to thank Karen Fuerherm for her love and loyal support and my family Vernon, Carol and Drew Wood for helping me to pursue my dreams.

I owe a debt of gratitude to Stefan Roth who helped me become mathematically literate again after a long business-world hiatus. Without Matthew Fellows and Wilson Truccolo this dissertation would have a different topic; thank you. In this respect I also owe thanks to John Donoghue, Stewart Geman, Elie Bienenstock, and Mayank Mehta. Thank you.

In addition to my advisor Michael Black and the committee members listed on the previous page (particularly Tom Griffiths), I also owe thanks to my collaborators Prabhat, Phil Kim, and Sharon Goldwater.

My life at Brown has suffered since Joseph Laviola graduated; I thank him for his tireless dedication to play. To my officemates Daniel Acevedo, Yanif Ahmad, and Russell Bent; thanks for letting me win most of the bets. I thank Anne Booker for helping to keep me sane.

Last but not least I would like to thank Grace, Mark, and Noelle Wood who made Rhode Island a home rather than merely the place I lived during graduate school.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Evidence is accumulating in many fields that neural computation is Bayesian in nature [Koerding and Wolpert, 2004; Weiss et al., 2002]. Conversely, Bayesian modeling may be the most promising tool outside of the wet laboratory for investigating the algorithmic basis of neural computation. If we accept these propositions then it would seem that exploring Bayesian modeling will help in development of a theory of neural computation. Additionally, recognizing that neural computation is superior in some domains (for instance object recognition, learning from a single example, source separation, etc.) makes it seem likely that studying the mechanisms of neural computation will yield important insights into improved Bayesian modeling. The synergy between neuroscientific investigation and progress in Bayesian modeling is a primary motivation for this dissertation.

The focus of this dissertation is on improved models for neural data analysis. Models play an important role in neural data analysis as quite often the true generative process underlying neural data is unknown or can't be observed directly. As such characteristics of the generative process of interest must be inferred using models constructed from related observations. A simple analysis that makes this clear is determining how many neurons are captured in any single electrophysiological recording. Such a recording is typically made using an electrode placed in neural tissue and usually consists of an amalgam of activity from many neurons, the number of which usually is neither known nor directly observable. In this example a model based approach must be used to infer how many neurons were recorded.

Unfortunately in this case and many others like it, because the true generative process is not observable there is uncertainty about what the "best" model is to use. One approach to coping with this uncertainty is to pick a model or family of models and a definition of

"best" and then to search for the best model. If it is certain that the definition of best, the model family, and the search procedure always produces a model that is closer to the true generative process, then confidence in outcomes from analyses based on the best model should be high. Unfortunately it is not often the case that the model and search procedure are guaranteed to produce a model closer to the true generative process. Because of this it often pays to take an agnostic approach to modeling.

This can be done by representing uncertainty in and about the models themselves. Two major sources of modeling uncertainty are determining the best parameters for the model and, often closely related, picking the best model complexity. Bayesian modeling approaches allow one to account for uncertainty about model parameters. Nonparametric Bayesian modeling (NPB) allow one to avoid uncertainty in choosing model complexity by subsuming estimation of it into the model itself. How this works is not meant to be clear at this point but it will be made clear later, particularly by reviewing one particular NPB model in Chapter 2. The important point here is that neural data analyses, or for that matter any analyses that are based on modeling approaches that model uncertainty and avoid a priori assumptions about model complexity will benefit in theoretical and practical terms. The benefits taking such an approach in analyses of neural data are most clearly demonstrated in Chapter 3 where a nonparametric Bayesian model of spike trains is developed and used in various example neural data analyses.

This work on nonparametric Bayesian modeling does not stand alone. To the contrary, nonparametric Bayesian models have recently enjoyed a surge in popularity in the machine learning community [Neal, 2000; Rasmussen, 2000; Griffiths and Ghahramani, 2005; Wood et al., 2006b; Wood and Griffiths, 2007] for many of the reasons stated. The NPB models in those publications and in this dissertation are all descendents of Dirichlet process models that have been known in the statistics community for some time [Antoniak, 1974; Ferguson, 1983]. Nonparametric Bayesian models have been successfully applied in many domains other than neural data analysis including studies of high level cognitive processing [Sanborn et al., 2006], object recognition [Sudderth et al., 2005], and hierarchical modeling [Teh et al., 2004]. To our knowledge we were the first to apply nonparametric Bayesian modeling in support of neural data analyses [Wood et al., 2006b].

Our general thesis is that adopting nonparametric Bayesian modeling will yield theoretical and practical improvements to analyses of neural data. In this introduction we clarify this assertion, starting here with a more detailed but still high level explanation of nonparametric Bayesian modeling then conclude with a preview of the models, algorithms, and applications developed in this dissertation.

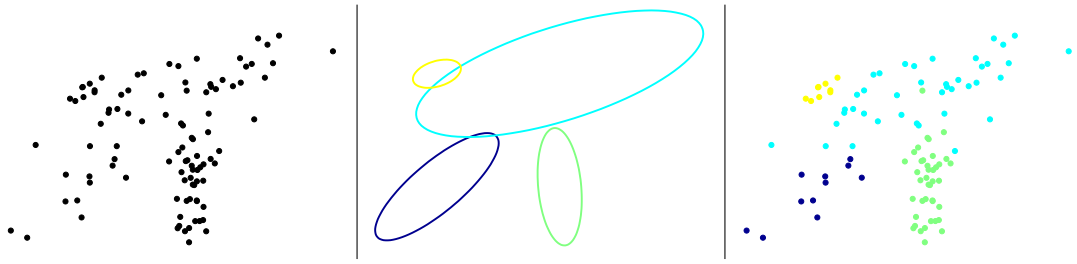

Figure 1.1: From left to right: random data, true generative mixture model, true labeling

1.1 Background

To start let's pretend that we have an analysis we would like to perform on data $\mathcal{Y} \in \mathbb{R}^2$ which we have reason to believe was generated by several distinct processes that generate groups of "similar" datapoints. An example of such data is shown on the left in Figure 1.1. In order to perform this hypothetical analysis we would be well served to utilize a model of the data that reflects this character of the true underlying generative process. A standard way to do this is to build a latent variable model, usually a mixture model, in which a correspondence is sought between mixture densities in the model and the distinct processes that generated the data. A more detailed explanation of mixture modeling is given in the next chapter. For now it is sufficient to understand that fitting a mixture model to the data effectively establishes this correspondence, as each point is attributed to one of the mixture densities in the mixture model.

Unfortunately determining how many distinct processes generated such data is often hard to do. For example look at the left panel of Figure 1.1 again. In this figure it is not immediately clear what the true generative process is (i.e. how many clusters are in the data). This is reinforced by Figure 1.2 in which constant variance isosurfaces are used to represent Gaussian mixture models with one to six component densities fitted to the data shown in Figure 1.1. Without knowing the ground truth is not immediately clear which model is the best in terms of correspondence to the true generative process behind the data (the true generative process is show in the right two panels of Figure 1.1).

In fact if one were to search for the mixture model that best represents the data using the likelihood of the data under the model one would find that increasing the number of mixture components will always produce a better model (up to the number of data points N at least). This is because one can always increase the likelihood of the data by placing each point in its own cluster. This highlights one of the sources of modeling uncertainty that taking a NPB modeling approach addresses; namely selecting the "complexity" of the

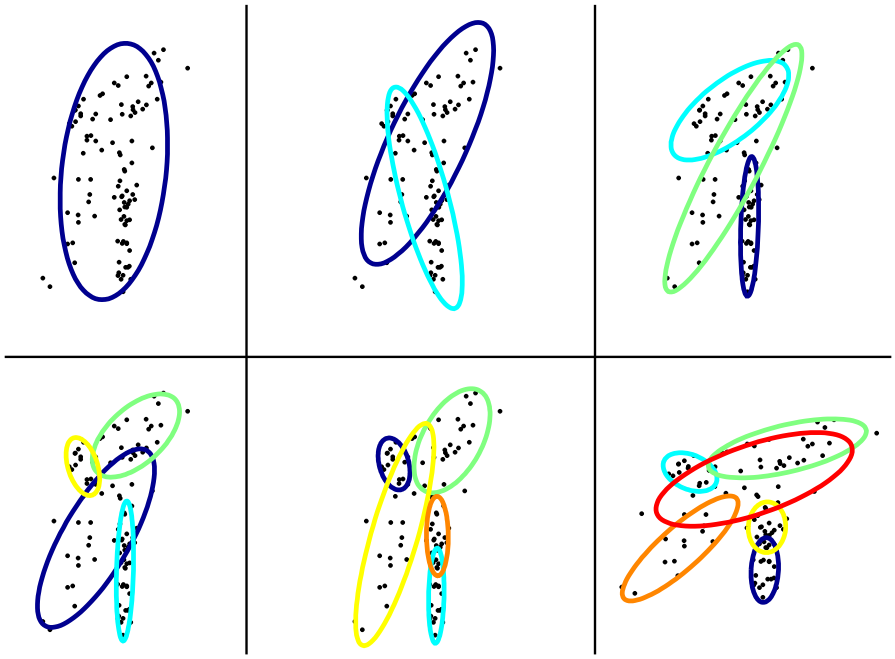

Figure 1.2: Candidate models for the data plotted in Figure 1.1

model. The effect on analysis outcomes from inference in models that vary in complexity may be significant. Too complex a model may result in overfitting and lack of generalization (i.e. potentially incorrect analysis outcomes), too simple a model may result in the converse (but still potentially incorrect analysis outcomes). Approaches for negotiating this tradeoff are instances of the general technique called model regularization [Duda et al., 2000].

A problem with traditional approaches to regularization is that there are many different ways to regularize a model, and often the "best" models selected under different regularization schemes will themselves be different. Picking the best model is thus complicated by needing to select between a number of best models (corresponding to each of the regularization schemes). One way of dealing with this is to be confident in one's modeling choices and to conclude that the problem of selecting the "best" model is solved given a good regularization approach and a sufficient amount of data. The Bayesian approach we advocate is different.

In the Bayesian framework regularization is subsumed into the model specification. To see this, let Θ be the set of all model parameters and $\mathcal{Y}$ be the training data. Let our model be a density function called the likelihood $P(\mathcal{Y}|\Theta)$ with parameters Θ. Bayesian models specify an a priori distribution of the parameters $P(\Theta)$ called the prior. The prior is how regularization is introduced in the Bayesian context. A prior that penalizes models with

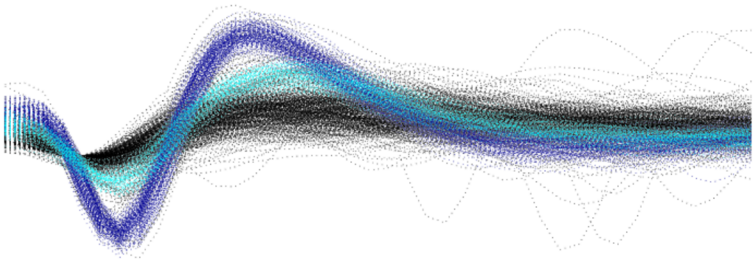

Figure 1.3: Hand sorted action potential waveforms.

high complexity can help in generalization by changing the model selection criteria from "find the best model" to "find the best model that also isn't overly complex".

The Bayesian perspective also dictates that we eschew our search for the best model in favor of learning a distribution over models. Bayes' rule tells us how to construct this distribution by combining the likelihood with the prior

$$P(\Theta|\mathcal{Y}) \propto P(\mathcal{Y}|\Theta)P(\Theta).$$

The distribution over model parameters $P(\Theta|\mathcal{Y})$ is called the posterior distribution. Analyses can be averaged over such a posterior distribution, accounting for modeling uncertainty while avoiding having to choose some "best" model.

Thus being Bayesian means that we no longer attempt to find a single "best" model (although we can, and this is called the maximum a posteriori (MAP) model and takes into account the influence of the prior) and that analyses will be averaged over a posterior distribution over models.

Unfortunately in traditional mixture modeling for example, the number of mixture densities K is not modeled as a random quantity

$$P(\Theta|\mathcal{Y}; K) \propto P(\mathcal{Y}|\Theta; K)P(\Theta; K)$$

with the obvious ramification that the problem of needing to specify or search for a "best" model is re-introduced. In mixture model clustering K is the number of clusters. Even in an otherwise Bayesian modeling context its value is often selected a priori or picked via some other optimization strategy, for instance cross-validation. There is a Bayesian way of learning a distribution over K, notably the reversible jump Markov chain Monte Carlo work of Green [1995], but in our opinion this approach has some undesireable shortcomings. Some evidence in support of this opinion is given in Chapter 5.

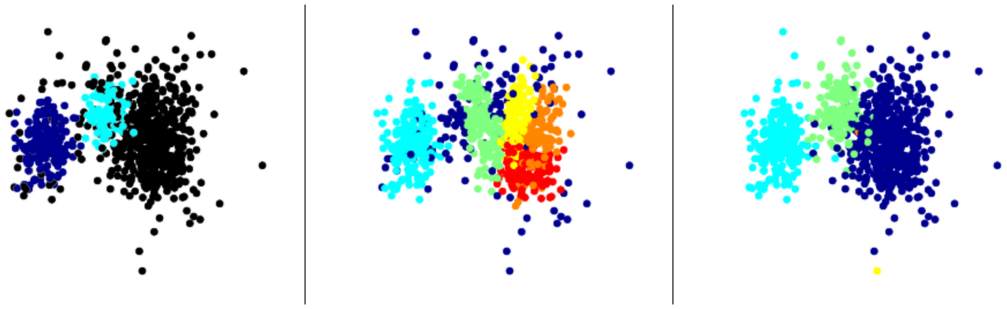

Figure 1.4: From left to right: hand labeled PCA coefficients for the waveforms plotted in Figure 1.3, maximum likelihood Gaussian mixture model labeling, and infinite Gaussian mixture model labeling

In this dissertation we advocate a nonparametric Bayesian approach to this problem where model complexity estimation is subsumed into the model. NPB models are typified by having unbounded prior complexity, which in the mixture model example corresponds letting $K \to \infty$. This choice of an infinite number of mixture densities would apparently complicate estimation and inference due to the need to compute with models having an infinite number of parameters. It also apparently violates the spirit of being Bayesian to have picked a "value" for K (∞). However in practice such models are designed to remain sparse in the sense that posterior distribution will admit only models with fewer mixture densities than the number of observations and often fewer than that depending on the model definition and parameterization. This approach admits interesting philosophical interpretations but is most easily appreciated as an elegant and computationally efficient way of avoiding a priori selection of model complexity. The consequence of taking this approach is that analyses cast as posterior inference in NPB models take into account uncertainties arising from both model parameter estimation and from model complexity selection.

1.2 Preview of results

It remains to expand this explanation of NPB modeling and to defend our claim that adopting such a nonparametric Bayesian approach yields theoretical and practical improvements to the analysis of neural data. We begin with an in-depth review of infinite Gaussian mixture modeling (IGMM) in Chapter 2, the IGMM being a nonparametric Bayesian model.

In Chapter 3 we introduce a novel use of the IGMM for neural data analysis based on spike trains [Wood et al., 2006a, see]. Neural data analyses based on spike trains are varied in nature and constitute a significant fraction of all neural data analyses. The process of

generating spike trains from electrophysiological recordings is called spike sorting. A more precise definition of spike sorting is given in Chapter 3; briefly however, spike sorting is the process of clustering a set of putative action potentials ("spikes") in order to uncover how many neurons generated them and which neuron generated which action potential. Mixture modeling is one way to do this clustering.

Here we preview some of the results from Chapter 3. Figure 1.3 shows hand-labeled spikes recorded from a single electrode. In this figure colors indicate the labels given to each action potential waveform by an expert human sorter, thus the red and green waveforms are labeled as having arisen from two distinct neurons. The black waveforms were determined by the expert sorter to be too ambiguous to retain in subsequent neural data analyses. That spike sorting may be cast as a clustering problem is clear from Figure 1.4 where there is one dot for each of the waveforms in Figure 1.3. To produce Figure 1.4, principal component analysis (PCA) [Jolliffe, 1986] was used to build a reduced dimensionality representation of the waveforms in Figure 1.3. In general we will show the PCA representation when presenting spike sorting results; however, it should be kept in mind that we are sorting spikes based on differences between waveform shapes. Different waveform shapes are caused by heterogeneous cell types and for cells of the same type from variation in the distance to the electrode tip.

Neuroscientists study many aspects of the brain and at many different levels of organization; however, many of the questions they ask can be cast as operations on spike trains. Unfortunately it is known that it is almost unavoidable that uncertainties will arise in spike sorting, and furthermore that spike train variability induced by different sortings can and will affect analysis results. [Harris et al., 2000; Wood et al., 2004a]. Sometimes experimental design and avoidance of ambiguous data can minimize the level of uncertainty in spike trains; however, in other cases experimental design might be adversely constrained. For instance chronically implanted electrodes do not allow the experimenter precise control over electrode placement nor do they admit easy adjustment of the electrodes if data recorded from them is ambiguous. In such cases one may still wish to make the greatest use of the available data. Doing so in a principled way requires a modeling approach that retains an estimate of the variability manifest in spike trains arising from uncertainty in spike sorting.

Our novel NPB spike sorting approach does this. As a prelude to expanded results presented in Chapter 3, consider Figures 1.4 which shows a finite mixture model fitted to the data shown in Figure 1.4 (we follow the literature in sorting in PCA space). This model was fit using a standard method for finding a "best" Gaussian mixture model called expectation maximization (EM) [Dempster et al., 1977]. In this case EM was initialized

using spectral clustering [Ng et al., 2001] and the model included no other regularization. Expectation maximization is the maximum likelihood parameter estimation procedure for models with missing or hidden variables. Ten restarts of expectation maximization were considered and the best model was retained. Further, the number of densities (model complexity) was selected according to the Bayesian information criteria (BIC) [Duda et al., 2000]. Here the best model according to the BIC had six Gaussian densities (models from with one to fifteen densities were considered). Figure 1.4 shows a single model from the NPB posterior distribution over models. As far as we know no nonparametric Bayesian approach has been applied to the spike sorting problem prior to our work.

These figures illustrate one of the practical benefits of NPB modeling. As a consequence of utilizing an explicit generative model the partitioning of the data looks quite reasonable as compared to the human labeling in Figure 1.4. What isn't captured by these figures is the theoretical benefit of the NPB spike sorting approach. While the maximum likelihood approach (EM) requires us to select a single "best" model, in the NPB case we construct a posterior distribution over models that represents our uncertainty about, for instance, the number of clusters in the data and the assignment of datapoints to clusters. Neural data analyses that are cast as operations on such a posterior distribution will be shown to benefit from theoretical and practical improvements.

It also should be noted that the amount of data involved in a typical neurophysiological recording is quite large (Gbs) and that current state of the art automatic clustering algorithms are offline batch processing algorithms. This means that the data can only be analyzed after it has been recorded. Current spike sorting algorithms also require very large amounts of computer time and a high degree of manual intervention [KlustaKwik]. An appealing consequence of our choice of NPB modeling is that it is possible to construct the model sequentially [Fearnhead, 2004; Wood and Griffiths, 2007]. This means that the posterior distribution can be estimated in a single online pass through the data, perhaps even at the time of recording. This is an enormous improvement over prior art because it means that spike sorting can be done sequentially in a single pass through the data, potentially at the same time as it is being recorded.

In further defense of our thesis we also develop a matrix factorization model, posterior estimation algorithms, and novel applications of these models that demonstrate their probable utility for neural data analysis as well. Figure 1.6 shows a subgraph of the maximum a posteriori causal structure for a databank of clinical stroke findings for 50 patients. These results come from an artificial modeling problem constructed by throwing away our knowledge about types of strokes (particular lesion locations) and which signs are expressed by

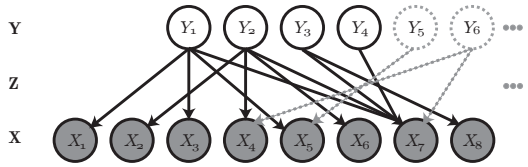

Figure 1.5: A hypothetical Bayesian network connecting hidden causes $Y_1, \ldots, Y_K$ to observed variables $X_1, \ldots, X_N$. We consider the case where the number of hidden causes, K, is unbounded. The state of the hidden causes, the observed variables, and the dependencies between them can all be summarized using binary matrices, being $\mathbf{Y}$, $\mathbf{X}$, and $\mathbf{Z}$ respectively.

patients having particular types of strokes. Given only a database of symptoms (in the true neurologist vernacular these would be called signs) expressed by 50 patients we infered a causal model of stroke symptoms. Our model consisted of a set of latent causes and the connectivity between those latent causes and the symptoms expressed. For instance, a left temporal lesion might cause language and cognitive impairments in addition to, potentially, right side facial weakness. Right side facial weakness on the other hand could be caused by a number of different lesions. So in this causal model, unlike the spike sorting model, the relation between cause and observations is many-to-many rather than one-to-one.

While the details of the model, estimation, and inference algorithms are left to Chapters 4 and 5, the model we chose was a "Noisy-Or" model in which the probability of observing a particular symptom (binary) is related to the number of diseases the patient has that are causally linked to that symptom. Such a model allows us to cast the causal structure learning problem as a binary matrix factorization model if we represent the causal connectivity of a bipartite graph from causes to observations, illustrated in Figure 1.5, by a binary adjacency matrix where there is a one if there is a link from a hidden variable to an observed variable and a zero if not.

Leaving the details to a later chapter, if $\mathbf{X}$ is a matrix of stroke symptom observations we can model it using probabilistic matrix factorization model

$$\mathbf{X} \sim \text{Noisy-Or}(\mathbf{ZY})$$

where, as noted, $\mathbf{Z}$ is a binary matrix that represents the adjacency structure in a bipartite graph between causes and observations and $\mathbf{Y}$ is a matrix of latent causes. The shared dimension of $\mathbf{Z}$ and $\mathbf{Y}$ stipulates the number of hidden causes and thereby constitutes model complexity. In the same way as in the spike sorting case we can do nonparametric Bayesian inference in a model of causal structures. Although here too the objective is to learn a distribution over causal structures, Figure 1.6 shows the MAP sample from a model of the Mt.

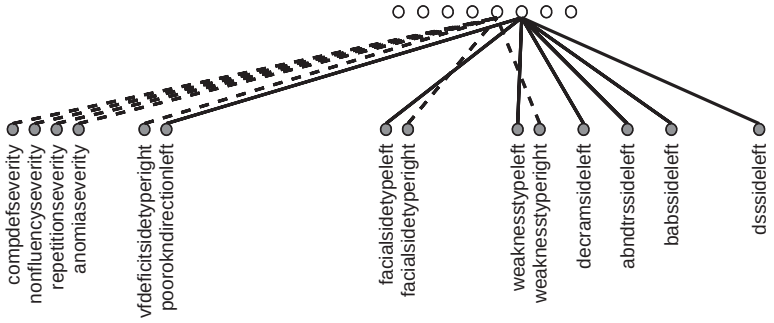

Figure 1.6: Maximum a posteriori causal structure subgraph for stroke findings

Sinai stroke databank [Tuhrim et al., 1991]. While the same argument for the theoretical superiority of the nonparametric Bayesian approach holds in this problem domain, this specific figure illustrates the utility of our novel model and estimation technique. To generate this figure we took our database of 50 patients and the binary stroke findings (symptoms) they exhibited and generated a distribution over causal models, each postulating stroke localizations (lesions locations) that would well explain the findings in the databank. What we show is a subset of the MAP causal structure and highlight two of the localizations this sample indicates. In dashed black we find a grouping of comprehension deficit, nonfluency, repetition, anomia, visual field deficit, facial weakness, and general weakness with the latter three on the right side, which is generally consistent, in part, with a left temporal infarct. In solid black we find a grouping of poor optokinetic nystagmus, lack of facial control, weakness, decreased rapid alternating movements, abnormal deep tendon reflexes, Babinski sign, and double simultaneous stimulation neglect, all on the left side, which is consistent with a right frontal/parietal infarct. These interpretations were developed by an expert clinical neurologist who further claimed that the other features (numerous) of the MAP causal structure were generally consistent and reasonable given the modest patient database available.

While it is comforting that this model performs well on a simple, somewhat artificial problem like that described, unlike the previous spike sorting example the main contribution here is not an analysis framework for stroke patient data. Instead the novel NPB binary causal structure model is itself the contribution. Many neural data analyses ranging from studies of neural connectivity to models of population spiking activity could benefit from such a model. The stroke data analysis merely illustrates the power of the model and provides a guide for its utilization in subsequent neural data analyses.

1.3 Summary

What is important to understand at this point is our thesis regarding the benefits to neural data analyses of utilizing nonparametric Bayesian modeling. We claim that nonparametric Bayesian modeling should allow more data to be utilized, particularly if discarding ambiguous data is a default practice. Furthermore as model complexity estimation is subsumed into the modeling framework nonparametric Bayesian modeling may avoid bias in analysis outcomes resulting from selection of a single best model that happens to be either too complex or not complex enough. Finally the NPB models we discuss can all be estimated sequentially. This means that model estimation can proceed in lockstep with the accumulation of observations. For many neural data analyses this is a desirable characteristic because of the nature and volume of data to be analyzed.

Chapter 2 starts with a review of nonparametric Bayesian mixture modeling.

Chapter 2

Nonparametric Bayesian Mixture Modeling

There are a wide variety of problems for which latent variable modeling is appropriate. A latent variable model utilizes "hidden" random variables to represent the state of processes that are not directly observable. The reasons for constructing such a model vary. In some cases the specific values of the postulated latent variables are of interest; in others the latent variable model is merely a sophisticated mechanism for model regularization. Additionally, in many cases it is difficult to determine the appropriate complexity for the latent variable model. Specifying too complex a model can result in a model that is too powerful and overfits the data. Specifying too simple a model can result in inappropriately smooth representations of the data. Ultimately avoiding both problems requires negotiating an often delicate balance between model specificity and generality.

In a maximum-likelihood modeling various mechanisms can be employed to negotiate this tradeoff; cross validation and model selection by penalized likelihood are two of the most common approaches. In cross validation some percentage of the data are withheld when constructing the model. How well the model fits the held out section is measured and this process is repeated. Cross validation achieves complexity control by selecting the model that performs best on average. This is a somewhat heuristic approach as the number of folds in the cross valuation must be established and the dependence of this decision on other model parameters is not accounted for. Approaches based on penalizing the likelihood based on information theoretic criteria such as the minimum description length, reviewed in Grünwald [2007], or the Bayesian information criteria introduced by Schwarz [1978] employ a similar heuristic approach. Multiple models of varying complexity are trained and for

each an information theoretic quantity is computed as a function of the model complexity. The model with the best combined score (highest likelihood and lowest complexity penalty) is judged to be the model with ideal complexity.

Various Bayesian approaches to model complexity exist; perhaps the most general of which is the reversible jump Markov chain Monte Carlo work of Green [1995]. In this work the standard Metropolis Hastings sampling of Metropolis et al. [1953] and later Hastings [1970] is extended to allow for transitions in model rather than parameter space. In this sense a posterior distribution over model complexity can be estimated from the data. Usually such transitions are considered only between models in the same parametric family where, for instance, the number of parameters is the measure of model complexity. A particular example of this is given in Richardson and Green [1997] for Gaussian mixture models.

This chapter serves as an introduction to nonparametric Bayesian (NPB) modeling, focusing in particular the practical aspects of mixture modeling. We introduce the nonparametric Bayesian modeling approach by focusing on one particular NPB model, the infinite Gaussian mixture model (IGMM) of Rasmussen [2000]. There are many kinds of nonparametric Bayesian models; however, we use the IGMM as an example because of the familiarity of Gaussian mixture modeling and also because in Chapter 3 we use an IGMM to approach the neuroscience problem of spike sorting in a different way.

The general Bayesian nonparametric methodology is to specify a model with unbounded a priori complexity, but to restrict it in such a way that models of finite realized complexity are manifest a posteriori given a finite amount of data. In doing so the model automatically captures and expresses the inherent complexity of the data. The IGMM is a specific case of a Dirichlet process mixture model introduced in part by Antoniak [1974] with per-class normally distributed observations. Here model complexity is indicated by the number of latent classes. As the IGMM has an infinite number of latent classes with each being a normal distribution it is a mixture model with infinite complexity,

In Section 2.1 we introduce notation and review finite Gaussian mixture modeling. In Section 2.2 we review the IGMM and derive the conditional distributions necessary for Gibbs sampling. We review two kinds of Gibbs sampling posterior estimation in Sections 2.2.1 and 2.2.2, expanded and collapsed Gibbs sampling respectively. In Section 2.2.3 we introduce a sequential posterior estimation algorithm for the IGMM. We conclude in Section 2.3 with a demonstration of IGMM modeling using synthetic data.

2.1 Finite Gaussian Mixture Modeling

A finite Gaussian mixture model (GMM) is a latent variable model formed by adding together weighted Gaussians. Finite GMM's are frequently used in both statistics and computer science. This is because they are quite flexible and can be used to both approximate arbitrary probabilitiy densities, particularly those exhibiting multi-modality, and to cluster data if the latent variables are treated as class labels.

The generative model for a multivariate finite GMM is

$$\begin{aligned} c_i|\vec{\pi} &\sim \text{Multinomial}(\vec{\pi}) \\ \vec{y_i}|c_i = k, \Theta &\sim \text{Gaussian}(\cdot|\theta_k) \end{aligned} \tag{2.1}$$

where $\mathcal{C} = \{c_i\}_{i=1}^N$ are class indicator variables. Here $\Theta = \{\theta_k\}_{k=1}^K, \theta_k = \{\vec{\mu}_k, \mathbf{\Sigma}_k\}$ are the mean and covariance for class k, and $\vec{\pi} = \{\pi_k\}_{k=1}^K, \pi_k = P(c_i = k)$, the class prior probabilities, are parameters of the model. The notation $x|y; z \sim f(x, y; z)$ means that the random variable x is conditionally distributed given the random quantity y and parameter z according to the probability distribution function f. Here by "Gaussian" we mean the multivariate normal (MVN) density function.

The latent class indicator variables $\mathcal{C}$ serve to indicate which class generated each datapoint (i.e. $c_i = k$ means that the i^{th} datapoint came from k^{th} latent class). Latent variables, also called hidden variables, are so named because they cannot be directly observed. If learned, however, they are often semantically useful depending on the application domain. For instance using them in a mixture modeling context can result in powerful automatic summarization of data. For this reason it is often the case that in mixture modeling the de facto goal is to learn the state of these latent variables.

Before getting into model estimation, understanding the generative view of this model helps to build intuitions about how it works. To generate data from this model first the number of latent classes K must be chosen; also the relative proportions of data arising from each class, $\vec{\pi}$ (i.e. π_k is the expected percentage of the data that will arise from class k). The number of observations N must also be chosen; as too the parameters for each latent class θ_k. Once these choices are made then N class labels c_i may be generated by sampling from a multinomial parameterized by $\vec{\pi}$ (i.e. pick $c_i = j$ with probability π_j). Finally N observations may be generated from the corresponding normal distributions with parameters $\vec{\mu}_{c_i}, \mathbf{\Sigma}_{c_i}$.

To model data using a GMM one must turn around this generative process and instead estimate the model parameters from the data (i.e., the latent variables $\pi, \mathcal{C}, \Theta$). Expectation Maximization (EM), introduced by Dempster et al. [1977], is the preferred method for estimating model parameters in situations where observations are missing or, equivalently, the model contains latent variables. Expectation maximization is a maximum-likelihood (ML) estimation technique consisting of an iterative procedure for finding a single setting of the parameters and missing variables that maximizes the log likelihood of the data given the model

$$\hat{\vec{\pi}}, \hat{\Theta} = \arg\max_{\vec{\pi},\Theta} \log(P(\mathcal{Y}|\vec{\pi}, \Theta)). \tag{2.2}$$

where $\mathcal{Y}$ is the set of all observations $\vec{y_i}$ and the likelihood is arrived at by marginalizing out the latent class indicator variables

$$P(\mathcal{Y}|\vec{\pi}, \Theta) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k P(\vec{y_i}|c_i = k, \Theta). \tag{2.3}$$

EM is a powerful algorithm that was shown by Neal and Hinton [1998] to always convergence to a local maximum of the likelihood. Unfortunately if the likelihood has multiple maxima (as may be the case), this guarantee is not sufficient to ensure that the global maximum will be found. For this reason the initialization of the EM algorithm can have marked effects on the results. Another problem not specific to EM but of mixture modeling in general is that the likelihood can always be increased by adding more mixture components. This can be problematic because in its basic form EM requires us to know the number of latent classes K a priori. Although EM variants that attempt to learn K exist they are not not straightforward.

Here we hit for the first time on a central theme of this work: the true number of latent classes K is usually not known a priori, cannot be directly observed, and therefore must be estimated from the data. There are several ways to estimate K in the maximum-likelihood setting, among them cross validation [Duda et al., 2000] and regularization through information theoretic model complexity penalties such as the Bayesian information criterion. The Bayesian information criterion is given by BIC $= -2\log(P(\mathcal{Y}, \mathcal{C}|\vec{\pi}, \Theta)) + \nu_K \log(N)$, where ν_K is the degrees of freedom of a model indexed by K (here the number of hidden densities) [Schwarz, 1978]. This can be interpreted as roughly penalizing models according to their complexity or roughly the log of the number of free parameters. Note that in the mixture modeling context this penalty puts a brake on improving the likelihood score by

adding more mixture components. At some point the cost of encoding the model's extra parameters will outweigh adding more components. By iteratively searching for this threshold one may choose the number of mixture densities.

Unfortunately such maximum-likelihood approaches all do something that one might like to avoid: They seek a single "best" model of the data. In some circumstances this is a reasonable thing to do, for instance, if a single state of the latent variables is actually the quantity of interest then finding the best setting of them is a reasonable thing to do. However if the ultimate goal is to infer something about the existing data or observations yet to come, it may be advantageous to consider multiple models of the data rather than a single best model. One way to do this is to estimate a distribution over models. This way inferences one might like to draw from the data can be cast as operations on such a distribution. Averaging (one such operation) can help avoid problems that arise from mistakenly choosing a suboptimal model (i.e., picking a best model that isn't actually the best) when doing ML modeling.

This is the Bayesian approach and taking it will allow us to work around model selection among other problems. Taking the Bayesian approach means that the model parameters will themselves be treated as random variables. This means that we should specify generative models for the parameters as well which are called "priors." Then, instead of estimating a single best set of parameters, our goal will be to estimate distribution over parameters called the posterior distribution. As a reminder, Bayes' rule tells us that the posterior probability of a set of parameters $\mathcal{M}$ given observations $\mathcal{Y}$ is proportional to the likelihood $P(\mathcal{Y}|\mathcal{M})$ times the prior probability of the parameters $\mathcal{M}$

$$P(\mathcal{M}|\mathcal{Y}) \propto P(\mathcal{Y}|\mathcal{M})P(\mathcal{M}). \tag{2.4}$$

If we view EM as a Bayesian procedure we see that it provides a point estimate of the posterior distribution assuming that the prior probability of all models is equal. Of course this is not our goal, our goal is instead to estimate the entire posterior distribution.

To estimate the posterior distribution we first have to specify priors for the model parameters. For both purposes of illustration and use in subsequent chapters we follow Fraley and Raftery [2005] in choosing priors of the following types for the model parameters: Dirichlet for $\vec{\pi}$ and Gaussian times Inverse Wishart for θ [Gelman et al., 1995]. These priors are chosen for mathematical convenience and interpretable expressiveness. They are conjugate priors [1] which will allow us to analytically perform many of the integrations

[1] A prior is conjugate if it yields a posterior that is in the same family as the prior [Gelman et al., 1995].

necessary in estimating the posterior distribution for this model. This choice of priors is denoted

$$\begin{aligned} \vec{\pi}|\alpha &\sim \text{Dirichlet}(\cdot|\frac{\alpha}{K},\ldots,\frac{\alpha}{K}) \\ \Theta &\sim \mathcal{G} \end{aligned} \tag{2.5}$$

where $\Theta \sim \mathcal{G}$ is shorthand for

$$\boldsymbol{\Sigma}_k \sim \text{Inverse-Wishart}_{\upsilon_0}(\boldsymbol{\Lambda}_0^{-1}) \tag{2.6}$$

$$\vec{\mu}_k \sim \text{Gaussian}(\vec{\mu}_0, \boldsymbol{\Sigma}_k/\kappa_0). \tag{2.7}$$

The formulas for these distributions are readily available on the web and in print, for instance in Gelman et al. [1995]. Here we focus on the modeling interpretation of these choices. The Dirichlet prior is used to encode prior knowledge about the number and relative prevalence of the hidden classes. Here we choose a uniform parameterization $\frac{\alpha}{K}$ for mathematical convenience when generalizing to the nonparametric model; however, if K is somehow known a priori then a non-uniform parameterization can be chosen accordingly. For now we will treat α as a hyperparameter to be user specified but it too can be treated as a parameter of the model for which a distribution is to be estimated. The parameters of the Gaussian times Inverse-Wishart prior, $\mathcal{H} = \{\boldsymbol{\Lambda}_0^{-1}, \upsilon_0, \vec{\mu}_0, \kappa_0\}$, are used to encode our prior beliefs regarding the shape and position of the mixture densities. For instance $\vec{\mu}_0$ specifies where we believe the mean of the mixture densities should be, where κ_0 is the number of pseudo-observations we are willing to ascribe to our belief. The hyper-parameters $\boldsymbol{\Lambda}_0^{-1}$ and υ_0 behave similarly for the mixture density covariance. We will refer to $\mathcal{H}$ as the hyperparameters of our model. In general these will be specified by a user and not estimated from the data. Note that this specification of the priors assumes that we know the value of K a priori. At this point we are focusing on building the Bayesian Gaussian mixture model; shortly we will show how to avoid this assumption. Also these are not the only choices that one could make for the priors. In particular other choices for the multivariate normal parameter priors may be more appropriate for certain applications.

A graphical model representing this model is shown in the middle of Figure 2.1. This graphical model illustrates the conditional dependency structure of the model. For instance π_k is independent of $\mathbf{y}_i$ given c_i. In this figure circles indicate random variables and plates indicate repetition. Links between variables indicate that they appear together in terms in

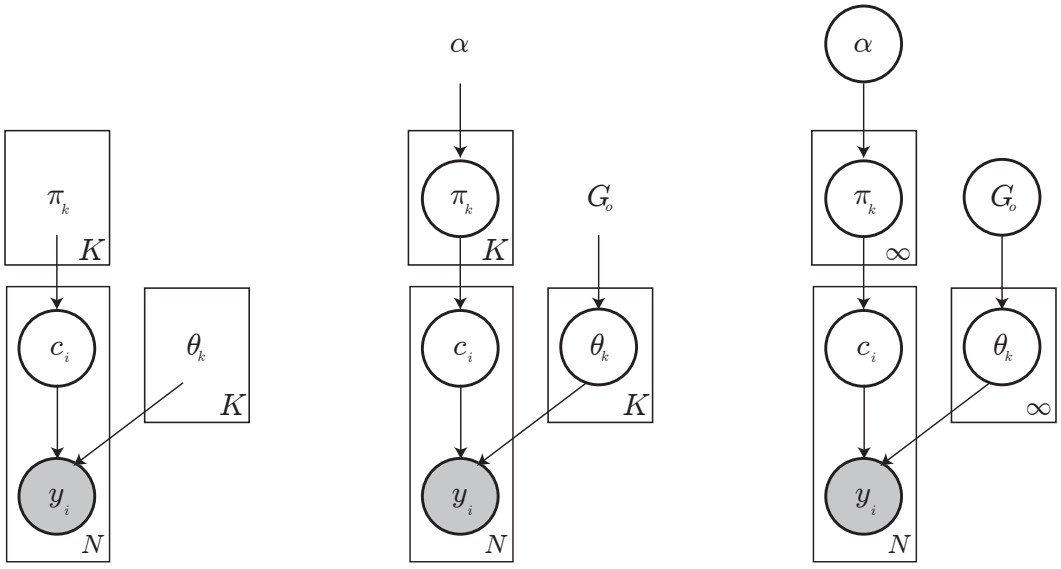

Figure 2.1: From left to right: graphical models for a finite Gaussian mixture model (GMM), a Bayesian GMM, and an infinite GMM

the joint probability distribution. From such a graphical model the joint distribution of the data and model parameters can easily be read out. Here that joint distribution is is given by

$$P(\mathcal{Y}, \Theta, \mathcal{C}, \vec{\pi}, \alpha; \mathcal{H}) = \left(\prod_{j=1}^{K} P(\theta_j; \mathcal{H}) \right) \left(\prod_{i=1}^{N} P(\vec{y}_i | c_i, \theta_{c_i}) P(c_i | \vec{\pi}) \right) P(\vec{\pi}|\alpha) P(\alpha). \tag{2.8}$$

Given this joint distribution it is simple to derive the posterior distribution by algebraic manipulation. The posterior distribution of the Bayesian Gaussian mixture model is

$$\begin{aligned} & P(\mathcal{C}, \Theta, \vec{\pi}, \alpha | \mathcal{Y}; \mathcal{H}) \\ & \propto \quad P(\mathcal{Y}|\mathcal{C}, \Theta) P(\Theta; \mathcal{H}) \left(\prod_{i=1}^{N} P(c_i|\vec{\pi}) \right) P(\vec{\pi}\,|\alpha) P(\alpha). \end{aligned} \tag{2.9}$$

where $P(\mathcal{Y}|\mathcal{C}, \Theta) = \prod_{i=1}^{N} P(\vec{y}_i | c_i, \theta_{c_i})$ and $P(\Theta; \mathcal{H}) = \prod_{j=1}^{K} P(\theta_j; \mathcal{H})$.

In Equation 2.9 the proportionality sign hides the marginal probability of the data under the model (also called the "evidence") which cannot be computed analytically. In

such cases Markov chain Monte Carlo (MCMC) methods such as those reviewed in Neal [1993] can be used to obtain a discrete representation of the posterior by sampling from this unnormalized density. These methods simulate a discrete Markov process whose equilibrium distribution is the posterior distribution. Other posterior inference methods can be used here, particularly the variational approach outlined in Jordan et al. [1999], but in this work we will focus on MCMC methods for reasons that will shortly become apparent.

What we have now is a Bayesian Gaussian mixture model. Citing the literature we have asserted that MCMC techniques can be used to estimate a posterior distribution in such models. However, we have not yet surmounted our assumption about knowing K a priori. This is important in modeling situations where either estimating K from the data is of importance or when models estimated with different values of K produce varying inference results.

What we would like is a model whose posterior distribution manifests models that vary in the number of mixture components. Inference using such a model would allow for the variability resulting from inference in different models with different values of K to be averaged away. In addition, since such a model will implicitly define a distribution over K, inference about the cardinality of hidden classes can also be performed.

In Green [1995] one approach to building this kind of posterior distribution was developed: reversible jump Markov chain Monte Carlo (RJMCMC). RJMCMC, as the name implies, is a MCMC technique. In RJMCMC additional special "dimension shifting" transition rules are specified such that K can vary from one step of the sampler to the next. These rules are typically complex as detail balance must be preserved across transitions that are not one-to-one. Regardless, RJMCMC applied in the context of a Bayesian Gaussian mixture model produces a posterior distribution over mixture models that vary in K. We will not review RJMCMC for Bayesian Gaussian mixture modeling here as our focus will soon shift to sequential posterior estimation which is not possible in the RJMCMC framework.

In contrast to RJMCMC there is what is called the "nonparametric Bayesian" approach to modeling. Nonparametric Bayesian models are models in which the prior complexity of the model is unbounded (here $K \to \infty$). To clarify: for RJMCMC as applied to finite Gaussian mixture modeling this is not the case; RJMCMC specifies transition rules between finite mixture models. A NPB GMM is a model with fixed infinite complexity. This choice of infinite prior complexity corresponds to assuming that there are an infinite number of hidden classes. This might seem problematic as the mixture model thereby consists of an infinite number of mixture densities. However first note that only a finite number can be

manifest by a finite dataset. Additionally note that if we utilize a prior that favors "sparsity" in the class prior probabilities $\vec{\pi}$ then most classes will be allocated zero posterior probability mass (this is the case for the choice of prior and its parameterization that we made earlier). The infinite Gaussian mixture model (IGMM) developed by Rasmussen [2000] is exactly such a model. The IGMM is just one specific example of a Dirichlet process mixture model of which many exist in the literature [Neal, 1998, 2000; MacEachern and Muller, 1998] to highlight just a few.

2.2 Infinite Gaussian mixture model

Towards understanding the IGMM note that the only terms in Equation 2.9 that explicitly depend on K are $P(c_i|\vec{\pi})$ and $P(\vec{\pi}|\alpha)$ as both involve $\vec{\pi}$ whose dimensionality is K. Of course the likelihood $P(\mathcal{Y}|\mathcal{C},\Theta)$ and the class parameter prior $P(\Theta;\mathcal{H})$ also implicitly depend on K as it is the number of hidden classes. This dependence on K is indirect however as both the likelihood and the class parameter prior only need to be evaluated for realized hidden classes (i.e. hidden classes to which at least one observation is attributed).

The IGMM arises as $K \to \infty$. While it is possible work with directly with such infinite dimensional models it is easier to consider the Bayesian mixture model having integrated $\vec{\pi}$ out. So, operating on $P(c_i|\vec{\pi})$ and $P(\vec{\pi}|\alpha)$, recognize that $\vec{\pi}$ can be marginalized out because the Dirichlet prior is conjugate to the multinomial likelihood

$$\begin{aligned} P(\mathcal{C}|\alpha) &= \int d\vec{\pi} \prod_{i=1}^{N} P(c_i|\vec{\pi}) P(\vec{\pi}\,|\alpha) \\ &= \frac{\prod_{k=1}^{K} \Gamma(m_k + \frac{\alpha}{K})}{\Gamma(\frac{\alpha}{K})^K} \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)}. \end{aligned} \tag{2.10}$$

This expression is the joint probability of a single labeling $\mathcal{C}$. Since permutation of the labels assigned to a single set of observations results in no change in the semantics of the labeling, we would like a model that expresses the probability of partitions of the data rather than labelings per se. A partition of the data is a grouping of the data into nonoverlapping subsets and is independent of the labels given to each of the subsets. The number of different assignments of K labels that can be assigned to a partioning of the data with $K_+ < K$ bins is $\frac{K!}{(K-K_+)!}$. As each partition has the same marginal probability under

Equation 2.10 the total probability of any one partitioning of the data is

$$P(\mathcal{C}|\alpha) = \frac{K!}{(K-K_+)!}\frac{\prod_{k=1}^{K}\Gamma(m_k+\frac{\alpha}{K})}{\Gamma(\frac{\alpha}{K})^K}\frac{\Gamma(\alpha)}{\Gamma(N+\alpha)}. \tag{2.11}$$

It is the limit of this expression as $K \to \infty$ that turns the Bayesian GMM into the IGMM [Rasmussen, 2000]. The limiting expression (taken from Griffiths and Ghahramani [2005]) is given here without derivation

$$P(\mathcal{C}|\alpha) = \alpha^{K_+}\left(\prod_{k=1}^{K_+}(m_k-1)!\right)\frac{\Gamma(\alpha)}{\Gamma(N+\alpha)} \tag{2.12}$$

where , $m_k = \sum_{i=1}^{N}\mathbb{I}(c_i = k)$ is the number of items in class k ($\mathbb{I}()$ is the indicator function) and $\Gamma()$ is the Gamma function with $\Gamma(\alpha) = (\alpha-1)\Gamma(\alpha-1)$. As before K_+ is the number of bins in the partition; this being equivalent to the number of classes containing at least one item.

In order to do posterior estimation in such a model one needs to be able to at a minimum evaluate Equation 2.12 which is clearly the case. In order to do Gibbs sampling in such a model one needs to be able to sample from $P(c_i = k|\mathcal{C}_{-i})$ where $\mathcal{C}_{-i}$ is all labels except for c_i. A sequential generative process for sampling from this conditional distribution was explored in depth by Pitman [2002] and is given here

$$P(c_i = k|\mathcal{C}_{-i}) = \begin{cases} \frac{m_k}{i-1+\alpha} & k \leq K_+ \\ \frac{\alpha}{i-1+\alpha} & k = K_+ \end{cases} \tag{2.13}$$

This process is known as the Chinese restaurant process (CRP) because of a story that illustrates the process [Pitman, 2002]. Repeated here, the story describes a method of sequentially seating customers stochastically at tables in a Chinese restaurant with an infinite number of tables: the first customer enters the restaurant and is seated at the first table. Subsequent customers enter the restaurant and are stochastically seated according to the number of people already sitting at each table. If there are many people already sitting at a table (i.e., m_k is large for table k) then the probability of being seated at that table is high and vice versa. However, there is always a small probability of being seated at a table that has no prior occupants; $\alpha/(i-1+\alpha)$ where i is the number of the current customer being seated.

The generative view of the infinite Gaussian mixture model with this prior on the class identifiers is that the class identifiers for N datapoints are generated from the Chinese

restaurant process. Depending on α this will yield some value of $K_+ < N$. Observations then are generated from each of K_+ Gaussian densities whose parameters are each drawn independently from the multivariate normal inverse Wishart (MVN-IW) prior.

The IGMM can be seen as a Dirichlet process mixture model in which the class ids are explicitly represented. Dirichlet process mixture models are alternately notated

$$\begin{aligned} \mathcal{G} &\sim \mathrm{DP}(\alpha \mathcal{G}_0) \\ \theta_i &\sim \mathcal{G} \\ \mathbf{y}_i|\theta_i &\sim \mathcal{N}(\cdot|\theta_i) \end{aligned}$$

where the variables have the same meaning as before; however, the generative story is different. Here a distribution $\mathcal{G}$ is generated from a Dirichlet process (DP) with parameters α and $\mathcal{G}_0$. From $\mathcal{G}$ not necessarily unique parameters θ_i are drawn for each datapoint circumventing the need for explicit class identifiers. In this formulation sharing of these class parameters serves the same purpose as the class indicator variables $\mathcal{C}$ in the limiting formulation above. The Dirichlet process is a prior over infinite dimensional multinomials (probability density functions) [Ferguson, 1973; Blackwell and MacQueen, 1973]. The distribution over partitions in Equation 2.12 is closely related to the the marginal distribution of the θ_i's in a DP mixture model in which $\mathcal{G}$ has been integrated out in that the pattern of sharing of θ_i's in such a model generates a partition of the data. The distribution of these partitions is the same as the distribution given in Equation 2.12.

Of the other ways to construct and sample from the Dirichlet process and by extension DP mixture models, the stick breaking construction of Sethuraman [1994] is one particularly important example. Among other benefits the stick breaking construction makes sampling methodologies different than those treated in this review possible. We limit ourselves in this review to the limiting construction because our ultimate interest is in sequential posterior estimation, which does not yet exist for the estimation approaches based on the stick breaking representation of the Dirichlet process.

Now that we have defined a nonparametric Bayesian prior for our model we need to figure out how to generate samples from the posterior distribution Equation 2.9 in this limiting case. Here we review and investigate three different samplers for doing so. The first two correspond to algorithms two and three of Neal [1998], which themselves were previously set forth in Bush and MacEachern [1996]; West et al. [1994]; MacEachern and Muller [1998]; Neal [1992]. The third is a sequential posterior estimation algorithm that

corresponds to the techniques in Fearnhead and Clifford [2003]; Fearnhead [2004]; Sanborn et al. [2006] and related to Wood and Griffiths [2007]. The first we will call the expanded Gibbs sampler in that the latent class parameters are explicitly represented in the sampler state, the second we will call the collapsed Gibbs sampler (following Liu [2001]) because the latent class parameters are integrated out and the sampler state consists of only the class identifiers. Each of these is an offline or batch sampling algorithm requiring that the entire dataset be present. In contrast, the third algorithm, which we will call the sequential or online sampler, is a sequential posterior estimation algorithm, meaning that each observation is processed once in sequence when estimating the posterior.

2.2.1 Expanded Gibbs sampler

Remember that our goal is build an estimate of the posterior distribution for the infinite Gaussian mixture model by using MCMC to draw samples from the posterior. The expanded Gibbs sampler is the first of three posterior estimation algorithms we present in this work. The expanded Gibbs sampler is an instance of one of the algorithms in Neal [1998] as applied to the IGMM of Rasmussen [2000]. The state of this expanded Gibbs sampler consists of of the class identifiers and the mixture density parameters $\{\mathcal{C}, \Theta\}$. In Gibbs sampling each variable is sampled from its conditional distribution given all other variables [Geman and Geman, 1984]. For this sampler the necessary conditional distributions can be derived by simple algebraic operations on the joint distribution given in Equation 2.12. The conditional distribution for Θ_k given the state of all other variables is

$$P(\theta_k|\mathcal{C}, \mathcal{Y}, \Theta_{-k}, \vec{\pi}, \alpha; \mathcal{H}) \propto \prod_{i:c_i=k} P(\vec{y}_i|c_i, \theta_k) P(\theta_k; \mathcal{H}). \tag{2.14}$$

where $\Theta_{-k} = \{\theta_k, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_N\}$. Likewise we sample all class identifiers $\mathcal{C}$ according to their individual conditional distributions.

$$P(c_i = k|\mathcal{C}_{-i}, \mathcal{Y}, \Theta, \vec{\pi}, \alpha) \propto P(\vec{y}_i|c_i, \Theta) P(c_i|\mathcal{C}_{-i}) \tag{2.15}$$

where $\mathcal{C}_{-i} = \{c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_N\}$. The prior predictive distribution $P(c_i|\mathcal{C}_{-i})$ is given in Equation 2.13 above.

To complete the specification of the expanded Gibbs sampler all that remains to be specified is a method for sampling the remaining free parameter, α. The can be done using a Metropolis update, although it too can be updated using a Gibbs step [Escobar and West, 1995]. The expanded Gibbs sampler algorithm is given in Algorithm 1.

Algorithm 1 Expanded Gibbs sampler for IGMM

$c_1, \dots c_k \leftarrow 0$
$\mathcal{C}_0 \leftarrow \{c_1, \dots c_k\}$
$\Theta_0 \leftarrow \{\}$
$K_+ = 0$
for $s = 1 : \#$ samples **do**
  $\mathcal{C}_s \leftarrow \mathcal{C}_{s-1}$
  $\Theta_s \leftarrow \Theta_{s-1}$
  for $i = 1 : N$ **do**
    if $c_i \neq 0$ **then**
      $m_{k,-i} \leftarrow \sum_{j=1}^{N} \mathbb{I}(c_j = c_i) - 1$
      if $m_{k,-1} = 0$ **then**
        $\{\Theta_s\} \leftarrow \{\Theta_s\} \backslash \Theta_{c_i}$
        $c_j = c_j - 1 \; \forall \; j > i$
        $K_+ = K_+ - 1$
      else
        $m_{k,-i} \leftarrow 0$
      end if
    end if
    sample $c_i \sim P(\cdot|\mathcal{C}_{-i}, \mathcal{Y}, \Theta, \dots)$ // *Using Eqn: 2.15*
    if $c_i \cap \mathcal{C}_s = \emptyset$ **then**
      sample $\vec{\mu}_{c_i}, \mathbf{\Sigma}_{c_i} \sim$ MVN-IW$(\mu_1, \kappa_1, \mathbf{\Lambda}_1, \nu_1)$ // *from the MVN-IW posterior given observation* y_i
      $\Theta_s \leftarrow \{\Theta_{s-1}, \{\vec{\mu}_{c_i}, \mathbf{\Sigma}_{c_i}\}\}$
      $K_+ = K_+ + 1$
    end if
  end for
  for $k = 1 : K_+$ **do**
    sample $\theta_k \sim P(\cdot|\mathcal{C}, \mathcal{Y}, \Theta_{-k}, \dots)$ // *Using Eqn: 2.14*
  end for
  sample α using a Metropolis step
end for

2.2.2 Collapsed Gibbs sampler

It is also possible in this model to more fully exploit our choices of conjugate priors. Having chosen the multivariate-normal inverse Wishart prior for the multivariate normal class distributions makes it possible to analytically integrate these parameters out as well yielding an expression for a posterior over only $\mathcal{C}$

$$\begin{aligned} P(\mathcal{C}|\mathcal{Y};\mathcal{H}) &= \int d\Theta P(\mathcal{C}, \Theta|\mathcal{Y};\mathcal{H}) \\ &\propto P(\mathcal{C};\mathcal{H}) \int d\Theta P(\mathcal{Y}|\mathcal{C}, \Theta;\mathcal{H}) P(\Theta;\mathcal{H}). \end{aligned} \tag{2.16}$$

The advantage of analytically marginalizing out the normal parameters Θ is that the state space of the sampler is reduced. Typically this leads to faster convergence of the

sampler to the equilibrium distribution [Liu, 2001]. Furthermore, posterior inference is made simpler and more accurate by analytically integrating out a set of parameters rather than doing Monte Carlo integration over a discrete posterior representation.

Continuing, here we demonstrate how to marginalize out Θ. In the following expression the joint over the data is considered and broken into K parts, each corresponding to one class.

$$\begin{aligned} P(\mathcal{C}|\mathcal{Y};\mathcal{H}) &= \int d\Theta P(\mathcal{C},\Theta|\mathcal{Y};\mathcal{H}) \\ &\propto P(\mathcal{C};\mathcal{H}) \int \cdots \int d\theta_1 \cdots d\theta_K \left(\prod_{j=1}^{K} P(\theta_j;\mathcal{H}) \right) \prod_{i=1}^{N} P(\vec{y}_i|c_i=j,\theta_j) \\ &\propto P(\mathcal{C};\mathcal{H}) \int \cdots \int d\theta_1 \cdots d\theta_K \prod_{j=1}^{K} \prod_{i=1}^{N} \mathbb{I}(c_i=j, P(\vec{y}_i|c_i=j,\theta_j)P(\theta_j;\mathcal{H})) \\ &\propto P(\mathcal{C};\mathcal{H}) \prod_{j=1}^{K} \int d\theta_j \prod_{i=1}^{N} \mathbb{I}(c_i=j, P(\vec{y}_i|c_i=j,\theta_j)P(\theta_j;\mathcal{H})) \end{aligned} \tag{2.17}$$

where $\mathbb{I}(expr, val)$ is an indicator function that returns val if $expr$ is true and one otherwise. This shows that the collapsed posterior is the product of the prior over $\mathcal{C}$ and K independent terms, each of which is the posterior probability of an independent set of observations under the class to which they are currently attributed.

If we isolate the set of datapoints attributed to a single class $\mathcal{Y}^{(j)} = \{y_i : c_i = j\}$ and denote the cardinality of that set $N^{(j)} = |\mathcal{Y}^{(j)}|$ then we can rewrite the right hand side of the final line of 2.17 in a simpler form.

$$\prod_{j=1}^{K} \int d\theta_j \prod_{i=1}^{N} \mathbb{I}\left(c_i=j, P(\vec{y}_i|c_i=j,\theta_j)P(\theta_j;\mathcal{H})\right) = \prod_{j=1}^{K} \int d\theta_j \prod_{i=1}^{N^{(j)}} P(\vec{y}_i^{(j)}|c_i^{(j)},\theta_j)P(\theta_j;\mathcal{H})$$

Focusing on a single cluster and dropping the class index j for now we see that

$$P(\mathcal{Y}|\mathcal{H}) = \int d\theta \prod_{i=1}^{N} P(\vec{y}_i|\theta)P(\theta;\mathcal{H}) \tag{2.18}$$

is the standard likelihood conjugate prior integral for a MVN-IW where, remembering that $\theta = \{\vec{\mu}, \mathbf{\Sigma}\}$ the expression for the MVN likelihood term, $P(\vec{y}_i|\theta)$ expands to the familiar

MVN normal joint distribution for N i.i.d. observations

$$\prod_{i=1}^{N} P(\vec{y}_i|\theta) = (2\pi)^{-\frac{Nd}{2}}|\mathbf{\Sigma}|^{-\frac{N}{2}}e^{-\frac{1}{2}\mathrm{tr}(\mathbf{\Sigma}^{-1}\mathbf{S}_0)} \tag{2.19}$$

where $\mathbf{S}_0 = \sum_{i=1}^{N}(\vec{y}_i - \vec{\mu})(\vec{y}_i - \vec{\mu})^T$. Here, following convention, $|\mathbf{X}|$ means the matrix determinant of $\mathbf{X}$.

For ease of reference we provide the MVN-IW prior $P(\Theta;\mathcal{H})$ here

$$\begin{aligned} P(\theta;\mathcal{H}) &= P(\vec{\mu},\mathbf{\Sigma}|\vec{\mu}_0,\mathbf{\Lambda}_0,\nu_0,\kappa_0) \\ &= \frac{(\frac{2\pi}{\kappa_0})^{-\frac{d}{2}}|\mathbf{\Sigma}|^{-\frac{1}{2}}e^{-\frac{\kappa_0}{2}(\vec{\mu}-\vec{\mu}_0)^T\mathbf{\Sigma}^{-1}(\vec{\mu}-\vec{\mu}_0)}}{2^{\frac{\nu_0 d}{2}}\pi^{\frac{d(d-1)}{4}}\prod_{j=1}^{d}\Gamma(\frac{\nu_o+1-j}{2})}|\mathbf{\Lambda}_0|^{\frac{\nu_0}{2}}|\mathbf{\Sigma}|^{-\frac{\nu_0+d+1}{2}}e^{-\frac{1}{2}\mathrm{tr}(\mathbf{\Lambda}_0\mathbf{\Sigma}^{-1})} \end{aligned} \tag{2.20}$$

Combining the joint multivariate normal likelihood in Equation 2.19 with the MVN-IW posterior given in Equation 2.20 yields the expanded form of the posterior given in Equation 2.18

$$= \frac{1}{\mathcal{Z}_0}\int\int d\vec{\mu}d\mathbf{\Sigma}|\mathbf{\Sigma}|^{-(\frac{\nu_0+d}{2}+1)-\frac{N}{2}}e^{-\frac{1}{2}\mathrm{tr}(\mathbf{\Lambda}_0\mathbf{\Sigma}^{-1})-\frac{1}{2}\mathrm{tr}(\mathbf{\Sigma}^{-1}\mathbf{S}_0)-\frac{\kappa_0}{2}(\vec{\mu}-\vec{\mu}_0)^T\mathbf{\Sigma}^{-1}(\vec{\mu}-\vec{\mu}_0)}$$

Here $\mathcal{Z}_0$ is the normalization constant for the MVN-IW prior

$$\mathcal{Z}_0 = (2\pi)^{\frac{Nd}{2}}(\frac{2\pi}{\kappa_0})^{\frac{d}{2}}2^{\frac{\nu_0 d}{2}}\pi^{\frac{d(d-1)}{4}}\prod_{j=1}^{d}\Gamma(\frac{\nu_0+1-j}{2})|\mathbf{\Lambda}_0|^{-\frac{\nu_0}{2}}. \tag{2.21}$$

Now, the choice earlier in this chapter of a conjugate prior for the MVN class parameters helps tremendously. This seemingly daunting integral has a simple analytical solution thanks to conjugacy. Following Gelman et al. [1995] pg. 87 in making the following variable substitutions

$$\begin{aligned} \vec{\mu}_n &= \frac{\kappa_0}{\kappa_0+N}\vec{\mu}_0 + \frac{N}{\kappa_0+N}\bar{y} \\ \kappa_n &= \kappa_0 + N \\ \nu_n &= \nu_0 + N \\ \mathbf{\Lambda}_n &= \mathbf{\Lambda}_0 + \mathbf{S} + \frac{\kappa_0 n}{\kappa_0+N}(\bar{y}-\vec{\mu}_0)(\bar{y}-\vec{\mu}_0)^T \end{aligned}$$

where

$$\mathbf{S} = \sum_{i=1}^{N} (\vec{y}_i - \bar{y})(\vec{y}_i - \bar{y})^T$$

then

$$P(\mathcal{Y};\mathcal{H}) = \frac{1}{\mathcal{Z}_0} \int\int d\vec{\mu} d\mathbf{\Sigma} |\mathbf{\Sigma}|^{-(\frac{\nu_n+d}{2}+1)} e^{-\frac{1}{2}\mathrm{tr}(\mathbf{\Lambda}_n \mathbf{\Sigma}^{-1}) - \frac{\kappa_n}{2}(\vec{\mu}-\vec{\mu}_n)^T \mathbf{\Sigma}^{-1}(\vec{\mu}-\vec{\mu}_n)}$$

can be solved immediately by realizing that this is itself a MVN-IW distribution and the integral is simply the inverse of its normalization constant with the same variable substitutions applied, i.e.,

$$\begin{aligned}
\mathcal{Z}_n &= \int\int d\vec{\mu} d\mathbf{\Sigma} |\mathbf{\Sigma}|^{-(\frac{\nu_n+d}{2}+1)} e^{-\frac{1}{2}\mathrm{tr}(\mathbf{\Lambda}_n \mathbf{\Sigma}^{-1}) - \frac{\kappa_n}{2}(\vec{\mu}-\vec{\mu}_n)^T \mathbf{\Sigma}^{-1}(\vec{\mu}-\vec{\mu}_n)} \\
&= (2\pi)^{\frac{Nd}{2}} (\frac{2\pi}{\kappa_n})^{\frac{d}{2}} 2^{\frac{\nu_n d}{2}} \pi^{\frac{d(d-1)}{4}} \prod_{j=1}^{d} \Gamma(\frac{\nu_n + 1 - j}{2}) |\mathbf{\Lambda}_n|^{-\frac{\nu_n}{2}}
\end{aligned}$$

which yields

$$\begin{aligned}
P(\mathcal{Y};\mathcal{H}) &= \frac{\mathcal{Z}_n}{\mathcal{Z}_0} \\
&= \frac{(\frac{2\pi}{\kappa_n})^{\frac{d}{2}} 2^{\frac{\nu_n d}{2}} \pi^{\frac{d(d-1)}{4}} \prod_{j=1}^{d} \Gamma(\frac{\nu_n+1-j}{2}) |\mathbf{\Lambda}_n|^{-\frac{\nu_n}{2}}}{(\frac{2\pi}{\kappa_0})^{\frac{d}{2}} 2^{\frac{\nu_0 d}{2}} \pi^{\frac{d(d-1)}{4}} \prod_{j=1}^{d} \Gamma(\frac{\nu_0+1-j}{2}) |\mathbf{\Lambda}_0|^{-\frac{\nu_0}{2}}} \\
&= (\frac{\kappa_0}{\kappa_n})^{\frac{d}{2}} 2^{\frac{d}{2}(\nu_n - \nu_0)} \frac{|\mathbf{\Lambda}_0|^{\frac{\nu_0}{2}} \prod_{j=1}^{d} \Gamma(\frac{\nu_n+1-j}{2})}{|\mathbf{\Lambda}_n|^{\frac{\nu_n}{2}} \prod_{j=1}^{d} \Gamma(\frac{\nu_0+1-j}{2})}
\end{aligned}$$

which if n is even simplifies further

$$\begin{aligned}
&= (\frac{\kappa_0}{\kappa_n})^{\frac{d}{2}} 2^{\frac{d}{2}(\nu_n - \nu_0)} \frac{|\mathbf{\Lambda}_0|^{\frac{\nu_0}{2}}}{|\mathbf{\Lambda}_n|^{\frac{\nu_n}{2}}} \frac{\prod_{j=1}^{d} [\Gamma(\frac{\nu_0+1-j}{2}) \prod_{i=1}^{\frac{N}{2}} (\frac{\nu_n+1-j}{2} - i)]}{\prod_{j=1}^{d} \Gamma(\frac{\nu_0+1-j}{2})} \\
&= (\frac{\kappa_0}{\kappa_n})^{\frac{d}{2}} 2^{\frac{d}{2}(\nu_n - \nu_0)} \frac{|\mathbf{\Lambda}_0|^{\frac{\nu_0}{2}}}{|\mathbf{\Lambda}_n|^{\frac{\nu_n}{2}}} \prod_{j=1}^{d} \prod_{i=1}^{\frac{N}{2}} (\frac{\nu_n + 1 - j}{2} - i)
\end{aligned}$$

Algorithm 2 Collapsed Gibbs sampler for the IGMM

$c_1, \dots c_k \leftarrow 0$
$\mathcal{C}_0 \leftarrow \{c_1, \dots c_k\}$
$K_+ = 0$
for $s = 1 : \#$ samples **do**
  $\mathcal{C}_s \leftarrow \mathcal{C}_{s-1}$
  for $i = 1 : N$ **do**
    if $c_i \neq 0$ **then**
      $m_{k,-i} \leftarrow \sum_{j=1}^{N} \mathbb{I}(c_j = c_i) - 1$
      if $m_{k,-1} = 0$ **then**
        $c_j = c_j - 1 \; \forall \; j > i$
        $K_+ = K_+ - 1$
      end if
    else
      $m_{k,-i} \leftarrow 0$
    end if
    sample $c_i \sim P(\cdot|\mathcal{C}_{-i}, \mathcal{Y}, \dots)$ // *Using Eqn's: 2.23 and 2.24*
    if $c_i \cap \mathcal{C}_s = \emptyset$ **then**
      $K_+ = K_+ + 1$
    end if
  end for
  sample α using a Metropolis or Gibbs step
end for

Now we have an expression for the posterior distribution that is simpler than that given in Equation 2.9

$$\begin{aligned} P(\mathcal{C}|\mathcal{Y};\mathcal{H}) &= \int d\Theta P(\mathcal{C}, \Theta|\mathcal{Y};\mathcal{H}) \\ &\propto P(\mathcal{Y}|\mathcal{C};\mathcal{H})P(\mathcal{C}) \\ &\propto \prod_{j=1}^{K_+} P(\mathcal{Y}^{(j)};\mathcal{H})P(\mathcal{C}). \end{aligned} \tag{2.22}$$

By integrating out the class parameters all that needs to be sampled now are the class identifiers. To sample a single class identifier c_i one computes the likelihood of the associated datapoint y_i under every currently existing cluster. This is simply the posterior predictive distribution for a MVN-IW model which has a standard known form which is given below. These likelihoods are weighted by the prior probability that c_i would take on a particular value given the DP prior and the current value of all the other class identifiers.

For the collapsed Gibbs sampler then the sampler state consists of $\mathcal{C}$ and α as shown in

Algorithm 2.2.2. The sampling step for α is the same as in the previous expanded Gibbs sampling algorithm.

Here, the updates for the class labels are given

$$\begin{aligned} P(c_i = j|\mathcal{C}_{-i}, \mathcal{Y}, \alpha; \mathcal{H}) &\propto P(\mathcal{Y}|\mathcal{C}; \mathcal{H})P(\mathcal{C}|\alpha) \\ &\propto \prod_{j=1}^{K_+} P(\mathcal{Y}^{(j)}; \mathcal{H})P(c_i = j|\mathcal{C}_{-i}, \alpha) \\ &\propto P(\mathcal{Y}^{(j)}; \mathcal{H})P(c_i = j|\mathcal{C}_{-i}, \alpha) \\ &\propto P(y_i|\mathcal{Y}^{(j)}\backslash y_i; \mathcal{H})P(c_i = j|\mathcal{C}_{-i}, \alpha) \end{aligned} \tag{2.23}$$

where $\mathcal{Y}^{(j)}\backslash y_i$ is the set $\mathcal{Y}^{(j)}$ without y_i (y_i is "removed" from the class to which it belongs when sampling). In the infinite case we also must consider the probability of starting a new cluster. This is given by

$$P(c_i \neq j, 1 \leq j \leq K_+|\mathcal{C}_{-i}, \mathcal{Y}, \alpha; \mathcal{H}) \propto P(y_i; \mathcal{H})P(c_i \neq j, 1 \leq j \leq K_+|\mathcal{C}_{-i}, \alpha) \tag{2.24}$$

where for our model $P(y_i|\mathcal{Y}^{(j)}\backslash y_i; \mathcal{H})$ is the posterior predictive distribution for a new datapoint under a multivariate model with a multivariate normal inverse-Wishart prior. From Gelman et al. [1995] pg. 88 we know that this is multivariate Student-t

$$y_i|\mathcal{Y}^{(j)}\backslash y_i; \mathcal{H} \sim t_{\nu_n - D+1}(\vec{\mu}_n, \mathbf{\Lambda}_n(\kappa_n + 1)/(\kappa_n(\nu_n - D + 1))) \tag{2.25}$$

where $\vec{\mu}_n, \mathbf{\Lambda}_n, \kappa_n, \nu_n$ are exactly as before and D is, also as before, the dimensionality of y_i. In the case for generating a novel cluster identifier the same posterior predictive distribution is used; however, as there are no other observations the original hyperparameters are used instead, i.e.

$$y_i; \mathcal{H} \sim t_{\nu_0 - D+1}(\vec{\mu}_0, \mathbf{\Lambda}_0(\kappa_0 + 1)/(\kappa_0(\nu_0 - D + 1))) \tag{2.26}$$

for ease of reference the multivariate Student-t density function is given here

$$P(y) = t_\nu(y|\mu, \mathbf{W}) = \frac{\Gamma((\nu + D)/2)}{\Gamma(\nu/2)\nu^{D/2}\pi^{D/2}}|\mathbf{W}|^{1/2}(1 + \frac{1}{\nu}(y - \mu)^T\mathbf{W}^{-1}(y - \mu))^{-(\nu+D)/2}$$

We now have reviewed two ways of sampling from the posterior distribution of an infinite

Gaussian mixture model, one in which the class parameters remained explicitly represented and one in which they were integrated away. The point of going through this review exercise, besides putting lots of extraneous gratuitous mathematics in this dissertation, is to provide enough introductory material to nonparametric Bayesian approaches to make uptake of the new models in Chapter 4 easy.

2.2.3 Sequential posterior estimation

One of the most exciting things about the nonparametric mixture modeling approach we are reviewing is that sequential posterior estimation is possible in such models. Sequential posterior estimation means that estimation of the posterior distribution is accomplished by processing each observation once in sequence. In the IGMM this is possible because the prior on $\mathcal{C}$ yields an analytic expression for the conditional distribution of the label of a single point given the labels for all of the others $P(c_i|\mathcal{C}_{-i})$.

To explain how this makes sequential posterior estimation possible remember that $\vec{y}_i$ is the i^{th} observation, and let $\mathcal{Y}^{(1:i)}$ be all observations up to and including the i^{th} (similarly for the class identifiers $\mathcal{C}^{(1:i)}$). Note that because of the form of the class identifier prior it is easy to sample from $P(c_i|\mathcal{C}^{(1:i-1)})$; to do so sample the i^{th} class identifier given the first $i-1$ identifiers directly from the prior (using Equation 2.13). Knowing that we can exploit this fact then applying Bayes' rule we can write the unnormalized posterior on $\mathcal{C}^{(1:i)}$ given $\mathcal{Y}^{(1:i)}$ in the following way

$$\hat{P}(\mathcal{C}^{(1:i)}|\mathcal{Y}^{(1:i)}) \propto P(\vec{y}_i|\mathcal{C}^{(1:i)}, \mathcal{Y}^{(1:i-1)})P(c_i|\mathcal{Y}^{(1:i-1)}).$$

Since we can evaluate $P(\vec{y}_i|\mathcal{C}^{(1:i)}, \mathcal{Y}^{(1:i-1)})$, we can obtain weighted samples from the normalized posterior $P(\mathcal{C}^{(1:i)}|\mathcal{Y}^{(1:i)})$ using importance sampling with a proposal distribution of

$$\begin{aligned} P(c_i|\mathcal{Y}^{(1:i-1)}) &= \sum_{\Delta\mathcal{C}^{(1:i-1)}} P(c_i|\mathcal{C}^{(1:i-1)})P(\mathcal{C}^{(1:i-1)}|\mathcal{Y}^{(1:i-1)}) \\ &\approx \sum_{l=1}^{L} w_{\{l\}}^{(1:i-1)} P(c_i|\mathcal{C}_{\{l\}}^{(1:i-1)}) \end{aligned}$$

where $w_{\{l\}}^{(1:i)}$ is the weight for the l^{th} sample having integrated i observations, $\mathcal{C}_{\{l\}}^{(1:i)}$ is the corresponding sample and $\Delta\mathcal{C}^{(1:i-1)}$ is shorthand for all possible assignments of labels 1

Algorithm 3 Particle filter posterior estimation for the IGMM

1: $\mathcal{C}^{(1)}_{\{:\}} \leftarrow 1$
2: **for** $i = 2 : N$ **do**
3: **for** $l = 1 : L$ **do**
4: sample $\mathcal{C}^{(i)}_{\{l\}} \sim P(\mathcal{C}^{(i)}_{\{l\}} = k|\mathcal{C}^{(1:i-1)}_{\{l\}})$ // *Using Eqn. 2.13*
5: calculate weight $\omega^{(i)}_{\{l\}} \propto P(\mathcal{Y}^{(i)}|\mathcal{C}^{(1:i)}_{\{l\}}, \mathcal{Y}^{(1:i-1)}, \ldots)$ // *Using Eqn.'s 2.23 and 2.24*
6: **end for**
7: normalize the weights $\omega^{(i)}_{\{l\}} \leftarrow \omega^{(i)}_{\{l\}} / \sum_{j=}^{L} \omega^{(i)}_{\{j\}} \; \forall \, l$
8: $\mathcal{C}^{(i)}_{\{:\}} \leftarrow \text{resample}(\omega^{(i)}_{\{:\}}, \mathcal{C}^{(i)}_{\{:\}})$
9: **end for**

through $i-1$. "Particle filter" is the name given to sequential density estimation approaches that follow this form. In a particle filter, weighted "particles" (the samples and weights) are used to form a discrete representation of the distribution of interest (here the posterior distribution over class identifiers given observations)

$$\{w^{(1:i-1)}_{\{l\}}, \mathcal{C}^{(1:i-1)}_{\{l\}}\} \approx P(\mathcal{C}^{(1:i-1)}|\mathcal{Y}^{(1:i-1)}).$$

These particles are updated once per observation in sequence. The discrete particle representation of the posterior distribution is an approximation in the traditional Monte Carlo integration sense. As the number of particles goes to infinity, averages over the discrete representation converge to expectations of the true distribution

$$\begin{aligned} \lim_{L\to\infty} \sum_{l=1}^{L} w^{(1:i-1)}_{\{l\}} g(\mathcal{C}^{(1:i-1)}_{\{l\}}) &= \mathrm{E}_{P(\mathcal{C}^{(1:i-1)}|\mathcal{Y}^{(1:i-1)})}[g] \\ &= \sum_{\Delta\mathcal{C}^{(1:i-1)}} P(\mathcal{C}^{(1:i-1)}|\mathcal{Y}^{(1:i-1)}) g(\mathcal{C}^{(1:i-1)}) \end{aligned} \tag{2.27}$$

Refering to Algorithm 2.2.3 and noting that $P(\vec{y}_i|\mathcal{C}^{(1:i)}, \mathcal{Y}^{(1:i-1)})$ is given by Equation 2.25 for existing clusters and Equation 2.26 for putative new clusters this completes the mathematical description of the basic particle filter.

Starting with samples representing the posterior distribution after one observation (this is easy as the first customer under the Chinese restaurant process always sits at the first table), an estimate of the posterior after two observations can be arrived at. This sets up a clear recursion for estimating the posterior distribution accounting for any number of observations. This type of sequential posterior estimation is known as a sequential importance

sampling or as a SIS particle filter [Doucet et al., 2001].

This approach to posterior estimation in nonparametric Bayesian models is not restricted to the IGMM nor to nonparametric models based on the Dirichlet process. For instance, in Chapter 4 a similar algorithm is developed for NPB matrix factorization models.

Unfortunately this basic implementation, while correct in the limit as the number of particles goes to infinity, will only work for a limited and small number of observations because of particle degeneracy. Particle degeneracy is a way to describe the situation in which the number of unique particles decreases, leaving only a few unique particles and correspondingly a sharply and possibly incorrectly peaked posterior. The resampling step used in sequential importance resampling particle filters [Doucet et al., 2001] helps but does not completely alleviate the problem in particle filters for Dirichlet process mixture models [Fearnhead and Clifford, 2003].

To get around the particle degeneracy problem we follow Fearnhead and Clifford [2003] in adopting a resampling strategy that ensures a diverse particle set. In Fearnhead and Clifford [2003] we are reminded that these two particle sets are equivalent

$$\frac{1}{3}, \{1, 2, 1\}; \frac{1}{3}, \{1, 1, 1\}; \frac{1}{3}, \{1, 1, 1\}$$

and

$$\frac{1}{3}, \{1, 2, 1\}; \frac{2}{3}, \{1, 1, 1\}$$

where here the fraction represents the weight given to each particle and the three integers in the brackets are the particles which themselves indicate the class labels given to three hypothetical observations. Clearly, the second representation is more efficient in that it requires fewer particles to represent the distribution with the same accuracy. This efficiency is merely a product of the measure being weighted rather than unweighted.

While the details of the exact methodology are available in Fearnhead and Clifford [2003], we give here a brief accounting for IGMM's. In the step in which the class label for the i^{th} observation is sampled from the CRP prior we need not restrict ourselves to sampling only a single new class label from the prior. In this model it is possible to exhaustively enumerate all possible values of $c_i|\mathcal{C}_{-i}$ (i.e. no sampling from the prior, merely enumerating all possible next labels). Doing this generates a particle set of size M from the original N particles where $M > N$, with each particle replicated as many times as the number of pre-existing unique labels in the particle plus one to account for the possibility of the new observation having been generated by an as yet unseen latent class. Weights for each of these particles can be computed as before using Equations 2.23 and 2.24, only now computing M

instead of N weights. Having increased the diversity and cardinality of our particle set we now must down-sample the representation to avoid exponential growth in the number of particles. One can resample N particles directly from this discrete distribution; however in Fearnhead and Clifford [2003] a resampling algorithm is developed that is provably optimal in terms preserving the expected weight of particles retained from the large sample set in the smaller sample set. Additionally, their approach limits the number of times a particle can be selected in the down-sampling step to one which yields an optimally diverse particle set. A trivial consequence of their approach is that the particle set is no longer unweighted after resampling like it is in regular SIS particle filtering.

In all of our experiments where we report results for particle filter posterior estimation the approach of Fearnhead and Clifford [2003] is used to maintain the particle set.

2.3 Experiments

Prior to applying this model to neural data it is imperative to test some of its characteristics. For instance: how sensitive is the model to its settings of the hyperparameters? In the following sections we investigate this and other characteristics of the infinite Gaussian mixture modeling approach. In Section 2.3.1 we use the IGMM model to generate synthetic data and then show that both collapsed Gibbs sampling and particle filtering can be used to estimate a posterior distribution that admits with high probability the true generative model. In Section 2.3.2 we investigate how well the model deals with partially overlapping clusters which we will call ambiguous clusters. We do this by estimating multiple models for high dimensional data, in each of which we cluster low dimensional projections of the original high dimensional data. By clustering low dimensional projections of high dimensional data we are able to manipulate the level of cluster overlap. In Section 2.3.3 we investigate the sensitivity of the IGMM to hyperparameter setting.

2.3.1 Estimation

Demonstrating that an estimation approach is able to discover a posterior distribution over models that is peaked at the true model yields reassurance that if the model and the hyperparameters are chosen correctly, then the estimation procedure will arrive at a posterior estimate that gives high score to the true model. The theoretical guarantees for this are difficult to establish; however, in practice this is often asserted as evidence of the correctness and effectiveness of a posterior estimation approach. To perform this check, four dimensional synthetic data was generated from the generative model underlying the

IGMM. The parameters used to generate a synthetic dataset were $\mu_0 = [0\ 0\ 0\ 0], v_0 = 50, k_0 = .05, \alpha = .4, \mathbf{\Lambda}_0^{-1} = \text{diag}(.1)$. One thousand samples were drawn from this model by first sampling the class identifiers by simulating a Chinese restaurant process with $\alpha = .4$. Sampling the class identifiers in this way implicitly determines K_+ the number of non-empty classes resulting in, for this synthetic dataset, $K_+^{\text{true}} = 6$. We then sampled K_+^{true} class distribution parameters $\{\mu_k, \mathbf{\Sigma}_k\}$ from the multivariate normal inverse Wishart prior as parameterized above. Finally 1000 points were sampled from the mixture densities specified by their true class id c_k. In Figure 2.3 all coordinate plane 2D projections of the 4D data are shown. In the figure the axis labels refer to the dimensions of the 4D data selected for plotting. Datapoints with the same color have the same class id. Worthy of note as it will be exploited later in this section is the fact that some of the projections are highly unambiguous, i.e., having almost no overlap between classes, whereas others introduce ambiguity by virtue of collapsing two classes on top of one another.

As we are interested in the performance of both the Gibbs sampler (here and onwards "Gibbs sampler" refers only to the collapsed Gibbs sampler in the context of infinite Gaussian mixture modeling) and the particle filter, each are demonstrated in estimating a posterior distribution given the synthetic data described in the paragraph above. For this experiment and for both of the posterior estimation algorithms all hyperparameters supplied to the algorithms were set to their true values except α which was varied accross multiple trials. The effects on estimation of varying the number of samples drawn from the posterior is demonstrated. For the Gibbs sampler the number of samples is simply the number of samples; for the particle filter it is the number of particles. We will interchangeably refer to the particles generated by the particle filter as samples as they are, like the Gibbs samples, simply a discrete distribution, and are only unlike the Gibbs samples in that they have nonuniform weights. The Gibbs sampler was run for 1.25 times the number of samples reported for purposes of "burn-in" meaning that the first 25% of the samples were discarded. Here this is appropriate as the Gibbs sampler was observed to converge rapidly. This rapid convergence may be a function of our implementation of the Gibbs sampler. Our implementation was not initialized with random class identifiers but instead the first Gibbs sweep through the data was the same as running the particle filter algorithm with one particle. The remainder of the Gibbs sweeps use the Gibbs updates without modification. We have found that this causes the Gibbs sampler to start very near the mode of the posterior distribution and correspondingly results in rapid convergence to the equilibrium posterior distribution.

In Figure 2.2 measurements of the posterior estimates produced by the particle filter

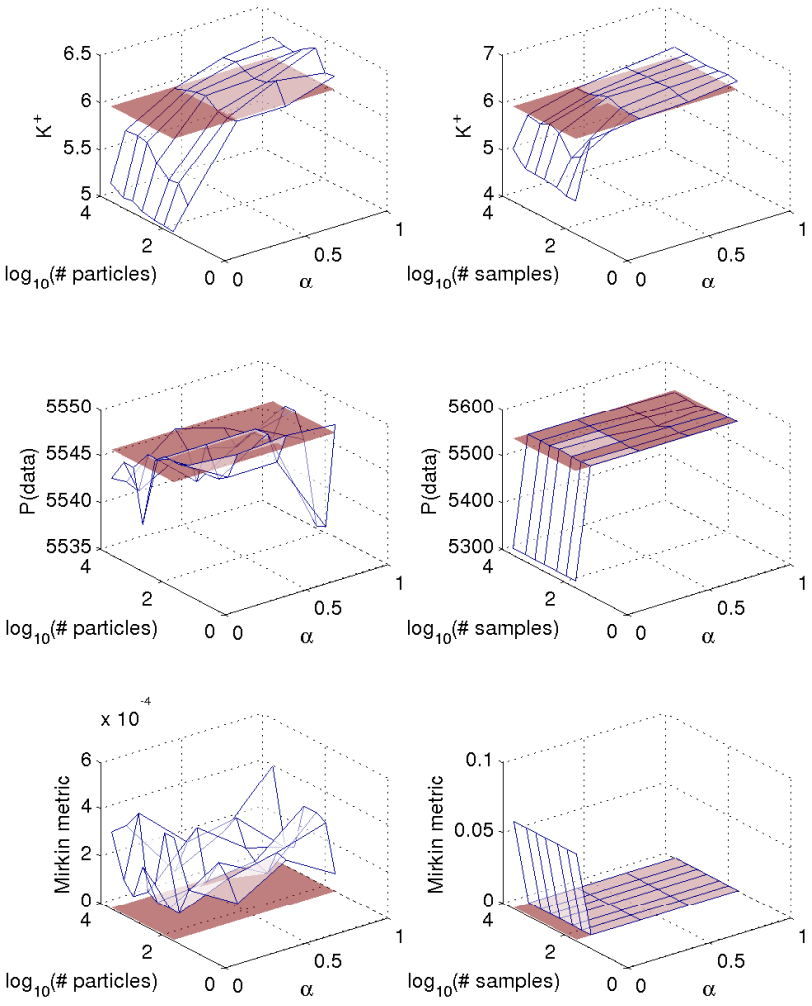

Figure 2.2: Infinite Gaussian mixture model posterior estimation results for synthetic data drawn from the generative model underlying the IGMM. All results shown are the result of averaging over four different pseudorandom number generator seedings. The left column are estimation results for the particle filter posterior estimation algorithm. The right column are estimation results for the Gibbs sampler. The first row of results show the expected value of K_+ as computed over the posterior distribution. The true value, $K_+ = 6$ is shown as a red plane. The second row shows the log posterior score of the MAP model in comparison to the true model, again given by the red plane. The third row shows the Mirkin metric for the MAP model which if greater than zero indicates an imperfect match between clusterings. In each figure the horizontal axes are the logarithm of the number of samples used in the posterior estimate and the value of α used to initialize the estimation routine.

and Gibbs sampler posterior estimation algorithms are shown. The left column shows results for the particle filter while the right column shows those for the collapsed Gibbs sampler. Plotted from top to bottom in both columns is $E[K_+]$, the expected number of latent classes; the posterior probability of the labeling; and the average normalized Mirkin metric, all averaged over four different random seed initializations. The normalized Mirkin metric developed by Mirkin [1996] is a metric for comparing clusterings which Meila [2002] interpreted as the two times the ratio of the number of point pairs which are in the same cluster in the ground truth labeling but are in different clusters under the model labeling to the total number of point pairs. For instance, if the model induces the same clustering but a different labeling, the Mirkin metric is zero.

In all plots the ground truth is shown in solid red while the model estimate is shown by the transluscent white surface with blue lines. Results are plotted over varying values of α and numbers of samples. The true value of $\alpha = .4$ was given above in describing how the synthetic data was generated. The actual (non-logarithmic) number of samples/particles used was 50, 100, 200, 500, 1000, 2000, and 5000.

What is shown here is that both posterior estimation algorithms converge to good posterior estimates in terms of these three measurements given remarkably few samples. Unfortunately the expected number of classes is shown to be sensitive to the assumed value of α. As a distribution over α can itself be estimated from the data, this is not of particular concern; however, in applications where the user wishes to hand specify α some jurisprudence should be applied in making inferences with respect to the marginal over class cardinality. Also, while the particle filter is at first glance worse that the Gibbs sampler in terms of Mirkin metric and log posterior score, the absolute error in both terms is very small, indicating that the particle filter is potentially suitable for most applications.

2.3.2 Handling overlapping classes

Unfortunately it is common in real world data modeling problems to have data with clusters that are not well separated. Multiple clusters may overlap or abut in ways that makes attributing a datapoint to a single class difficult. This is a main justification for adopting a Bayesian approach to clustering. Testing the ability of the nonparametric models to find good clusterings in the presense of ambiguous cluster boundaries is important. To study this we start with the same 4D dataset as described in the previous experiment but cluster it using lower dimensional (3D and 2D) projections. By modeling each of the two dimensional (2D) cardinal projections of the data we develop a measure of the model's ability to perform

in situations of ambiguous cluster boundaries. There is one caveat that is important to mention before proceeding. If a projection direction is chosen to be orthogonal to the line segment between the means of some two clusters then it will be impossible to distinguish between them in the lower dimensional space. However, if the projection direction is slightly skewed and the two clusters do not fall directly on top of one another then the resulting data will have some degree of ambiguity because of cluster overlap but in principle it can still be modeled. In this case we would hope that the infinite Gaussian mixture model would capture and preserve this ambiguity in the posterior distribution over models.

In Figure 2.3 all 2D cardinal projections of the 4D synthetic data are shown. The true latent class cardinality is $K_+ = 6$ and the labels for all datapoints are illustrated by distinct color. In most of the projections some degree of ambiguity due to overlap has been introduced. In particular the subplot in the second row and second column shows a projection in which many of the classes are nearly indistinguishable. In this situation we would expect that the model would have a difficult time recovering the true number of latent classes. On the other hand, the subplot in the bottom row of the first column is a projection where all of the classes are nearly perfectly separated. Here we expect the model to cluster the data almost perfectly, despite operating in reduced dimensionality space.

Figure 2.4 shows the maximum a posteriori labeling of data as classified by a model estimated from that specific 2D projection of the data. The order in which projections were assigned to subplots is the same as in Figure 2.3 for ease of comparison. Here again the cluster labels are indicated by distinct color. In the bottom row, ostensibly the two least ambiguous projections, the MAP clustering from the model is nearly perfect both in assignment of data to classes and in latent class cardinality. It is the case that the class identifiers assigned to the clusters in these MAP samples are different than the class identifiers assigned in the ground truth labeling (notice that the colors assigned to individual clumps vary between the ground truth and the MAP sample). The IGMM posterior is invariant to permutations in the labeling of the clusters.

As the labeling in Figure 2.4 is the maximum a posteriori labeling it clearly only represents a small part of the modeling story. We noted that in the the original ground truth 2D projections (Figure 2.3) ambiguities arose as a function of clusters being projected on top of one another. Figure 2.5 illustrates how the model identifies and represents these ambiguities. Figure 2.5 was constructed by computing the entropy of the posterior conditional distribution over labels for each datapoint $\mathcal{H}[P(c_i = k|\mathcal{Y}, \ldots)]$. The color of each datapoint is assigned according to the average information that would be required to communicate the label of each datapoint under its conditional label distribution. This corresponds to

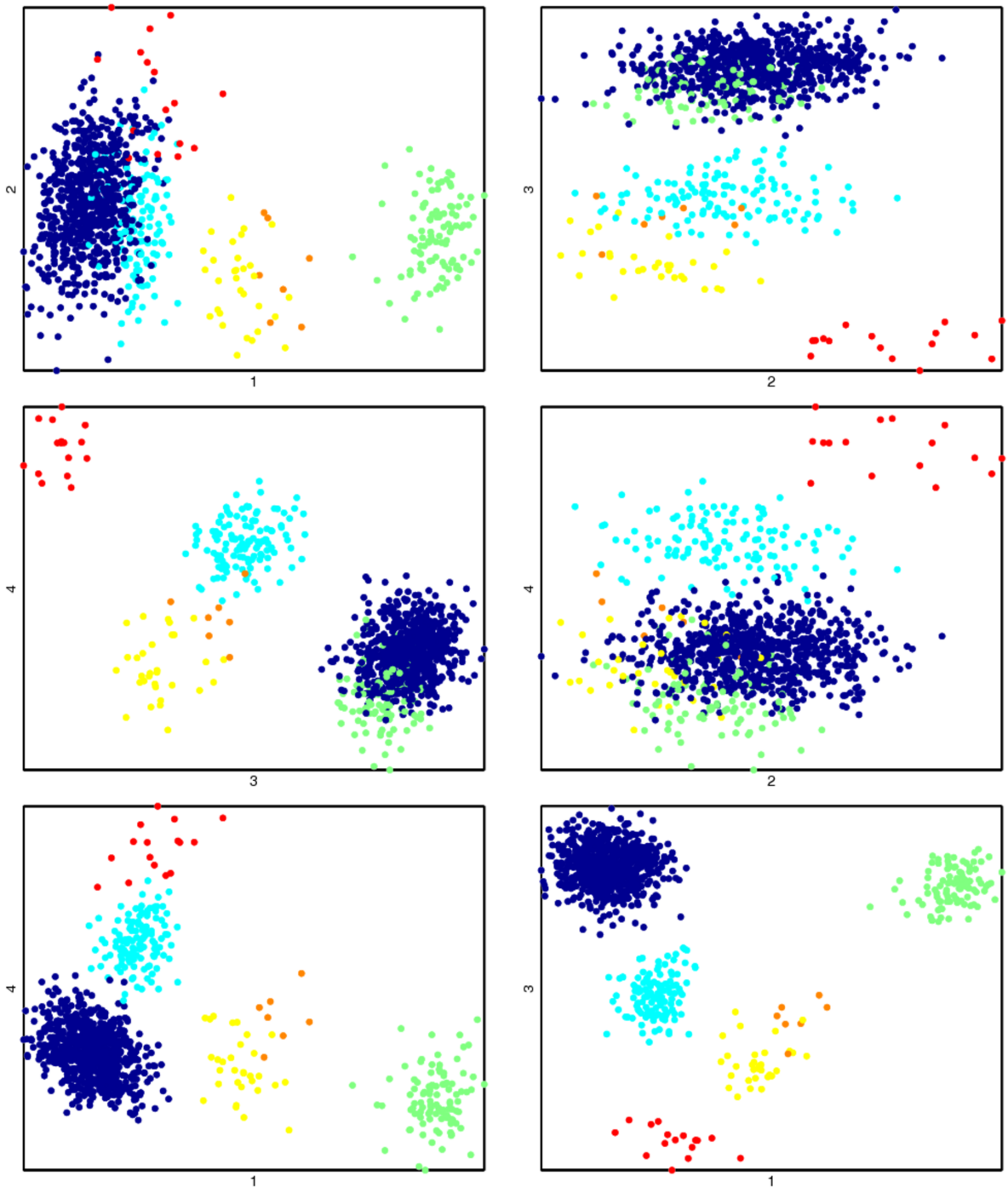

Figure 2.3: Synthetic 4D data: ground truth labeling; all 2D projections

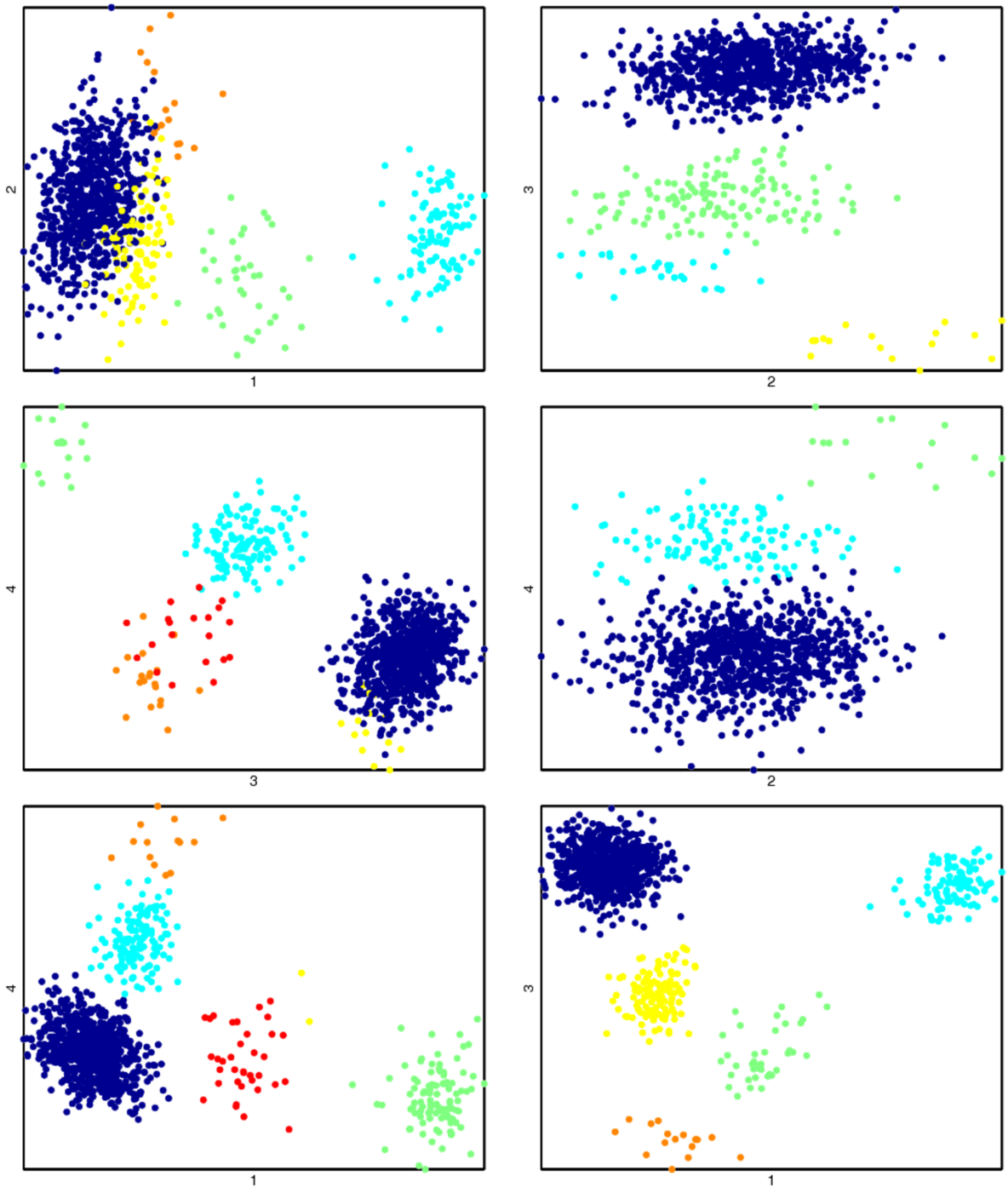

Figure 2.4: IGMM maximum a posteriori labeling of 2D projections of 4D synthetic data. One IGMM was trained for each panel of 2D data. The color of each dot indicates its class membership.

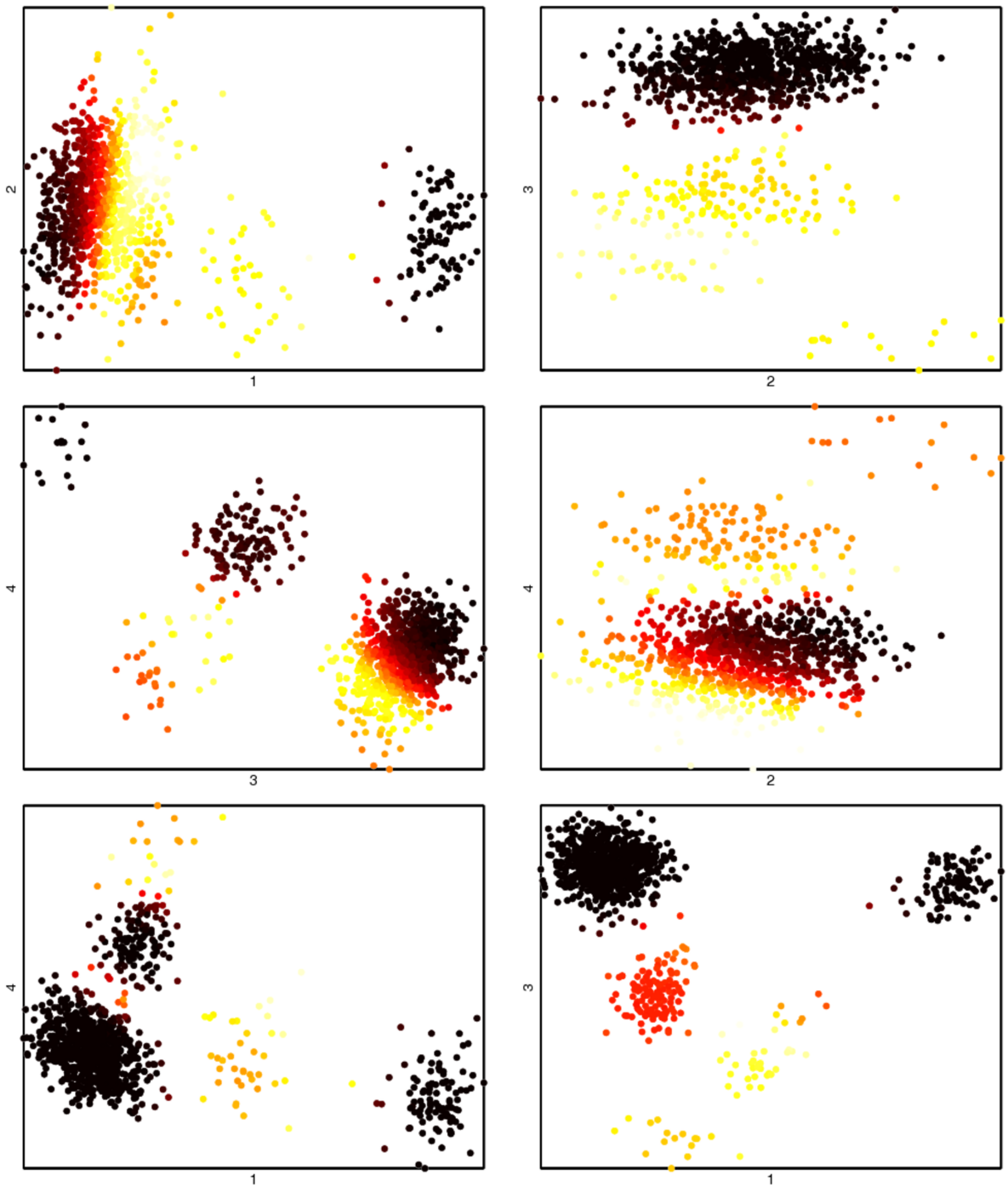

Figure 2.5: Entropy of class label marginal distribution for models estimated from 2D data in Fig 2.4. The color of each point corresponds to the entropy of the marginal distribution over its labeling. The "hotter" the color the more uncertain the label under the IGMM.

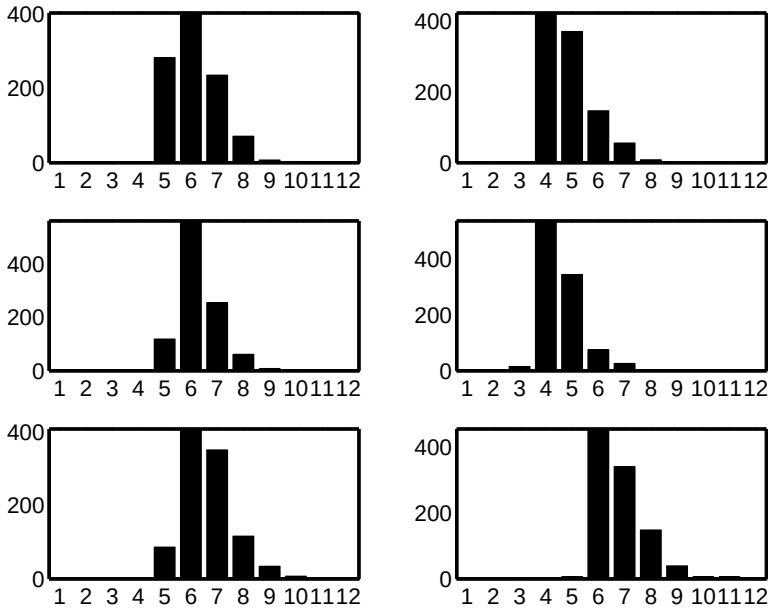

Figure 2.6: Latent class cardinality marginals from clustering all 2D projections of 4D data.

the "ambiguity" of the data as a label distribution with maximal entropy implies that the label is equi-likely to take on any one of a number of labels. Accordingly, the colors in this plot range from black which indicates little ambiguity through red, orange, and yellow which indicate increasing levels of ambiguity (more bits to convey the label). The colormaps assigned to each panel of this plot are different in that the colors should only be interpreted as a measure of ambiguity relative to the individual panel. Absolute bit scales are not given because, although they could be reported, they would not be particularly meaningful given that the purpose of this figure is only to illustrate which parts of the data the model identifies as ambiguous and to what relative extent.

What we observe in Figure 2.5 is that the model correctly identifies the ambiguous regions in the low dimensional data and that the datapoints that reside in those regions of ambiguity will be assigned to different latent classes frequently. This will be of consequence in subsequent inference tasks that utilize the latent variable distribution.

Also relevant to our analysis of the performance of the model in ambiguous data domains is the marginal distribution over class cardinality. In Figure 2.6 and Figure 2.7 the latent class cardinality marginal distribution is shown for all possible 2D projections and 3D projection clusterings of the same original 4D data. The subplots in Figure 2.6 are ordered in the same way as the corresponding plots of clusters; in Figure 2.7 all possible three dimensional projections of the data are shown. What the figures show is not unexpected;

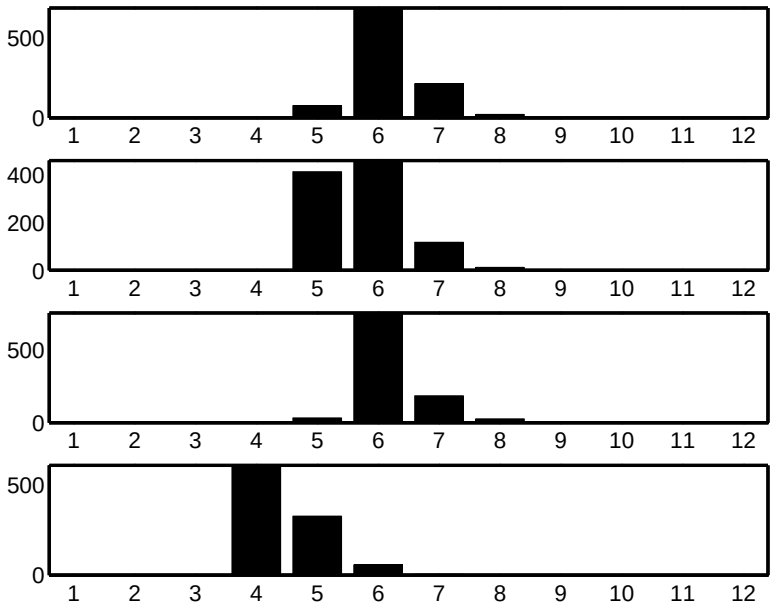

Figure 2.7: Latent class cardinality marginals from clustering all 3D projections of 4D data.

in projections that do not induce indistinguishable clusters the model produces a marginal over class cardinality that is peaked in the right place (around 6). Further, we observe that as the level of ambiguity is decreased (by adding a dimension) the distribution over latent class cardinality becomes more peaked, generally about the correct value. In the cases where the distribution is peaked in the wrong case (such as the last column of Figure 2.7 where the projection is unrecoverable in the same way as in the second column and third row of Figure 2.4), the projections can be observed to be degenerate in an unrecoverable way; however, the model still admits clusterings of the data with the correct number of latent classes.

2.3.3 Sensitivity analysis

In Bayesian modeling sensitivity analysis usually refers to testing how sensitive a statistical conclusion is to particular choices of priors. Here we investigate the sensitivity of the model to the parameter settings of the particular prior we described in Section 2.2. The way that we address this is by empirically investigating how much data is necessary to overcome an improper setting of the hyperparameters.

Here the size of the parameter spaces makes a complete investigation impossible. In order to fully investigate the sensitivity of the model with respect to hyperparameter settings we would need to explore a potentially high dimensional parameter space. For instance for

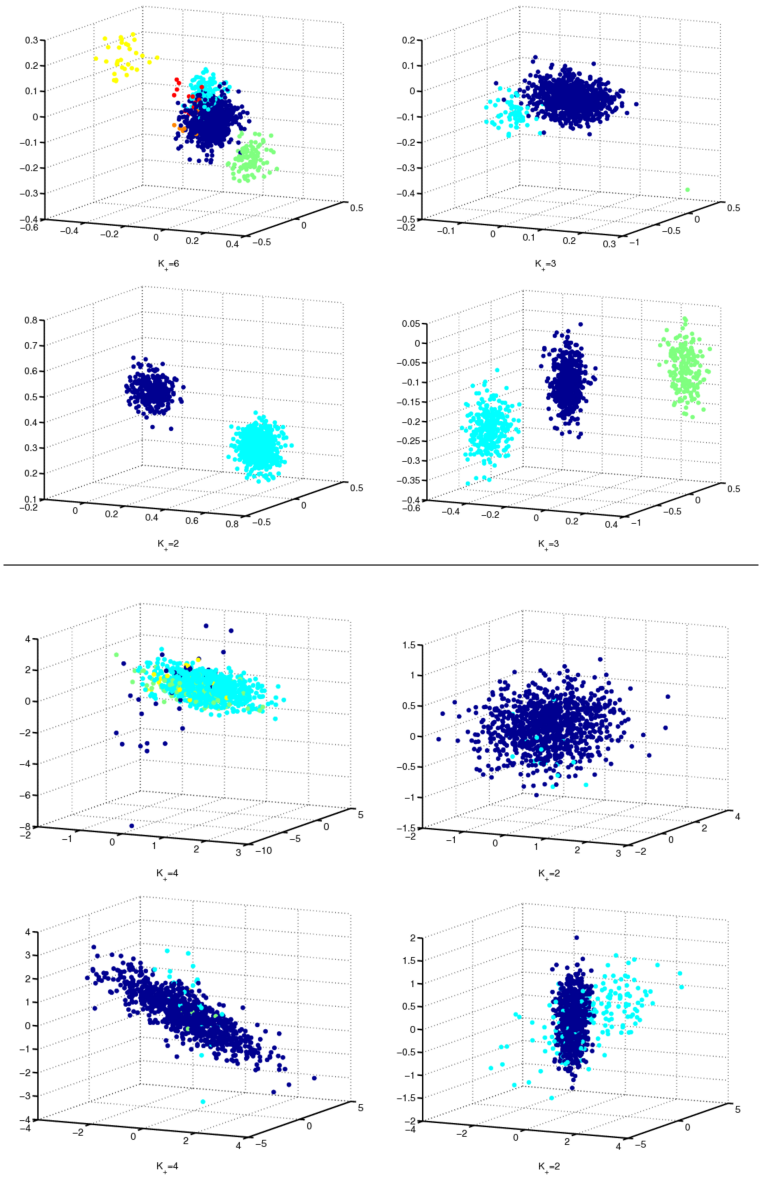

Figure 2.8: The upper leftmost figure is the 3D data (1000 points) used in producing Figure 2.9. The remaining three figures (each also 1000 points) above the horizontal line are additional datasets drawn from the same model (i.e., the same hyperparameter settings as the one in the upper left: $\mu_0 = [0\ 0\ 0], \kappa_0 = .05, \mathbf{\Lambda}_0^{-1} = \text{diag}(.1), \nu_0 = 50, \alpha = .4$). The figures below the horizontal line were generated (each also 1000 points) from an IGMM whose hyperparameters were set to the "wrong" values used for Gibbs and particle filter sampling in the generation of Figure 2.9 ($\mu_0 = [0\ 0\ 0], \kappa_0 = 10, \mathbf{\Lambda}_0^{-1} = \text{diag}(1), \nu_0 = 5, \alpha = .2$). These bottom figures illustrate the influence of the choice of hyperparameter settings, here the "wrong" choice leads to more broadly spread, overlapping clusters.

4 dimensional data the parameter space is 23 dimensional. This is clearly too complex a space to explore fully. So we do the simplest possible thing, namely, we set the values of the hyperparameters incorrectly to arbitrary, somewhat unreasonable values (values a complacent novice user might choose accidentally). These choices are illustrated in Figure 2.8. Figure 2.8 shows example synthetic datasets drawn from the IGMM generative model with two sets of hyperparameters. First, on the top half of the figure four draws from the generative model are shown with the same set of "ground truth" parameters. The data which we use to build a posterior distribution is shown in the upper, upper left. This data is characterized by, generally speaking, nicely separated clusters with approximately equal variance. The bottom half of the figure shows data generated from the IGMM generative model parameterized as the Gibbs sampler was parameterized. Here each class is more broadly spread, there are fewer classes on average (given the same number of datapoints), and the classes tend to overlap.

Figure 2.8 shows metrics of posterior distributions estimated from the data in the upper upper left of Figure 2.8 (training data). The sampler was initialized with parameters different than those used to generate the training data (values given in the caption of Figure 2.8). Two pivotal estimation quantities were varied; first the number of training observations and second the number of samples used to represent the posterior. This experiment was repeated for both the Gibbs sampler (right column) and the particle filter (left column). The variation of information developed by Meila [2002] is an information theoretic comparison of clusters invariant to cluster cardinality and label permutations. On the top row we observe that for both the particle filter and the Gibbs sampler both increasing the number of observations and increasing the number of samples improves the clustering. We computed the expected variation of information by averaging the variation of information between each sample from the posterior and the true clustering. The second and third rows show the expected and maximum a posteriori estimate of K_+ where $K_+^{\text{true}} = 6$ (shown in red). Both the Gibbs sampler and the particle filter produce posterior estimates that generally improve as the number of observations and samples increases.

Another aspect of the IGMM modeling one might wish to understand is how well the model does in fitting multiple sets of data all drawn from the same model. So far we have performed tests on single datasets where we vary the parameterization of the posterior estimation algorithm. In Figure 2.10 we show the difference between expected (left subfigure) and MAP (right subfigure) value of K_+ under the posterior distribution and the true value for estimates of multiple datasets. To generate this figure we generated a single dataset for pairs of random number generator seeds (Matlab's rand and randn). Each initialization of

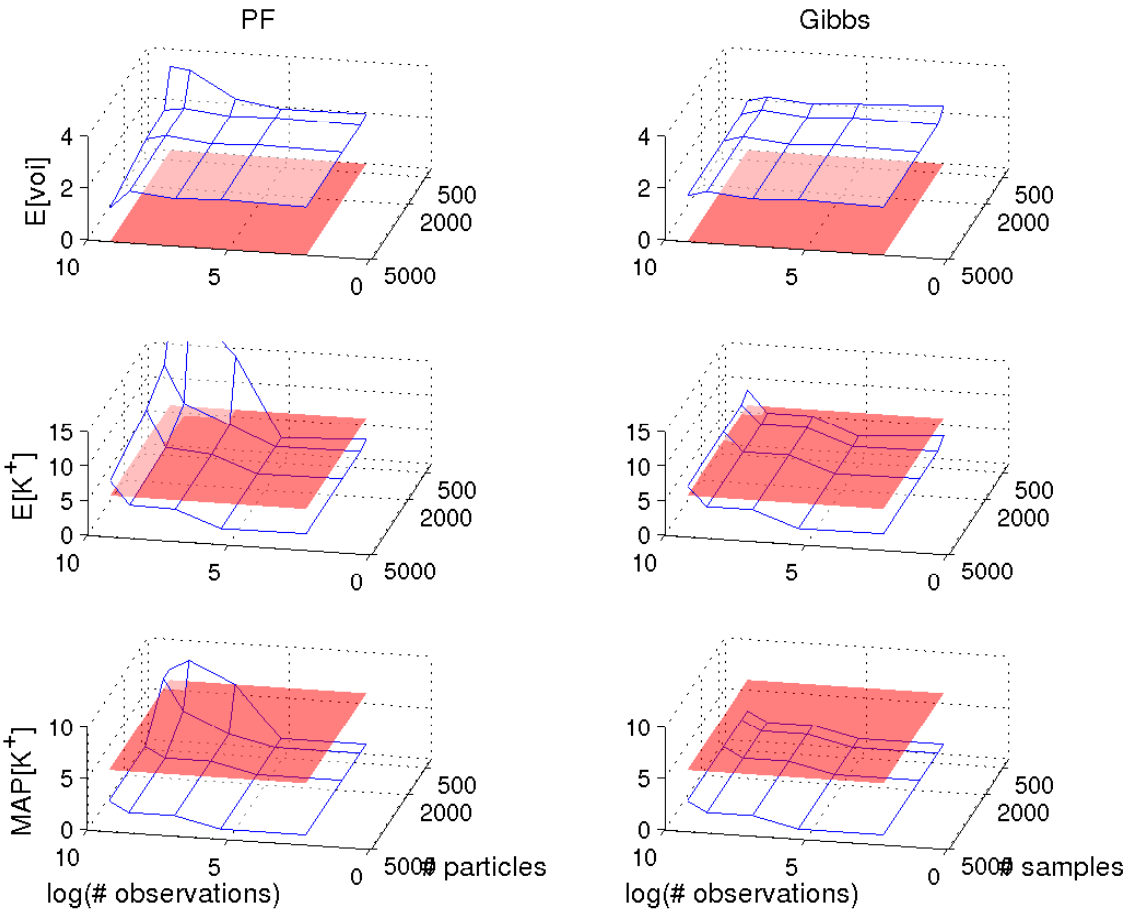

Figure 2.9: This figure illustrates the sensitivity of IGMM posterior estimation to hyperparameter choice. The right column shows results for the Gibbs sampler; the left the particle filter. Here we generated data from a model with hyperparameters $\mu_0 = [0\ 0\ 0], \kappa_0 = .05, \mathbf{\Lambda}_0^{-1} = \text{diag}(.1), \nu_0 = 50, \alpha = .4$ and the estimated a posterior by initializing the samplers with hyperparameters $\mu_0 = [0\ 0\ 0], \kappa_0 = 10, \mathbf{\Lambda}_0^{-1} = \text{diag}(1), \nu_0 = 5, \alpha = .2$. The effect of varying the number of samples/particles used in the posterior estimate and the amount of training data is shown. The top row shows a metric of cluster similarity while the bottom rows shows $E[K_+]$ and the maximum a posteriori value of K_+ for the estimated model.

the random number generators resulted in a different dataset. The horizontal axis in both subfigures is labeled with a trial identifier of which there were eleven; the verticle is the number of clusters in the resulting dataset. The number of clusters is implicitly determined by running the Chinese restaurant process as described in earlier sections. Each different rand seed could potentially have generated a unique number of clusters however in this case it did not. For this experiment we did not test the particle filter and accordingly only show results for the Gibbs sampler. What we can informally note from this figure is that the Gibbs sampler estimated posterior distribution slightly overestimates the number of latent classes in the data, but not always and usually not significantly.

2.4 Discussion

In this chapter we reviewed the infinite Gaussian mixture model by relating it to the finite mixture model and working through much of the math necessary to understand the nonparametric case. We reviewed two different posterior estimation methodologies then concluded the chapter by exploring characteristics of both the model and the estimation procedures. Missing from this chapter is any discussion of other posterior estimation approaches such as the variational approach developed by Blei and Jordan [2005] and the split-merge sampling algorithm developed by Jain and Neal [2004]. Secondarily no analysis of different priors was included, perhaps changing the multivariate normal inverse Wishart distribution with the multivariate Jeffreys. Additionally, sampling of α was only briefly mentioned however it is easy to do with either a Gibbs or Metropolis sampling step. In subsequent neural data analyses we do just that, but here we do not. Likewise nor do we consider estimating a distribution over the other hyperparameters.

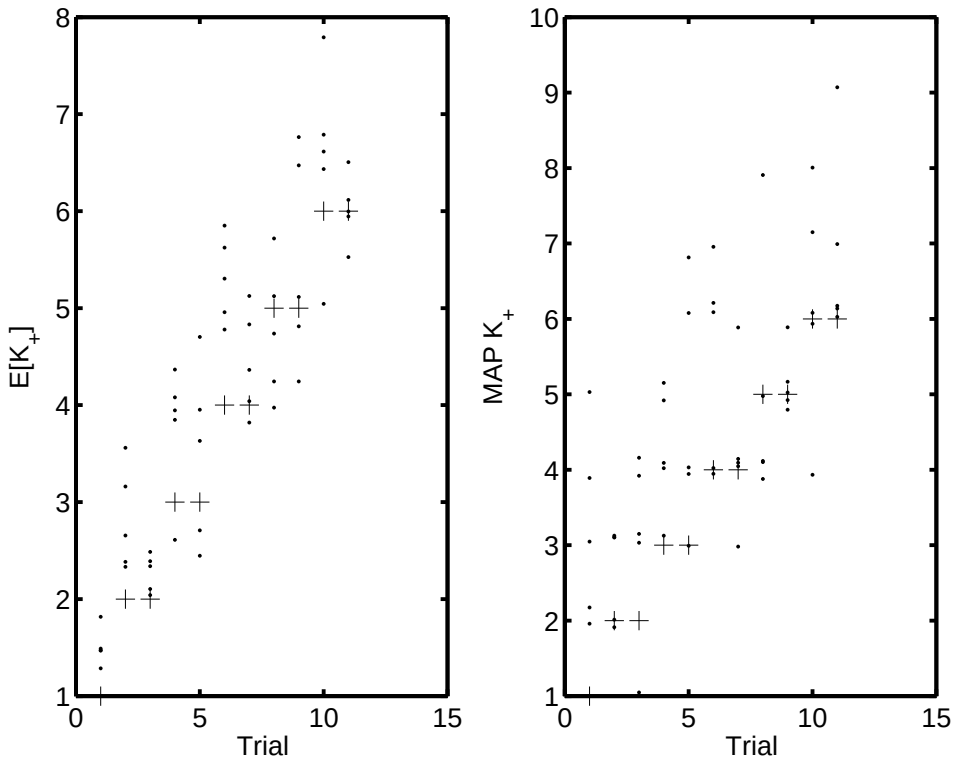

Figure 2.10: Performance of the IGMM with respect to latent class cardinalities. In this figure we show the difference between K_{true} and both the expected value of K_+ under the posterior and the maximum a posteriori value of K_+. The intial state of the pseudo random number generators was initialized differently for each x, y pair. Here the horizontal axis is the trial number (corresponding the the setting of one of the random number generator seeds) and the vertical is the resulting true value of K_+. For each trial there are multiple vertical points; each corresponds to a different dataset (initialization of of the other pseudorandom generator). In general we see that the model tends to slightly overestimate the number of classes both in expectation and in MAP. In the right subfigure normally distributed jitter was added to the vertical position of each point; otherwise all points would overlap on integer values.

Chapter 3

Application: Spike Sorting

3.1 Introduction

Labeled sequences of discrete action potential generation times are at the heart of many neural data inference tasks. Such "spike trains" are generated from electrophysiological recordings by a process called spike sorting. Spike sorting was introduced in Chapter 1. To expand on the introduction here, spike sorting is the task of detecting and labeling all action potentials that occur in a recording. The inferred labels identify from which neuron each spike came.

In this chapter a new approach to spike sorting is presented; differing from almost all prior art in that no single best spike train is sought. Instead a *distribution over spike trains is estimated* for the purposes of expressing uncertainties that arise during the spike sorting process in the outcome of neural data inference tasks. In this chapter the infinite Gaussian mixture model reviewed in the previous chapter is demonstrated in this capacity.

3.2 Motivation

This work is motivated by traditional neural data analyses that can be cast as operations on spike trains. Among many others these include hypothesis tests about single cells or population characteristics, and decoding from single or multiple neurons. Currently it is typical of such analyses to to assume that the spike trains on which they are based are unambiguously correct. In other words, any uncertainties that might have arisen during the conversion from analog voltage trace to discrete spike train are ignored. To combat this it has become common practice to report in published results spike sorting methodologies and signal to noise ratios for analyzed data. Unfortunately this in and of itself may not be

sufficient recourse as Harris et al. [2000] and Wood et al. [2004a] have both demonstrated that there is uncertainty that arises in spike sorting and in particular that expert spike sorters can and do produce spike trains that vary extensively. It is unfortunately also typical of traditional neural data analyses to ignore the ultimate effect of spike train variability on the outcome of spike train analyses as it is usually neither investigated nor reported.

In defense of experimental practice we recognize that in some recordings and for some types of recording technology there are situations in which it might be true that the data are unambiguous and the inferred spike trains accurately reflect the true activity of the recorded neuron(s). Additionally we do not believe that most conclusions wrought from traditional analyses of spike trains would be changed if some accounting scheme for spike sorting uncertainty were developed and retroactively applied. To the contrary, we submit that most will stand unchanged due to the tendency towards conservativeness that we have noted anecdotally amongst experimentalists: usually only the most unambiguous, noisefree data is admitted for analysis.

We also submit that a simple reason for the field having de facto accepted the assumption that the results of spike sorting are "correct" is the paucity of statistical tools that would allow one to avoid accepting that assumption, the work of Nguyen et al. [2003] being an exception. Without such tools it is unclear how to avoid accepting this assumption other than to have many individuals sort the same data and to do repeated analyses from all resulting spike trains. Unfortunately it isn't clear how exactly this would help as there is no principled way to average amongst the various human interpretations of the data. A second reason for tolerating the assumption may be that there is typically more data than can be practically analyzed so searching for and only analyzing spike trains that seem to be mostly unambiguous may be more practical than searching for ways to account for uncertainties that might arise in spike sorting when ambiguous data could just as well be discarded.

That aside, there may be situations in which either discarding data is undesirable or impractical. A clear example of this is chronically implanted human and primate subjects where one might expect the implanted recording device to work for years without post-implant adjustment. This means that in such subjects it may be undesireable or impractical to adjust the position of the recording device or to reimplant the subject. The nature of chronic recording setups leaves them unusually susceptable to changes or degradation of the signal over time. In cases like this in may be of utmost importance to make the best of the available signal, even if it is ambiguous.

Even in situations where it might be practical to seek out unambiguous data, there is evidence that neural data that appears to be unambiguous may in actuality less so than

appearances would suggest. In Harris et al. [2000] it is shown that there is a persistent absolute error rate of around 10% in sorting data recorded from tetrodes and worse for data from single electrodes. In their study ground truth neural activity was determined by recording the activity of a neuron intracellularly. At the same time they recorded the activity of the cell extra-cellularly as well. This allowed them to report the error rate with respect to the ground truth. This work inspired a similar study by Wood et al. [2004a] for data recorded from microelectrode arrays. Unfortunately the physical characteristics of microelectrode arrays prohibit measurement of the ground truth so only subjective variability results could be reported. The results from this study showed high variability between expert sorters and are reviewed in the next section. Finally the recent work of Blanche et al. [2005] suggests that data recorded using tetrodes is more ambiguous than is widely thought. In this work data was recorded using a new microelectrode array consisting of fifty densely spaced recording sites. The recorded data was then sorted in multiple ways. One way used information from all fifty recording sites. In the other ways "virtual" tetrodes were constructed by selecting subgroups of four recording sites out of the fifty. The original data was then sorted using information from only the four recording sites in the virtual tetrode. By doing this it was clearly demonstrated that neurons that were indistinguishable given only features from four recording sites became distinct when information from additional recording sites was utilized.

Unfortunately in none of these studies were the effects of the spike train variability on subsequent analyses analyzed. This raises the question; how sensitive are neural data analyses based on spike trains to variation in the given spike trains? Here we first review evidence of spike train variability in human intepretations of microelectrode array data before illustrating the effect of this variability on neural decoding.

3.3 Evidence of uncertainty in spike sorting data

Expert spike sorters are known to produce different spike trains given the same data; a fact that has been demonstrated for data recorded from multiple types of recording technology.

3.3.1 Human interpretations of spike sorting data

In our own study of human spike sorting performance (Wood et al. [2004a]), we quantitatively assessed the inherent ambiguity of spike sorting data recorded from a multi-electrode array (CKI) by asking five expert subjects to sort a set of waveforms recorded from the arm area of primary motor cortex in two different monkeys. Using this data we computed the

subjective variability of spike trains produced by different subjects. In our view this subjective variability is a proxy for inherent ambiguity in the spike sorting data and provides a measure of the spike train variability we should anticipate in subsequent analyses. With data recorded from this type of micro-electrode array however there is no way to establish "ground truth" and consequently no way to quantitatively measure the error in human sorting performance. To address this, we generated a set of synthetic channels containing realistic action potential waveforms and asked the same subjects to sort them as well. The synthetic channels were designed to be indistinguishable from the real and allowed us to establish quantitative error rates with realistic waveforms.

The five expert subjects were graduate students, research assistants, or postdoctoral researchers from the same laboratory who all had significant experience in sorting neural recordings. The subjects sorted twenty independent channels using Plexon's offline spike sorter [Plexon Inc.], labeling units and waveforms as they would for their own research and were given as much time as they liked to sort the data. Plexon's software provides users with various tools to sort waveforms one channel at a time. A commonly used approach is manual cluster selection and refinement in a graphical display constructed by projecting the waveforms onto subsets of the principal components.

3.3.2 Variability of spike trains produced by manual sorting.

Figure 3.1 illustrates the kind of subjective variability we observed between subjects sorting the same real data. The figure shows the sorting results for two subjects (D and E), where for simplicity only a subset of the waveforms from one of the real channels in the dataset is shown. The green waveforms indicate agreement between the subjects on the presence of a single neuron (in green). The subjects disagree however on the number of units present with subject E hypothesizing two additional neurons in the data. These results are typical of what we observed throughout the study.

Figure 3.1 provides another view of the same dataset. Here a two-second segment of the data is viewed as a spike train. Each vertical line corresponds to a spike while the color corresponds to the classification in Figure 3.1. Black lines correspond to waveforms that were deemed to ambiguous to sort. Note the level of disagreement in the classified spike trains. Even for the green spikes which have similar waveforms, the two subjects included very different numbers of spikes. We will show in Section 3.4 (Table 3.2) that spike train variability affects decoding performance. We further posit that differences such as these as this might lead to variations in conclusions drawn from other neural data analyses such as

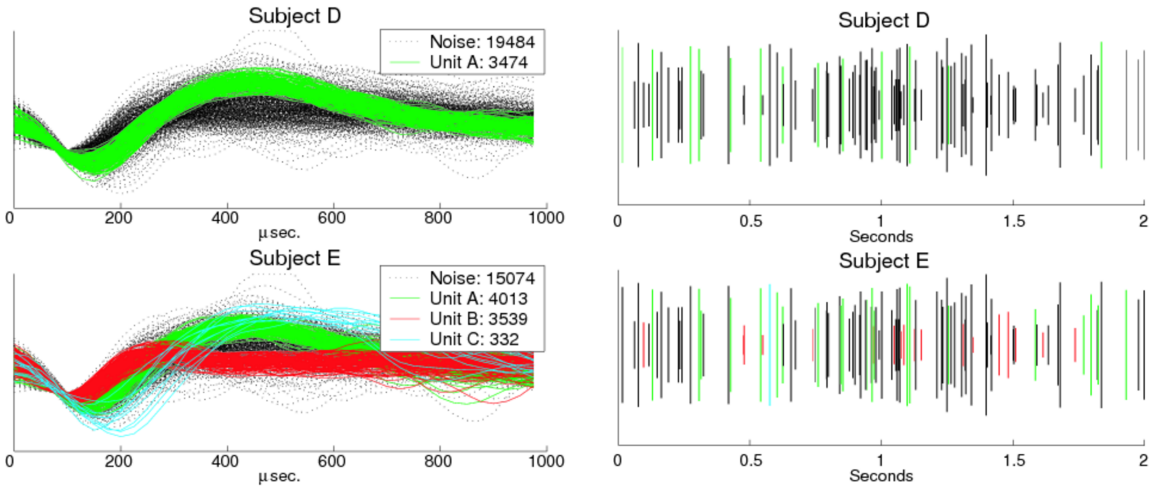

Figure 3.1: Manual classification of the same (real) waveforms by two different subjects. The waveforms on the left are shown twice, once for each subject and are colored to reflect each subjects' spike sorting choices. Subject D identified one neuron in the recording and attributed 3,474 action potentials to it. Subject E identified three neurons with the green labeled neuron being the same as that identified by subject D, only here the number of action potentials attributed to it was 4,013. The colors of the spike trains on the right correspond to the labeling of the waveforms on the left and the height of the lines corresponds to the peak to trough height of the action potential. All waveforms and spikes colored black were deemed by the experts to be too ambiguous to sort.

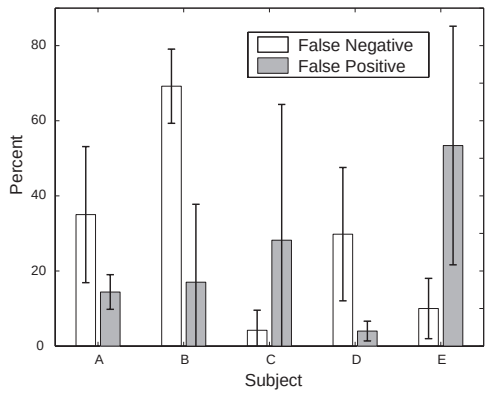

Figure 3.2: Mean ± std. dev. false positive (FP) and false negative (FN) error rates over all synthetic channels per subject. FP's occurred when a subject inappropriately counted noise waveforms as having come from the single generating unit FN's occurred when a subject miss-classified a true spike as noise.

modeling of neural encoding or detection of excess synchrony.

Table 3.1 shows the number of units and number of spikes detected by each of the subjects. The subjects agreed on the number of units in the real channels only 25% of the time, and most of these consensus channels either contained no neural activity or were extremely well isolated. The number of units detected varied by roughly a factor of two (from a low of 18 to a high of 35) while the total number of spikes varied even more, with subject E finding approximately four times as many as subject B. Even when the subjects agreed on the number of units in a given channel, quite often they disagreed about how many spikes each generated.

Variability of manually sorting synthetic data

Because it is impossible to know the "ground truth" for data recorded from chronically implanted micro-electode arrays we also conducted a study where we asked the same human subjects to sort 5 synthetic channels, each consisting of one "true" neuron plus noise characteristic of microelectrode array data. These synthetic channels were constructed using a multivariate Gaussian generative model for waveforms arising from a single neuron and an empirical process for generating irrelvant confounding waveforms. One well isolated neuron from the twenty channels was chosen to train the generative model. The confounding waveforms were generated by drawing waveforms randomly from every channel except the one used to train the generative model, excluding all waveforms that had been classified as spikes in the any sorting of the 20 channels.

It is common to use a plot of the inter-spike intervals (ISI) when performing spike sorting as violations of the absolute refractory period indicate misclassification. Consequently, we assigned timestamps to the non-confounding synthetic waveforms by sequentially drawing inter-spike intervals from an empirical ISI distribution. Additionally, for realism, the synthetic channels needed to exhibit the same recording artifacts the real channels exhibited. The real data was captured when crossing upwards through a voltage threshold, and, in the real channels all the captured waveforms were aligned on the threshold crossing. To replicate this in the synthetic data we aligned both the synthetically generated waveforms and the sampled noise waveforms using an appropriate threshold.

Figure 3.2 shows how the subjects performed on the task of spike sorting the synthetic channels. On average the subjects had overall 23% FP and 30% FN error rates for the synthetic channels. The data illustrated in Figure 3.2 also suggests that the subjects employed individual sorting strategies; this phenomenon was also observed by Harris et al.

Subject	A	B	C	D	E
Spikes	99160	50796	150917	77194	202351
Units	28	32	27	18	35

Table 3.1: Totals for each subject, all real channels combined. Spike counts include all identified spikes; unit counts include all neurons each subject found.

[2000]. Despite large variability, it seems that subjects A,B, and D used a sorting strategy that consistently worked to minimize false positives while subjects C and E chose one that worked to minimize false negatives. It also suggests that it might not be possible to overcome the trade-off between false positives (FP) and false negatives (FN) using the sorting tools employed. This might be due to inherent similarities between spike shapes or between spike and confounding waveforms. It is also possible that the tools our subjects employed restricted them from being able to definitively segment and classify the activities of individual neurons; for example, the software only allows users to view 2D projections onto the principal components.

Anecdotally, one of the individuals who served as a subject in this study was the same person who provided the training data (sorted real data) for the synthetic data. This person unknowningly sorted the same channels twice, once to provide the training data for synthetic data generation and then once a month later as a subject in this study. This individual determined that the real channels contained 12% more neurons the second time around (25 $\rightarrow$ 28) yet classified 8% fewer of the waveforms as neural activity (108073 $\rightarrow$ 99160 spikes). Although this demonstrates the kind of subjective variability we found, it also affects the analysis of our results for synthetic data. While the reported subject-to-subject variability for synthetic data would remain unaffected, the FN and FP rates we reported for synthetic data could vary depending on the way the training channels were initially sorted by our expert.

3.4 Effect of spike train variation on inference from neural data

Unfortunately, the effect of spike train variability on the outcomes of spike train analyses is not well studied, though it is mentioned as a significant concern for the field in a review by Brown et al. [2004]. In this section we review our finding that spike train variability affects decoding performance [Wood et al., 2004a]. We assert that will it affect other analyses as

well; the extent to which we do not yet know.

In Section 3.3 we found that expert human sorters produced different spike trains given the same recording, however we did not show how this affected the results of analyses based on spike trains, decoding being one such analysis. By asking volunteers to sort another dataset, this one with associated kinematics information, we were able to ascertain how subjective variability in spike trains affects decoding results.

Decoding arm position from M1

That motor cortical neural spiking activity is correlated to movement has been known for a very long time. In Georgopoulos et al. [1982] it was established that arm coding neurons in primate motor cortex modulate their firing rate as a function of hand movemement direction. In particular it was established that each neuron has a "preferred direction" of movement which causes the neuron to fire most rapidly. Movement in directions out of phase with that preferred direction were found to cause less rapid firing. As a follow on to this work it was established in Georgopoulos et al. [1986] that hand movement direction could be "decoded" from the firing activity of a population of neurons whose preferred directions were known. Subsequent to this study many algorithmic improvements to neural decoding have been made; one of the more notable was the adoption of the Kalman filter [Kalman, 1960] as a neural decoding algorithm. In this study we utilize the Kalman filter to decode arm position from the firing activity of cells in primate motor cortex.

To investigate the effect of spike train variability on decoder output we used the Kalman filter decoding method described in Wu et al. [2003] to decode arm position from spike trains generated from multiple channels of motor cortical neural data. The Kalman filter was trained on 3 minutes of rate and kinematic data binned in 70 millisecond time bins. Variations in the input spike trains manifest themselves as variations in the estimated rates. Average decoding results are given for 5 independent 1 minute data segments. The kinematics were encoded as a 6 attribute vector comprised of hand x, y position, velocity, and acceleration. Our implementation differed from Wu et al. [2003] in that we found that the optimal lag between rate and kinematics for decoding was 0 milliseconds instead of 140 milliseconds. Rate was computed by counting the number of spikes that occured between the time at the beginning and end of each of the bins.

The data used in this study was from a single monkey in which, following task training, a Bionic Technologies LLC (BTL) 100-electrode silicon array (CKI) was implanted in the arm area of primary motor cortex (M1). The recording setup was similar to that used in

Serruya et al. [2002]. where an animal was trained to use a two-joint planar manipulandum to control the motion of a feedback cursor on a computer screen. The simultaneous recording of hand kinematics and neural activity allowed the study of motor cortical encoding of hand motion and the training of decoding methods. A recording from this animal performing a "pinball" tracking task (Wu et al. [2003]) was used in this study.

The recording used for this study was a collection of 96 independent channels where the waveform capture threshold for each channel was empirically determined at the time of recording, and was set low enough to ensure that a majority of the spiking activity was captured. The array was placed in M1, but by virtue of array design and insertion technique the location of each electrode with respect to individual neurons was unknown. Correspondingly, every channel may have had waveforms from multiple neurons. Additionally, on each channel some number of threshold-crossing waveforms consisting of surrounding neural activity that was too ambiguous to sort were recorded. A single contiguous 600 second segment of data was extracted from a part of the recording featuring high arm movement and neural firing rates. Unlike other studies where decoding is done only on data recorded between start and stop cues, we extracted a single continuous segment where there were periods of no arm movement.

Four subjects, all graduate students or postdoctoral researchers and all expert spike sorters, were given this recording and asked to sort it using any tool at their disposal. They were instructed to sort it in the way they thought would maximize decoding performance. As in the previous section, all subjects used Plexon's Offline sorter software (Plexon Inc.) to sort the dataset. Units were identified and spikes were assigned to them by manually cluster cutting waveforms projected into various two dimensional PCA subspaces. The resulting spike trains were decoded and tabular results are displayed as A,B,C, and D in Table 3.2.

To better understand exactly how spike sorting variability impacts decoding performance we also decoded spike trains produced by both randomly sorting and not sorting at all. No sorting ('None' in Table 3.2) means that all waveforms on each channel were attributed to a single neuron. Randomly sorting ('Random' in Table 3.2) means that on each channel three neurons were posited and waveforms were randomly attributed to them with equal probability.

Variation in decoding results

Table 3.2 summarizes the decoding results. In the table 'Ave. Human' is the average of subjects 'A', 'B', 'C', and 'D'. Reported are the total number of neurons and spikes identified,

Subject	Neurons	Spikes	MSE (cm^2)
A	107	757674	11.45 ± 1.39
B	96	335656	16.16 ± 2.38
C	78	456221	13.37 ± 1.52
D	88	642422	12.37 ± 1.22
Ave. Human	92	547993	13.46 ± 2.54
Random	288	860261	13.28 ± 1.54
None	96	860261	12.78 ± 1.89

Table 3.2: Kalman filter decoding results for rates estimated from different spike trains. Shown here are average Kalman filter decoding results for 5 independent 1 minute spike trains as sorted by 4 human subjects and 2 control algorithms ('None' and 'Random') Note that the average human decoding results are worse than those for both not sorting and sorting randomly.

the correlation coefficients between the decoded and the true x and y hand position, and the mean square error (MSE) between the same. These results not only confirm the variability of spike sorting results demonstrated in Section 3.3 but also show that decoding performance is affected by spike sorting, sometimes significantly.

3.5 Related work

Having established the fact that spike sorting often involves dealing with ambiguous data and that spike train variability resulting from that ambiguity does affect neural decoding results we now suggest a solution. Our proposed nonparametric Bayesian approach differs from what is typical of automated spike sorting work. As stated in the introduction to this chapter our aim is not strictly speaking a new and improved automated spike sorter like those proposed in Shoham et al. [2003]; Wood et al. [2004b]; Hulata et al. [2002]; Lewicki [1998]; Takahashi et al. [2003]; or Sahani et al. [1998]. Those all aim to produce a single "best" spike train given a recording. Instead we embrace the uncertainties illustrated in the previous sections as inherent to the spike sorting problem and develop a probabilistic model of spike trains that allows us to represent this uncertainty and automatically account for its effect on spike train analyses. Such an approach allows expression of the level of confidence one came have in neural data analysis outcomes with respect to spike sorting variability. This approach requires a slight but significant rephrasing of our aim: instead of seeking the best spike train upon which to perform some analysis we now a seek a spike train model in which to do inference.

In Nguyen et al. [2003] the foundation for this approach was laid. They took the Bayesian

approach of Richardson and Green [1997] in analyzing mixtures with an unknown number of components and applied it in the context of Gaussian mixture model spike sorting. A summary of this approach is that a posterior distribution over spike trains was estimated using reversible jump Markov chain Monte Carlo (RJMCMC). This is quite similar in many respects to the approach we take in our work; however, they refrained from utilizing the full posterior distribution in subsequent spike train analyses, opting instead to consider only the maximum a posteriori sample from their model. In effect this means that their adoption of the techiniques of Richardson and Green [1997] accomplished little more than placing a complex but reasonable prior on the Gaussian mixture model and doing maximum a posteriori estimation and automatic model complexity selection.

In this section we apply nonparametric Bayesian estimation and inference techniques to the problem of spike sorting. We develop a spike sorting approach based on the infinite Gaussian mixture model we reviewed in the previous chapter. Like Nguyen et al. [2003] we demonstrate that our approach is a practical improvement over the a maximum likelihood approach to finite Gaussian mixture spike sorting. However we go beyond Nguyen et al. [2003] and show how to to express spike train uncertainty at the level of inference from spike trains. We argue the benefits of doing so and illustrate the effects on simple example spike train analyses.

Like Nguyen et al. [2003] and many of the more traditional spike sorting papers we focus on a very restricted subset of the overall spike sorting problem. Our focus is on the process of determining from which neuron out of an unknown number of neurons each action potential arose based on action potential waveshape alone. In doing so we overlook the equivalently difficult problem of spike detection and instead refer readers to Radons et al. [1994] for treatment of that topic. Additionally we will ignore the problem of detecting overlapping spikes (deconvolving coincident action potentials) here refering readers to the work of Fee et al. [1996] and Görür et al. [2004].

In our restricted view of spike sorting we assume that the statistics of action potential waveforms generated by neurons are sufficient to distinguish between neurons. We also assume that timing information is uneccesary to distiguish between cells. Thus our data consists solely of a large number of waveforms generated by an unknown number of cells. These assumptions are clearly not true in reality, but we adopt them for purposes of exposition and model simplicity.

We have established that we will use an infinite Gaussian mixture model to do spike sorting. In Chapter 2 we reviewed IGMM's in detail. Here we provide an overview of the translation of the spike sorting problem into the notation of the IGMM.

Suppose that we have N action potential waveforms from a single channel neurophysiological recording $R = [\vec{t}_1, \ldots, \vec{t}_N]$, where each snippet is a vector of n voltage samples $\vec{t}_i = [t_i^1, \ldots, t_i^n]^T \in \mathbb{R}^n$. For single electrode recordings $n = 40$. Our goal is to build a posterior distribution over sortings of this recording; simultaneously estimating how many neurons are present and which waveforms came from which neurons.

Instead of clustering in such high dimensional spaces, we use a reduced dimensionality representation of the waveforms, where each $\vec{t}_i$ is represented in a lower dimensional basis obtained via principal component analysis (PCA) such that $\vec{t}_i \approx \sum_{d=1}^{D} y_i^d \vec{u}_d$. Here $\vec{u}_d$ is the d^{th} PCA basis vector, and the y_i^d are the linear coefficients. Our spike sorting algorithm clusters the low dimensionality representation of the waveforms $\mathcal{Y} = [\vec{y}_i, \ldots, \vec{y}_N]$ rather than the full waveforms, so, for the remainder of this paper, when we write "event", "spike", or "waveform" it should be read as shorthand for "low dimensionality waveform representation".

We follow Lewicki [1998] in making the common assumption that the distribution of waveforms from a single neuron is well approximated by a multivariate Gaussian. Under this assumption our choice of an (infinite) Gaussian mixture model is natural.

To express the spike sorting modeling problem in the notation in Chapter 2 we do the following. The class indicator variables in the IGMM indicate from which neuron each waveform arose (i.e. $c_i = k$ means that the i^{th} waveform came from neuron k). Recapitulating the generative view of the finite mixture model in the spike sorting setting helps build intuition for how the model represents the data. To generate a set of action potential waveforms from a finite Gaussian mixture model first the number of neurons K must be chosen, as too the relative firing rates $\vec{\pi}$ of the neurons, the number of spikes N, and the statistical characteristics of the waveforms originating from each neuron, θ_k. Once these choices are made then N class labels c_i are generated from a multinomial parameterized by $\vec{\pi}$, and finally N waveforms each from the corresponding multivariate normal distribution with parameters $\vec{\mu}_{c_i}, \mathbf{\Sigma}_{c_i}$.

The Bayesian extension of this finite mixture model spike sorting approach follows along the lines of Chapter 2 as well. Here the Dirichlet prior encodes prior knowledge about the number and relative prevalence of neurons and their firing rates. The typical covariance structure observed for waveforms discharged from neurons can be encoded using the priors over the mixture densities' shape and position $(\mu_0, \kappa_0, \Lambda_0, \nu_0)$. Lastly, the nonparametric assumption behind the IGMM corresponds to assuming that there are is an infinitely diverse set of neurons of which only a finite but unknown number are present in any recording.

3.6 Inference using the NPB spike train model

Having reviewed the IGMM in the previous chapter and having provided justification for its adoption as a spike train model in this chapter, it remains to demonstrate its utility in performing neural data analyses.

As a reminder we are not particularly interested in the sorting results per se nor the resulting spike trains; although clearly both must be reasonable in order for us to be successful. Instead we are interested in demonstrating improvements to spike train analyses that result from accounting for spike train variability arising from neural data uncertainty. Here we perform two analyses of each of two motor cortical recordings. First we show how to use the NPB spike train model to address a hypothesis about low level physiological organzation, namely that the preferred direction of motor cortical neurons forms the basis of a fine grained motor cortical somatotopy. Our aim with this example is to show how the IGMM posterior distribution over spike trains allows us to utilize data that might otherwise not be utilized and to express uncertainty in our conclusions. Second, as we have corresponding behavioural variables (arm position) for these recordings we demonstrate neural decoding utilizing the NPB spike train model.

We start by describing the motor cortical datasets. Two different primate motor cortical datasets are used in this work. Both were recorded from the same monkey but on two different dates over a year apart. The tasks performed by the monkey on each occasion were different; the first was a "pursuit tracking" task and the second was a "pinball" task. In the discussion and results these datasets will be distinguished by task name.

As described by Wu et al. [2005] and Serruya et al. [2003] both the pursuit tracking task and the pinball task are voluntary two dimensional arm control tasks. In the pursuit tracking task the objective is to track a continuously moving target by two dimensional dextrous manipulation of a manipulandum. Both the target and a cursor representing the current manipulandum location are shown to an a monkey. In the pinball task the task objective is to acquire (move a cursor onto) a sequence of targets that appear sequentially at random points in the space reachable by the manipulandum. This can be seen as a generalization of the center-out reaching task commonly utilized in the motor control literature. The pursuit tracking dataset consists of ten trials each containing approximately five to ten seconds of movement. The pinball task dataset consists of fifty trials each consisting of approximately the same movement duration.

For both tasks the firing activity of a population of cells was recorded from a chronically implanted microelectrode array manufactured by CKI of which six were selected on the basis

of having good task-related tuning. The interelectrode spacing in these microelectrode arrays is large enough that each recording site on the electrode is independent. This means that the microelectrode array data can be thought of as twelve independent single electrodes. Threshold crossing events (waveforms) were saved on all channels along with the time at which they occured. The thresholds were empirically set by the experimenter at the time of recording to exclude non-neural waveforms from being captured. Following application of the waveform dimensionality reduction procedure given above to each channel individiually, the waveforms from the the pursuit tracking task were projected onto the first two PCA basis vectors accounting for on average 66% of the waveform variance (51%, 87%, 69%, 78%, 58%, and 55% per channel respectively, all rounded to the nearest percent). The waveforms for the pinball data were projected onto the first three PCA basis vectors accounting for on average 56% of the total waveform variance (59%, 72%, 52%, 54%, 54%, and 47% per channel)

Both datasets were sorted three ways. One way was expectation maximization with finite Gaussian mixture models and Bayesian information criterium model selection. We will refer to this as the maximum likelihood approach. The other two ways were infinite Gaussian mixture modeling; both Gibbs sampling and particle filtering posterior estimation. The hyperparameters chosen for the IGMM were the same for both datasets, $\mu_0 = [0\ 0], \kappa_0 = .2, \Lambda_0 = \text{diag}(.1),$ and $\nu_0 = 20$ (for the pinball data μ_0 and Λ_0 were appropriately sized for three dimensional observations). Sorting results for the the pursuit tracking data are shown in Figure 3.3 and for the pinball data in Figure 3.5.

Each of these figures is divided into six sections by thin black lines. We refer to these sections as panels. The upper left panel in both figures shows the six selected channels before spike sorting. Each dot is the projection of one waveform. The upper right panel shows a manual labeling of the data. In this panel the color black is significant as it indicates waveforms that were identified by the human sorter as being too ambiguous to sort. The first panel in the second row is the expectation maximization penalized maximum likelihood solution. The remaining three panels illustrate the posterior distribution over labelings induced by the IGMM. Each of these three panels is one sample from the IGMM posterior. Looking closely one can see that many of the individual waveform labels change from one sample to another as both new classes are introduced and destroyed and individual waveforms are attributed alternatively to one of a number of neurons.

As ascertaining how uncertainty is represented by this model we also provide a plot that illustrates this in Figures 3.4 and 3.6. Both figures redisplay the unsorted (unlabeled) data for ease of reference along with a plot that illustrates the relative (to the dataset) uncertainty

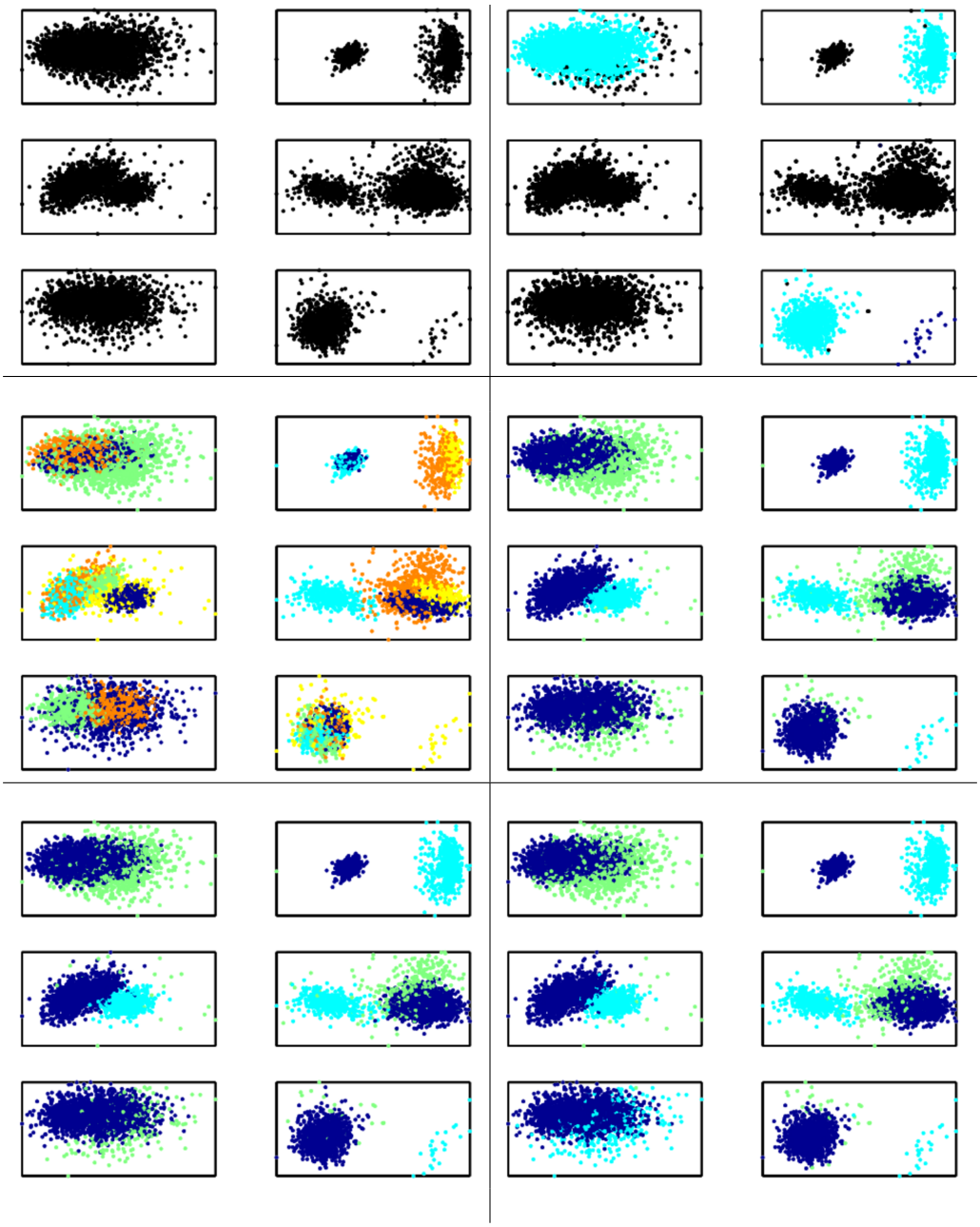

Figure 3.3: Results from sorting six channels of the pursuit tracking neural data using infinite Gaussian mixture modeling (IGMM). PCA projections of waveforms from channels 1-6 are shown in each of the six panels. The top left panel shows the unsorted waveforms, the top right shows a manual labeling (the black points are not assigned to any neuron), the first panel in the second row shows a maximum likelihood labeling, and the remaining three are samples from the IGMM posterior.

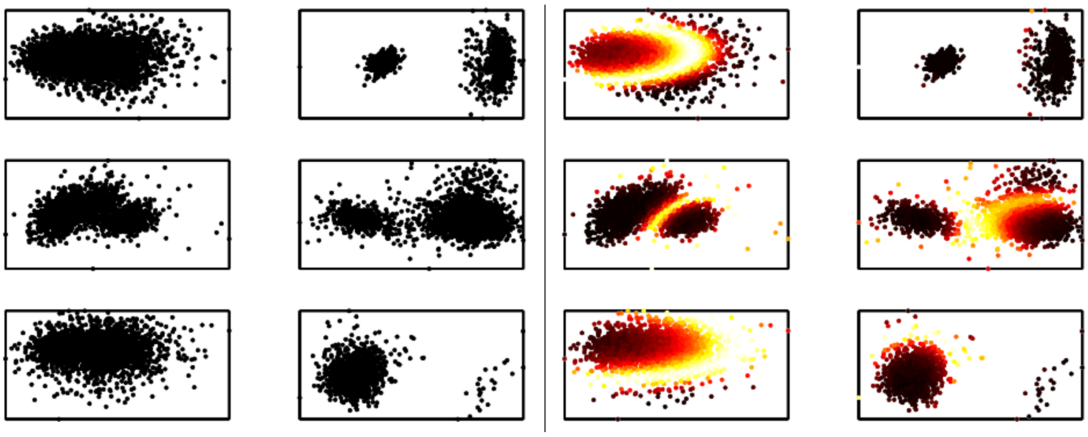

Figure 3.4: Results from sorting six channels of the pursuit tracking neural data using infinite Gaussian mixture modeling (IGMM). Channels 1-6 are shown in both panels. On the left is the unsorted data, on the right the same points are plotted but with colors that indicate how uncertain the cluster label for that point is (the entropy of the conditional label distribution over labels for that point). Black means certain; red, orange, and yellow through white mean more uncertain.

of the label given to each datapoint. This plot was constructed in exactly the same was as was Figure 2.5 in the previous chapter. The higher the entropy of the marginal distribution over a points labeling the hotter the color assigned (black → red → orange → yellow → white). Points that are nearly white have marginal posterior labeling distributions that have high entropy which means that the cluster to which they are attributed is uncertain under the model.

Both the pursuit tracking a pinball datasets have channels that have relatively unambiguous cluster boundaries and channels that have ambiguous boundaries. Additionally, the apparent cluster cardinalities in both sets of channels vary from apparently unambiguous to very ambiguous. Half of the pursuit tracking channels are characterized by having relatively certain cluster memberships and class cardinalities this is reflected in Figure 3.4 (upper and bottom right, middle left). In the pinball data all but one of the channels have apparently quite ambiguous cluster membership and cardinalities. This is reflected in an equivalent plot for the pinball data in Figure 3.6. We conclude from visual inspection of the unsorted data and the models' estimate of uncertainty that the pinball data is more ambiguous that the pursuit tracking data, but that uncertainty about both cluster membership and cardinality are evident in both datasets.

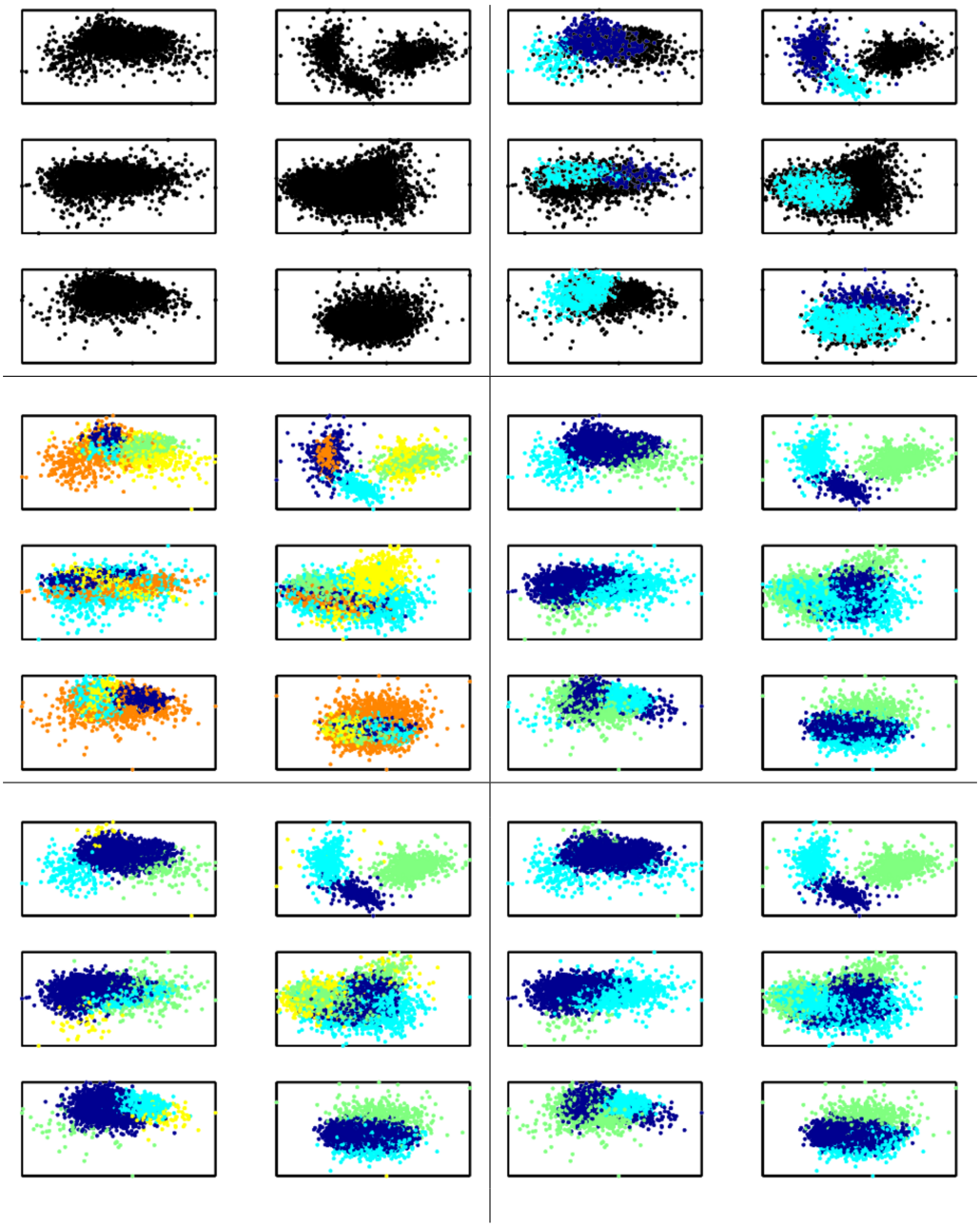

Figure 3.5: Results from sorting six channels of the pursuit tracking neural data using infinite Gaussian mixture modeling (IGMM). PCA projections of waveforms from channels 1-6 are shown in each of the six panels. The top left panel shows the unsorted waveforms, the top right shows a manual labeling (the black points are not assigned to any neuron), the first panel in the second row shows a maximum likelihood labeling, and the remaining three are samples from the IGMM posterior.

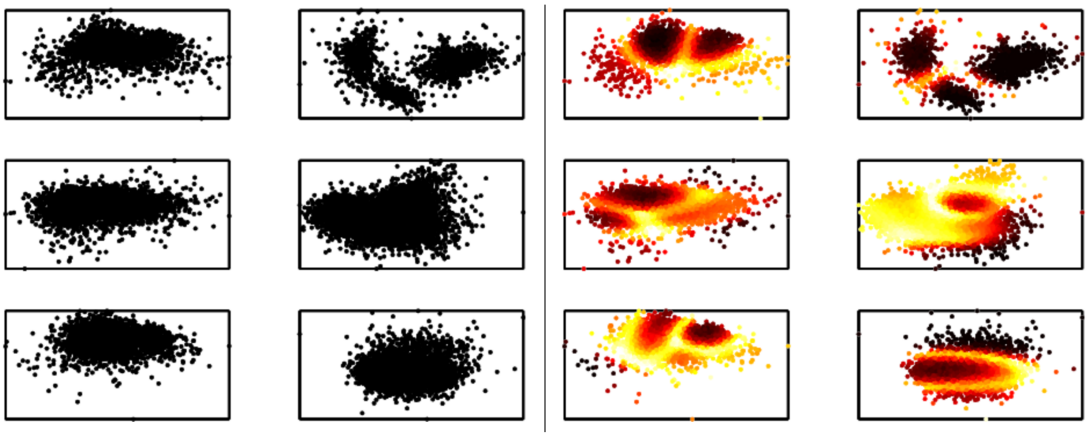

Figure 3.6: Results from sorting six channels of pinball neural data using infinite Gaussian mixture modeling (IGMM). Channels 1-6 are shown in both panels. On the left is the unsorted data, on the right the same points are plotted but with colors that indicate how uncertain the cluster label for that point is (the entropy of the conditional label distribution over labels for that point). Black means certain; red, orange, and yellow through white mean more uncertain.

3.6.1 Preferred direction somatotopy?

Given a model that represents spike sorting uncertainty we now show how to exploit it in a neural data inference task. The inference task we illustrate here is largely arbitrary and the amount of data anlyzed is insufficient to make strong claims about the validity of the results. The real point of this example inference task is to demonstrate posterior inference in the IGMM spike sorter.

Our inference task is to disprove a "null hypothesis" (quotes here because hypothesis testing is somewhat incompatible with Bayesian methods in general) which states that "preferred direction" is the basis of a fine grained motor cortical somatotopy. We seek to infer that this hypothesis does not hold and would like to be able to express our level of confidence in this result. Somatotopy is the word used to describe mappings of regions or parts of the body to specific functional areas of the cerebral cortex. There is widespread historical evidence of such mappings in visual, sensory, and motor cortex; however much of the work is focused on high level organization. For instance, as reviewed by reviewed by Aflalo and Graziano [2006], arm movement is encoded above hand movement (closer to the top of the head) but below leg movement in motor cortical somatotopic organization. Here we hypothesis a basis for fine grained somatotopic organization. We are interested in understanding the fine grained organization of the arm coding region of motor cortex as it

will affect neural decoding algorithm design decisions.

In Georgopoulos et al. [1982] it was established that neural firing rates in primate motor cortex change in response to the direction of hand movement in two dimensional movement tasks. The direction that evokes a cell's maximum firing rate is called its preferred direction. Furthermore they found that the firing rate of a cell is higher for movement in directions near a cell's preferred direction and falls off smoothly as a function of the difference between the direction of movement and the cell's preferred direction. It was shown that a cosine function (cosine tuning curve) fit the relationship between average firing rate and movement direction well. The preferred direction for a cell may be established by cosine regression and differentiation.

In Georgopoulos et al. [1986] this observation was exploited to "decode" hand position from the firing rate of a population of motor cortical neurons. Given a population of cells for which preferred directions are known they showed that a simple decoder may be constructed that uses these preferred directions and the cells' instantaneous firing rates. This is done by multiplying the instantaneous firing rate of each cell with a normalized vector that points in the direction of its preferred direction. Adding these weighted vectors together produces an estimate of the true movement direction. They also established that speed may be decoded by linearly regressing the aggregate population firing rate onto speed.

Underlying this decoding approach is the assumption that knowing precisely which spike originated from which neuron will result in better decoding performance. This is because each cell has its own preferred direction and utilizing this information should only help. However, in Wood et al. [2004b] evidence emerged that indicated that this might not actually be the case. In that study it was shown that not sorting multielectode data produced better decoding results than carefully sorting the data. There are many possible interpretations of this finding: the most reasonable among them is that decoding algorithms are generally susceptable to noisy rate estimation. This means that by adding the signal of multiple neurons together, so long as they don't have opposed preferred directions and similar firing rates will help rather than hurt decoding. As the per channel signal will be more robust (there clearly are always more spikes per channel than there are spikes per neuron per channel) the instantaneous rate estimate for the multiunit comprised of all neurons recorded on a channel will be less noisy than the individual rate estimates for all of the individual neurons on a channel. So long as neurons on the same channel (i.e. neurons that are very close to one another in the motor cortex) have preferred directions that are not diametrically opposed gains in decoding performance may be had by simply combining the output of all of the neurons; treating the result as a single neuron whose preferred direction is an average

of the individual cells preferred directions.

Here the hypothesis about preferred direction being the basis of a fine grained motor cortical somatotopy emerges. If it is the case that preferred direction is the basis of fine grained motor cortical somatotopy then the preferred directions of cell recorded on the same channel of a microelectrode array should be the same or near the same. If this is the case then averaging the output of cells recorded on the same channel will almost certainly always improve decoding performance. In other words spike sorting is unecessary for good decoding. If it is not the case then decoding performance should vary, perhaps significantly, depending on the spike trains produced.

A characterization of the distribution of preferred directions in a small volume of tissue can be arrived at by estimating the preferred directions for cells recorded on the same electrode. We have 12 channels (electrodes) from primate motor cortex as well as associated arm position information. In Figure 3.7 we show an estimate of the preferred direction distribution for the pursuit tracking data. This plot has the same row and column ordering as the subpanels in Figure 3.3. The red dashed lines indicate the estimated preferred directions for the manual labeling. The solid black lines of varying length protruding like spines out from the inner circle are a representation of a marginal of the posterior distribution learned by the IGMM spike sorter. The length of each of these lines is proportional to the normalized count of the number of times a cell with a given preferred direction was found in the posterior distribution.

To be concrete, remember that the IGMM posterior distribution is a collection of spike trains that are samples drawn from the posterior (in the particle filtering case these samples are weighted). For each spike train sample the preferred directions for each posited cell were computed. The normalized count of the number of times a cell with a given preferred direction was found was computed by dividing the number of times a cell with a particular preferred direction was found by the number of cells identified in all of the posterior samples. This we call the posterior marginal distribution over cell preferred direction. It is the distribution over cell preferred directions and indicates the level of certainty one can have in concluding that a cell with a certain preferred direction exists on the channel.

In Figure 3.7 there is a direct correspondence between the subpanels of the figure on the left and the subpanels of the figure on the right. The figure on the left is the manual sorting of the pursuit tracking data repeated from Figure 3.3. The figure on the right displays a marginal distribution computed using the IGMM spike train model which represents the probability of finding a cell with any preferred direction on that channel.

The upper right subpanels of both subfigures in Figure 3.7 correspond to a channel that

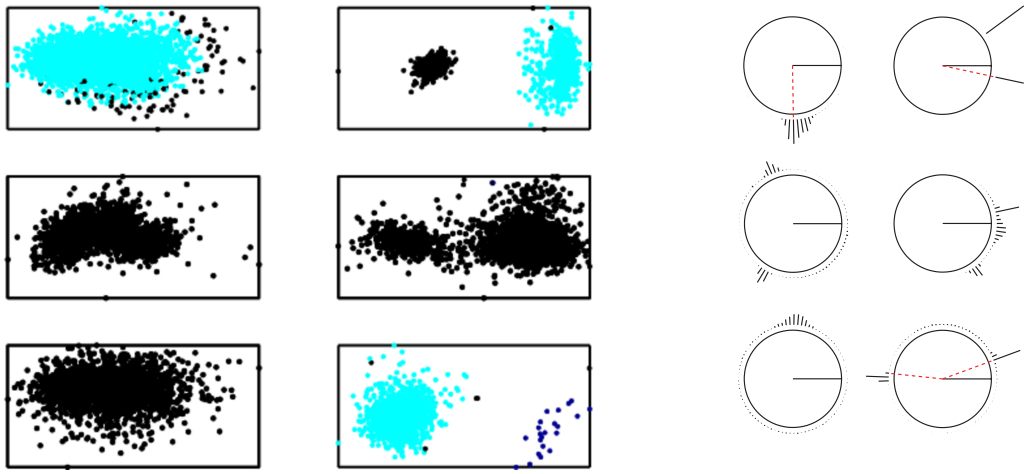

Figure 3.7: On the left: human sorted pursuit tracking waveforms. On the right: preferred direction distribution for the pursuit tracking channels. The solid black line in the circle indicates movement to the right. The dashed red lines indicate the preferred direction of cells identified in the manually sorted data. The radial ticks outside of the circle are the normalized histogram of cell preferred direction counts from the IGMM posterior.

has two clear clusters. The tickmarks that surround the circle in the preferred direction marginal panel show two tightly peaked modes in the posterior marginal over cell preferred direction cardinality; one with preferred direction of approximately $\pi/4$ and the other with preferred direction of approximately $15\pi/8$. The solid horizontal black line inside the circle corresponds to 2π radians which is rightward horizontal hand motion (positive x hand motion). The dashed red line is the preferred direction of the manually identified cell. Here there is good correspondence between the manual labeling and one of preferred direction modes on the channel. By comparing the two panels it can plainly be seen that the human sorter only found one cluster unambiguous enough to label. The upper left panel in Figure 3.7 shows good correspondence between the manual labeling and the posterior distribution mode. Here the posterior marginal over preferred direction cardinality is not as tightly peaked however. This spread is due to the uncertainties that arise in assigning spikes to neurons. These uncertainties were illustrated in Figure 3.4 where points that lie close to natural cluster boundaries shift from one class to another. This relabeling of points results in shifts in the computed preferred direction. A possible effect of this preferred direction variability as a consequence of spike sorting ambiguity is illustrated bottom right panels of the subfigures in Figure 3.7 where the manual labeling identifies a neuron with a preferred direction that is skewed from the most likely preferred direction of the cell(s) on the channel

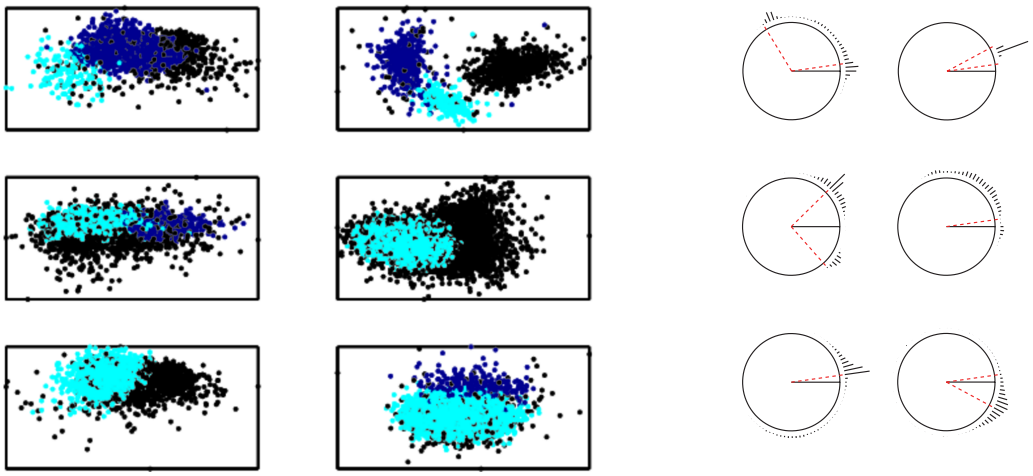

Figure 3.8: On the left: human sorted pinball waveforms. On the right: preferred direction distribution for the pinball channels. The solid black line in each of the circles indicates movement to the right. The dashed red lines indicate the preferred direction of cells identified in the manually sorted data. The radial ticks outside of the circle are the normalized histogram of cell preferred direction counts from the IGMM posterior.

under the model.

The middle row of Figure 3.7 and first column of the third row have no red dotted lines. This is because the manual labeling identified no neural waveforms. Using the IGMM spike train model reveals clear evidence that there is more than one cell on the two of these channels as the posterior marginal over preferred direction cardinality is multimodel for each.

Figure 3.8 shows similar results for the pinball data, except here the correspondence between the manual labeling and the posterior modes is better than for the pursuit tracking data.

We find reassurance in these figures due to fact that the analysis based on the manually sorted data and the analysis based on the IGMM spike train model are in agreement with respect to the ultimate conclusion: the distribution of preferred directions between closely spaced neurons does not follow a fine-grained preferred direction somatotopic organization. This means that either preferred direction isn't the basis of fine grained motor cortical somatotopy or perhaps that the low level organization of this region of motor cortex isn't somatotopically organized.

The ultimate point of this inference excercise was not to establish definitively whether or not preferred direction forms the basis of a fine grained somatotopic mapping in motor

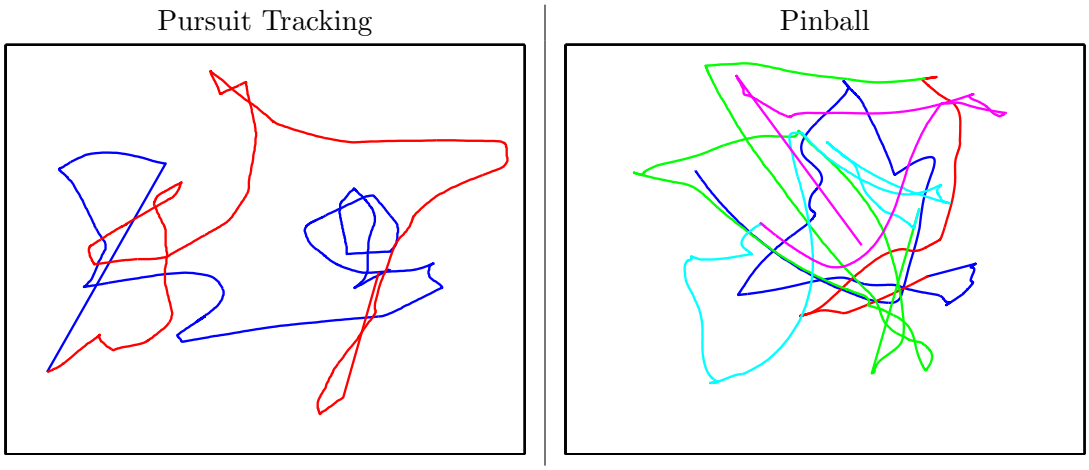

Figure 3.9: On the left: two example pursuit tracking hand position trajectories, on the right: five example pinball hand position trajectories.

cortex but was instead to demonstrate how the IGMM posterior distribution could be utilized by casting a spike train analysis as inference from using a spike train model. Here we framed a neuroscientifically relevant hypothesis in such a way as to allow it to be evaluated over the IGMM posterior distribution over spike trains: "Is preferred direction the fine grained somatotopic coordinate system in motor cortex?" As the posterior distribution over preferred direction cardinality was multimodal we can conclude that it is not. Additionally the posterior marginal distribution over preferred direction cell cardinality allows us to express our (un)certainty about this finding steming from spike sorting uncertainties, both in the number of neurons and in the attribution of spike to neurons. Here we do not compute a numerical certainty in our inference finding as the multimodality of the posterior distribution is visually apparent. In the next section more quantitative measures of confidence in inference results are reported.

3.6.2 Neural decoding

Subsequent to the seminal decoding work of Georgopoulos et al. [1986] many additional neural decoders have been developed, each generally speaking successively better than the next. It was established that the linear firing rate models employed in Serruya et al. [2003] and Wu et al. [2005] are generalizations of the cosine tuning population decoding algorithm. In these the preferred direction framework is subsumed into a more general linear regression model of neural firing rate as a function of hand velocity . In the case of the work of Serruya et al. [2003] a linear filter was used to decode hand position from neural firing rate given

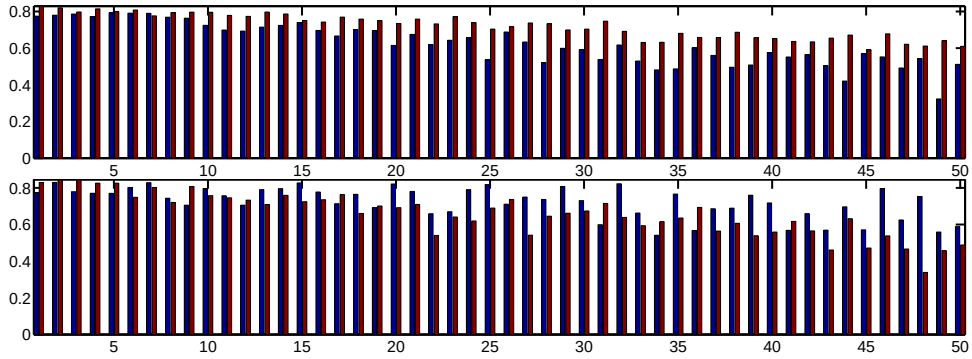

Figure 3.10: Results for decoding pursuit tracking hand position from the six channels shown in Figure 3.3. The top subplot shows average correlation coefficients (x in blue, y in red) decoding results for ten random partitionings of each channel into the number of bins given on the horizontal axis. The bottom figure shows the same for data partitioned by K-means [Duda et al., 2000] into K different partitions where K is given on the horizontal axis. Note that decoding performance does not suffer until the data is partitioned into ten or more partitions and not significantly until it is partitioned into many more. Also note that the decoding results for both partitioning styles are comparable to the results for decoding from the best automated models.

regression coefficients learned from training data. The work of Wu et al. [2005] extended this by embedding the linear regression model into a recursive Bayesian estimation framework. We will adopt the recursive Bayesian estimation framework proposed for neural decoding in Wu et al. [2005] in this section.

Evidence that there may be multiple cells with distinct preferred directions per channel implies that knowing the correct labeling could improve neural decoding provided that the resulting rate estimates are not too noisy. Unfortunately there are multiple convolved factors in decoder performance. These we will touch on later.

Our neural decoding problem involves inferring intended hand position from motor cortical neural activity alone. We follow Wu et al. [2005] in treating this as a recursive Bayesian inference task where $\vec{x}_t$ is the intended hand position we aim to recover given neural firing activity $\vec{z}_t$. Our goal is to show that accounting for spike train uncertainty results in decoding performance that is no worse than not doing so and in general can result in improved decoding performance.

Let $\vec{x}_t \in \mathbb{R}^6$ be a six component vector consisting of planar cartesian hand position, velocity, and acceleration. This observed quantity was measured using a manipulandum under task specific control by a monkey performing a pinball task as documented in Serruya et al. [2002]. The movement coordinate system was established with respect to the center

of the plane to which the movement of the manipulandum was restricted. Radial movement direction is reported such that zero degrees corresponds to positive horizontal movement. Example hand trajectories for two different tasks are shown in Figure 3.9.

Decoding hand position $\vec{x}_t$ is easiest to describe with respect to a single spike train. Given a posterior distribution over spike trains this procedure is repeated for each sample and the results are averaged straightforwardly. For each cell on all channels rates were calculated by counting the number of spikes that occured in 70 msec bins. The correspondence between rate z_t and x_t was established by aligning the rate from 140 msec in the future with the current hand position. As in Wu et al. [2005], for each posited neuron the binned spike counts were transformed by subtracting the mean count from all bins and then taking the square root of the result in each bin. Transforming the rate in this way makes the resulting distribution of transformed rates closer to being normally distributed, an assumption made by the decoding algorithms to follow. A matrix $\mathbf{Z} \in \mathbb{R}^{NxT}$ consisting of the transformed binned counts of all cells was constructed where N is the total number of cells posited on all channels and T is the number of 70 msec. bins in the recording minus two to account for the typical lag between firing rate and motor output. A similar matrix $\mathbf{X} \in \mathbb{R}^{DxT}$ was constructed for the kinematic correlates $\mathbf{X}$ with $D = 6$ rows (x, y position, velocity, and acceleration) and T columns, T here being the same as for $\mathbf{Z}$.

Following Wu et al. [2005] we used a Kalman filter to recursively estimate a posterior predictive distribution over $\vec{x}_t$, $P(\vec{x}_t|\vec{z}_t \ldots \vec{z}_1)$. The Kalman filter [Kalman, 1960] is a closed form solution to the following recursion

$$P(\vec{x}_t|\vec{z}_t) \propto P(\vec{z}_t|\vec{x}_t) \int P(\vec{x}_t|\vec{x}_{t-1}) P(\vec{x}_{t-1}|\vec{z}_{t-1}) d\vec{x}_{t-1}.$$

The Kalman filter makes a number of strong assumptions about both the generative observation process, $\vec{z}_t = f(\mathbf{X}, \mathbf{Z})$ and the state process $\vec{x}_t = g(\mathbf{X}, \mathbf{Z})$. In particular the following linear Gaussian and conditional independence assumptions are made

$$\begin{aligned} \vec{x}_t &= \mathbf{A}\vec{z}_t + \vec{w}, \quad \vec{w} \sim \mathcal{N}(\vec{0}, \mathbf{W}) \\ \vec{z}_{t-1} &= \mathbf{H}\vec{z}_t + \vec{,q} \quad \vec{q} \sim \mathcal{N}(\vec{0}, \mathbf{Q}) \end{aligned}$$

where $\vec{w} \sim \mathcal{N}(\vec{0}, \mathbf{W})$ means $\vec{w}$ is distributed normally with zero mean and covariance $\mathbf{W}$. Partitioning $\mathbf{X}$ and $\mathbf{Z}$ into training data and test data, the matrices $\mathbf{A}$ and $\mathbf{H}$ are arrived at by solving the linear equations, $\mathbf{Z}^\dagger = \mathbf{A}\mathbf{X}^\dagger$ and $\mathbf{Z}^\dagger_{2:T} = \mathbf{H}\mathbf{Z}^\dagger_{1:(T-1)}$ where $\mathbf{X}^\dagger$ is the training

data subset of $\mathbf{X}$, $\mathbf{Z}^\dagger$ is the same for the rate matrix, and $\mathbf{Z}^\dagger_{2:T}$ is all rows of $\mathbf{Z}^\dagger$ and all but the first column. The matrices $\mathbf{W}$ and $\mathbf{Q}$ are the sample covariance of the respective error residuals.

Pursuit tracking task decoding

The pursuit tracking data consisted of ten epochs of pursuit tracking. Example trajectories are shown in Figure 3.9. Eight tracking sequences of $\mathbf{X}$ and $\mathbf{Z}$ from the recording were extracted and used as training data. Two were extracted and used as test data. Linear Gaussian parameters were estimated for the state and observation processes from the training data as explained above. The Kalman filter was used to recursively estimate test hand position given test firing rates as described in the work of Wu et al. [2005]. Results are reported for test tracking epochs alone. The resulting predicted trajectory was compared to the true trajectory by computing the correlation coefficients and mean squared error between the true and predicted values of $\vec{x}_t$ for all points in the trajectory.

Figure 3.11 and Figure 3.12 illustrate the advantage of accounting for spike sorting uncertainty in decoding from motor cortical firing activity. Both the correlation coefficient and the mean squared error between true and estimated hand position are shown. Higher correlation coefficients are better, whereas lower mean squared errors are better. For the mean squared error only the relative ranking of spike train analysis methods can be deduced from as the errors are not scaled to real world units. Reference decoding results were also obtained from spike trains created by sorting randomly, not sorting, sorting using unregularized maximum likelihood (ML) finite Gaussian mixture modeling (GMM) with BIC model selection, and sorting manually. The random sorting was performed in the same was as described in the generation of Table 3.2 by randomly assigned spikes to one of three different neurons. No sorting means that all spikes were assigned to the same neuron. The manually sorted data was decoded twice, once excluding waveforms discarded because that were deemed too ambiguous to sort (black dots in the upper right panel of Figure 3.3), and once treating those excluded waveforms as having been generated by a single other cell.

The box plots representing decoding using the IGMM spike train model visualize the distribution of results obtained by stochastically selecting one spike train per channel from the corresponding posterior distribution, decoding to generate an estimated trajectory, then repeating this process. The median of this distribution is an approximation to the true posterior mean over all possible decodings. To compute the true posterior mean and variance, decoding would have to be repeated number of times exponential in the number of channels.

In the case of the particle filter posterior the spike trains were selected according to their weights.

In generating these plots the Kalman filter was trained and tested using only those partitions that generated more than 5 percent of the total number of spikes in the posterior distribution, discarding the rest. This means that some amount of the information available for decoding was discarded. This was necessary due to numerical stability issues arising in the computing the inverse covariance matrix for classes with very few points.

Additionally these decoding results are highly convolved with characteristics of the Kalman filter decoder. For instance it is the case as shown in Figure 3.10 that, up to a limit, even randomly partitioning the data does not significantly hurt and may even improve decoding results so long as there is sufficient signal to estimate and invert all class covariance matrices. This is because the linear regression model and the Kalman filter will automatically learn the preferred direction of each partition. It is as if the activity from each true neuron is partitioned into several less active pseudo-neurons, each with their own preferred direction. So long as the spikes assigned to each partition are consistently from the same neuron (or neurons with the same preferred direction) then there should be no degradation of decoding performance. Thus the only gain to be had in taking the NPB approach arises from the difference between ensuring that spikes from the same neuron are grouped together and partitioning the data in such a way that some partitions mix spikes from cells with opposed or nearly opposed preferred directions. We expect this effect to be small.

That said, in this dataset we see evidence that taking into account spike sorting uncertainty may improve neural decoding results on average. In Figure 3.11 we see a recapitulation of the finding in Section 3.4 that decoding from spike trains arrived at by not sorting at all or even randomly sorting does, in general, just as well or better than decoding from manually sorted spike trains. We also see that decoding using the IGMM spike train model produces results that are on average as good or better than decoding from spike trains produced by other sorting approaches. Most importantly, the "error bars" (actually quartiles of the decoding results) produced from the IGMM decoding results indicate what level of confidence one may have in the average decoder output as a function of spike train uncertainty. This is the primary contribution we claim here.

Pinball task decoding

The pinball data consisted of fifty epochs of pinball target acquisition data. Fourty five pinball sequences of $\mathbf{X}$ and $\mathbf{Z}$ from the recording were extracted and used as training data. Five were extracted and used as test data. Example pinball hand position trajectories are shown in Figure 3.9. All other data processing and decoding processes were repeated as described for the pursuit tracking data above.

In Figure 3.12 decoding results are shown for the pinball data. The claims to be made here are similar to those for the pursuit tracking data; however, here the ML spike sorting approach appears to have produced spike trains that result in good decoding results. While it may be the case that ML spike sorting always will produce spike trains that are better for decoding, that may not always be the case and as the ML approach produces a single best spike train without any estimate of uncertainty ones confidence in this ranking cannot be established. The ML decoding results are still within the range of variability predicted by decoding from the NPB spike train model, although for this data at the high end of the scale. Here again the primary contribution is a demonstration of how to account for spike train variability in subsequent spike train analyses.

For both the pursuit tracking and pinball data the error bars for the Gibbs sampler IGMM posterior estimator encompass decoding results for not sorting, randomly sorting, ML sorting, and so forth. There is insufficient evidence to make any claims about improved decoding, but that in and of itself was not our aim. Being able to illustrate the effect of spike train variability in a systematic way was out aim and these figures show just that. Of some concern and of interest with respect to future work is why the particle filter posterior estimate of the decoding variability due to spike train uncertainty is smaller that the Gibbs sampler posterior estimate of the same. We are not entirely sure why this is so and future work will include investigating the cause behind this effect.

3.7 Discussion

There are a number of shortcomings to the IGMM spike sorting approach, and these we will dispense with first, prior to highlighting the real and potential benefits of adopting the nonparametric Bayesian spike sorting approach. Perhaps the biggest problem is that the technique does not extend down to modeling the raw voltage trace. Computational infeasability rapidly becomes an issue when even contemplating such a model as the amount of data to be modeled in that case is enormous. However, that our proposed method does

not extend to detection means that not all variability in spike sorting is expressed in the posterior distribution over sortings. Necessarily variability in spike detection cannot be included because it isn't modeled. This shortcoming is coupled with another shortcoming, namely that our method cannot detect and resolve overlapping spikes. In Görür et al. [2004] a mixture of factor analyzers technique for resolving overlapping waveforms was developed that should be integrable with our technique with relatively little trouble besides increased computational complexity. Our model also does not explicitely exclude spikes from occurring in the refractory period, i.e. two spikes can be generated by the same neuron under our model even if the time between them is less than a few milliseconds. As our generative model is of waveshape alone we would need to integrate timing information into the model. This is quite feasible under this model and could be easily and computationally efficiently implemented. Simply appending the timing information as another dimension of the waveform and constructing a likelihood function whereby the likehood of generating spikes within some sort time difference is very near zero for any particular neuron. A similar same kind of strategy can be employed to account for changing spike waveshape. It should be noted here, however, that the anisotropic Gaussian distribution is fairly robust to waveshape change already. Lastly, for long term chronic spike sorting applications such as unattended neuroprosthetics, disappearance and appearance of classes should be modeled. In it's current form, even in the case of the online posterior estimation algorithm, the model cannot handle disappearance and appearance of new neurons. Leveraging the dependent Dirichlet process work of Srebro and Roweis [2005] and Griffin and Steel [2006] should allow nonparametric spike sorting models for chronic spike sorting applications. Lastly our model is somewhat sensitive to hyperparameter settings. This can be addressed in two ways: one is to use less informative priors, particularly in place of the MVN IW prior; the other is to treat them in a Bayesian way as well. This is relatively simple to do in the Gibbs sampling case as a distribution can be estimated for the hyperparameters themselves. Depending on the choice of priors this will increase the dimensionality of the sampling space which necessarily will make sampling more challenging, particularly in the particle filtering case.

While these shortcomings are significant and may limit the applicability of the non-parametric spike sorting approach to a subset of rather than all neural data inferences (for instance, tests for excess synchrony would be an inappropriate use of this model as overlapping waveforms are not detected), it still remains that this IGMM approach to spike sorting is a significant step towards accounting for spike sorting uncertainty in inference from neural data. While our approach is similar in this respect to Nguyen et al. [2003], they did not demonstrate how the utility of the full posterior distribution. Furthermore,

the IGMM posterior estimation approach is simpler to implement and admits a sequential estimation variant.

Beyond reporting confidence intervals and accounting for spike train variability arising from spike sorting uncertainties, particle filter posterior estimation enables online spike sorting, a long time goal of the community. Looking towards a future in which online IGMM spike sorting is feasible (specialized software and/or dedicated hardware would be necessary to achieve this now) opens up a number of possible novel spike sorting applications. For instance, it is possible to imagine an experiment where one wants to find a neuron that modulates its firing rate in response to some particular stimulus. Running an online IGMM spike sorter in conjunction with positioning the recording apparatus might allow the experimenter to test for the presence of such a cell in the recording at the same time as positioning the device.

Additionally, in the case of chronic implants for neural prosthetics and long term experimentation an online spike sorter could run for the duration of the implant, sorting everything as it is recorded, dramatically enhancing the amount of spike train data available to analyze.

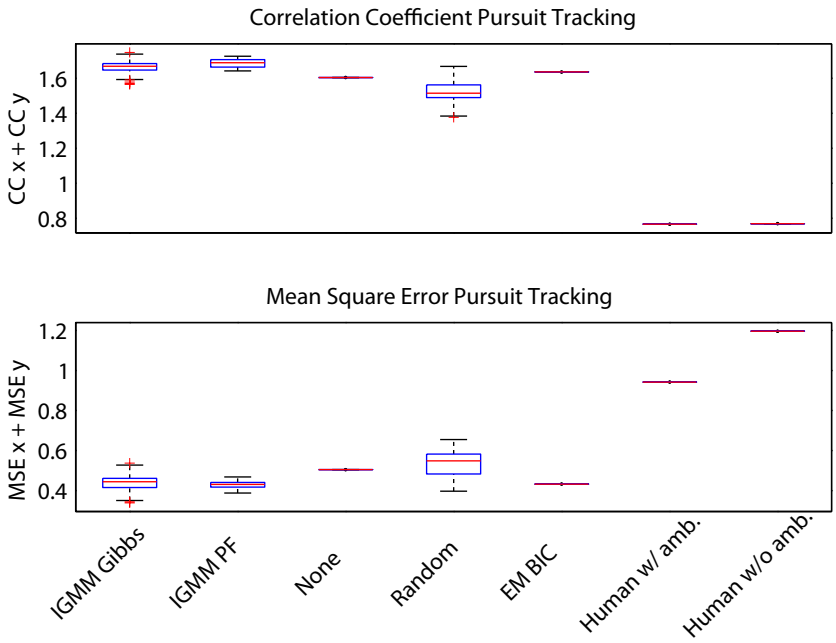

Figure 3.11: Results for decoding pursuit tracking hand position from the six channels shown in Figure 3.3. The top subplot shows the sum of x and y correlation coefficients (CC) between decoding and true hand positions; the bottom shows the sum mean squared error for the same. The label for each box and whisker is printed below the horizontal axis of the bottom figure. Results are shown for our NPB spike train model, particle filter and gibbs sampler posterior estimation, expectation maximization finite Gaussian mixture model spike sorting with BIC model selection, no sorting, random sorting, and manually sorted spike trains both including spikes that were deemed too ambiguous and excluding those spikes. For the NPB and random distributions over spike trains the middle line in the box indicates the median decoding result, the box indicates the first quartile, and the whiskers indicate the full extent of the results. Although it may be possible to improve decoding results by accounting for spike sorting uncertainty, our contribution is the development of a model that makes it possible to report meaningful "error bars" that represent the affect of spike sorting uncertainty on spike train analysis results.

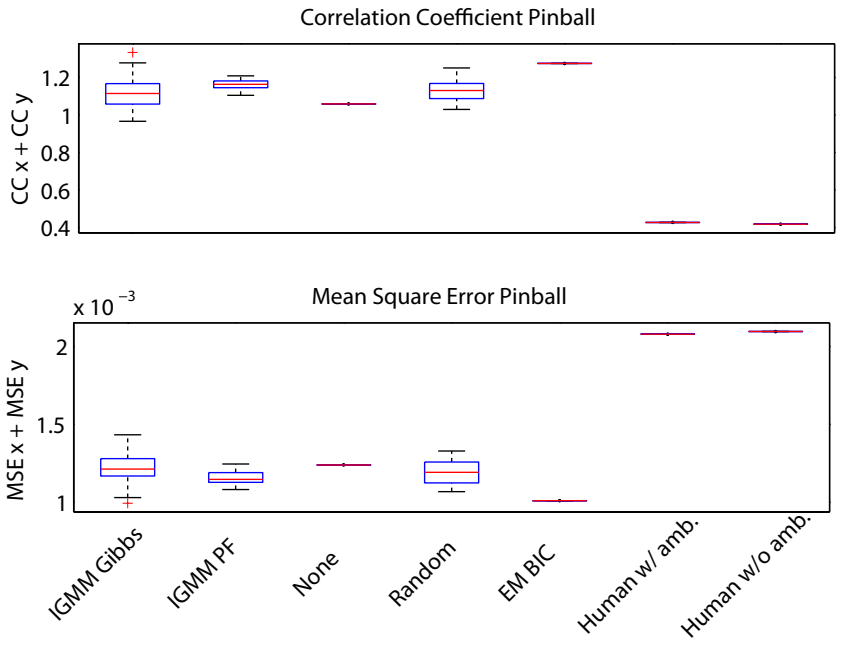

Figure 3.12: Results for decoding pinball task hand position from the six channels shown in Figure 3.3. The top subplot shows the sum of x and y correlation coefficients (CC) between decoding and true hand positions; the bottom shows the sum mean squared error for the same. The label for each box and whisker is printed below the horizontal axis of the bottom figure. Results are shown for our NPB spike train model, particle filter and gibbs sampler posterior estimation, expectation maximization finite Gaussian mixture model spike sorting with BIC model selection, no sorting, random sorting, and manually sorted spike trains both including spikes that were deemed too ambiguous and excluding those spikes. For the NPB and random distributions over spike trains the middle line in the box indicates the median decoding result, the box indicates the first quartile, and the whiskers indicate the full extent of the results. Although it may be possible to improve decoding results by accounting for spike sorting uncertainty, our contribution is the development of a model that makes it possible to report meaningful "error bars" that represent the affect of spike sorting uncertainty on spike train analysis results.

Chapter 4

Nonparametric Bayesian Matrix Factorization

In Chapter 2 we reviewed a simple nonparametric latent variable model, the infinite Gaussian mixture model (IGMM), and illustrated its utility by using it to build a model of spike trains. There our contribution was a novel application of the IGMM to the problem of spike train modeling. The purpose of reviewing the IGMM was to review nonparametric Bayesian modeling and to prepare the reader for this half of the dissertation in which the contributions are more theoretical in nature. In this chapter we present a new nonparametric Bayesian latent variable model and associated estimation algorithms. In the next chapter we demonstrate this model on an inference task from neurological data.

More specifically we present a novel nonparametric Bayesian binary matrix factorization model in which, unlike the IGMM, each observation can be influenced by multiple hidden causes. Additionally we develop particle filter posterior estimation algorithms for a class of NPB matrix factorization models. We start by briefly reviewing classical probabilistic, Bayesian, and NPB matrix factorization before introducing our new model. We conclude this chapter by introducing particle filter sequential posterior estimation for a family of NPB matrix factorization models.

4.1 Matrix factorization

One of the goals of unsupervised learning is to discover the latent structure expressed in observed data. The nature of the learning problem will vary depending on the form of the data and the kind of latent structure it expresses, but many unsupervised learning problems

can be viewed as a form of matrix factorization – i.e. decomposing an observed data matrix, $\mathbf{X}$, into the product of two or more matrices of latent variables. If $\mathbf{X}$ is an $N \times D$ matrix, where N is the number of D-dimensional observations, the goal is to find a low-dimensional latent feature space capturing the variation in the observations making up $\mathbf{X}$. This can be done by assuming that $\mathbf{X} \approx \mathbf{ZY}$, where $\mathbf{Z}$ is a $N \times K$ matrix indicating which of (and perhaps the extent to which) K latent features are expressed in each of the N observations and $\mathbf{Y}$ is a $K \times D$ matrix indicating how those K latent features are manifest in the D dimensional observation space. Typically, K is less than D, meaning that $\mathbf{Z}$ and $\mathbf{Y}$ provide an efficient summary of the structure of $\mathbf{X}$.

A standard problem for unsupervised learning algorithms based on matrix factorization is determining the dimensionality of the latent matrices, K. Nonparametric Bayesian statistics offers a way to address this problem: instead of specifying K a priori and searching for a "best" factorization, nonparametric Bayesian matrix factorization approaches such as those in [Griffiths and Ghahramani, 2005] and [Wood et al., 2006b] estimate a posterior distribution over factorizations with unbounded dimensionality (i.e. letting $K \to \infty$). This remains computationally tractable because each model uses a prior that ensures that $\mathbf{Z}$ is sparse, based on the Indian Buffet Process (IBP) [Griffiths and Ghahramani, 2005]. The search for the dimensionality of the latent feature matrices thus becomes a problem of posterior inference over the number of non-empty columns in $\mathbf{Z}$.

In this chapter we review previous nonparametric Bayesian matrix factorization results which use Gibbs sampling for posterior estimation [Griffiths and Ghahramani, 2005] and revisit our our work which first appeared in [Wood et al., 2006b]. In particular we review the infinite linear Gaussian matrix factorization model of [Griffiths and Ghahramani, 2005] and then our own infite binary matrix factorization model. As suggested in Chapter 2, Gibbs sampling is the standard estimation algorithm used in nonparametric Bayesian modeling. However, as also reviewed in Chapter 2, sequential Monte Carlo methods such as particle filtering can provide an efficient alternative to Gibbs sampling in Dirichlet process mixture models [Fearnhead, 2004; MacEachern et al., 1999].

Accordingly in this chapter we develop a novel particle filtering algorithm for posterior estimation in matrix factorization models that use the IBP, and illustrate its applicability to two specific models – one with a conjugate prior, and the other without a conjugate prior but tractable in other ways. Our particle filtering algorithm is by nature an "on-line" procedure, where each row of $\mathbf{X}$ (observation) is processed only once, in sequence. This stands in comparison to Gibbs sampling, which must revisit each row many times to converge to a reasonable representation of the posterior distribution. We present simulation results

showing that our particle filtering algorithm can be more efficient than Gibbs sampling and discuss its applicability to the broad class of nonparametric matrix factorization models based on the IBP.

4.2 Bayesian matrix factorization

Let $\mathbf{X}$ be an observed $N \times D$ matrix. Our goal is to find a representation of the structure expressed in this matrix in terms of the latent matrices $\mathbf{Z}$ ($N \times K$) and $\mathbf{Y}$ ($K \times D$). This can be formulated as a statistical problem if we view $\mathbf{X}$ as being produced by a probabilistic generative process, resulting in a probability distribution $P(\mathbf{X}|\mathbf{Z}, \mathbf{Y})$. The critical assumption necessary to make this a matrix factorization problem is that the distribution of $\mathbf{X}$ is conditionally dependent on $\mathbf{Z}$ and $\mathbf{Y}$ only through the product $\mathbf{ZY}$. Although defining $P(\mathbf{X}|\mathbf{Z}, \mathbf{Y})$ allows us to use methods such as maximum-likelihood estimation to find a point estimate, our goal is to instead compute a posterior distribution over possible values of $\mathbf{Z}$ and $\mathbf{Y}$. To do so we need to specify a prior over the latent matrices $P(\mathbf{Z}, \mathbf{Y})$, and then we can use Bayes' rule to find the posterior distribution over $\mathbf{Z}$ and $\mathbf{Y}$

$$P(\mathbf{Z}, \mathbf{Y}|\mathbf{X}) \quad \propto \quad P(\mathbf{X}|\mathbf{Z}, \mathbf{Y})P(\mathbf{Z}, \mathbf{Y}). \tag{4.1}$$

This constitutes Bayesian matrix factorization, but two problems remain: the choice of K, and the computational cost of estimating the posterior distribution.

Unlike standard matrix factorization methods that require an a priori choice of K, nonparametric Bayesian approaches allow us to estimate a posterior distribution over $\mathbf{Z}$ and $\mathbf{Y}$ where the size of these matrices is unbounded. The models we discuss in this dissertation place a prior on $\mathbf{Z}$ that gives each "left-ordered" binary matrix (see [Griffiths and Ghahramani, 2005] for details) probability

$$P(\mathbf{Z}) = \frac{\alpha^{K_+}}{\prod_{h=1}^{2^N-1} K_h!} \exp\{-\alpha H_N\} \prod_{k=1}^{K_+} \frac{(N - m_k)!(m_k - 1)!}{N!} \tag{4.2}$$

where K_+ is the number of columns of $\mathbf{Z}$ with non-zero entries, m_k is the number of 1's in column k, N is the number of rows, $H_N = \sum_{i=1}^{N} 1/i$ is the N^{th} harmonic number, and K_h is the number of columns in $\mathbf{Z}$ that when read top-to-bottom form a sequence of 1's and 0's corresponding to the binary representation of the number h. This prior on $\mathbf{Z}$ is a distribution on sparse binary matrices that favors those that have few columns with many ones, with the rest of the columns being all zeros.

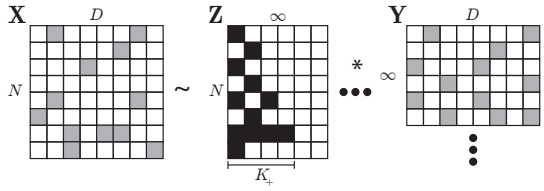

Figure 4.1: Nonparametric Bayesian matrix factorization. The data matrix $\mathbf{X}$ is the product of $\mathbf{Z}$ and $\mathbf{Y}$, which have an unbounded number of columns and rows respectively.

This distribution can be derived as the outcome of a sequential generative process called the *Indian buffet process* (IBP) [Griffiths and Ghahramani, 2005]. Imagine an Indian restaurant into which N customers arrive one by one and serve themselves from the buffet. The first customer loads her plate from the first Poisson(α) dishes. The i^{th} customer chooses dishes proportional to their popularity, choosing a dish with probability m_k/i where m_k is the number of people who have choosen the k^{th} dish previously, then chooses Poisson(α/i) new dishes. If we record the choices of each customer on one row of a matrix whose columns correspond to a dishes on the buffet (1 if chosen, 0 if not) then (the left-ordered form of) that matrix constitutes a draw from the distribution in Eqn. 4.2. The order in which the customers enter the restaurant has no bearing on the distribution of $\mathbf{Z}$ (up to permutation of the columns), making this distribution exchangeable.

In this work we assume that $\mathbf{Z}$ and $\mathbf{Y}$ are independent, with $P(\mathbf{Z}, \mathbf{Y}) = P(\mathbf{Z})P(\mathbf{Y})$. As shown in Fig. 4.1, since we use the IBP prior for $P(\mathbf{Z})$, $\mathbf{Y}$ is a matrix with an infinite number of rows and D columns. We can take any appropriate distribution for $P(\mathbf{Y})$, and the infinite number of rows will not pose a problem because only K_+ rows will interact with non-zero elements of $\mathbf{Z}$. A posterior distribution over $\mathbf{Z}$ and $\mathbf{Y}$ implicitly defines a distribution over the effective dimensionality of these matrices, through K_+. This approach to nonparametric Bayesian matrix factorization has been used for both continuous [Griffiths and Ghahramani, 2005, 2006] and binary [Wood et al., 2006b] data matrices $\mathbf{X}$.

Since the posterior distribution defined in Eqn. 4.1 is generally intractable, Gibbs sampling has previously been employed to construct a sample-based representation of this distribution. However, generally speaking, Gibbs sampling is slow, requiring each entry in $\mathbf{Z}$ and $\mathbf{Y}$ to be repeatedly updated conditioned on all of the others. This problem is compounded in contexts where the the number of rows of $\mathbf{X}$ increases as a consequence of new observations being introduced, where the Gibbs sampler would need to be restarted after the introduction of each new observation.

4.3 A semi-conjugate model: infinite binary matrix factorization

In this model, first presented in the context of learning hidden causal structure [Wood et al., 2006b], the entries of both $\mathbf{X}$ and $\mathbf{Y}$ are binary. Each row of $\mathbf{X}$ represents the values of a single observed variable across D trials or cases, each row of $\mathbf{Y}$ gives the values of a latent variable (a "hidden cause") across those trials or cases, and $\mathbf{Z}$ is the adjacency matrix of a bipartite Bayesian network indicating which latent variables influence which observed variables. Learning the hidden causal structure then corresponds to inferring $\mathbf{Z}$ and $\mathbf{Y}$ from $\mathbf{X}$. An example application of this model and a more detailed introduction to its interpretation as a model of causal structure is given in Chapter 5. The model fits our schema for nonparametric Bayesian matrix factorization model (and hence is amenable to the use of our particle filter) since the likelihood function it uses depends only on the product $\mathbf{ZY}$.

The likelihood function for this model assumes that each entry of $\mathbf{X}$ is generated independently $P(\mathbf{X}|\mathbf{Z},\mathbf{Y}) = \prod_{i,d} P(x_{i,d}|\mathbf{Z},\mathbf{Y})$, with its probability given by the "noisy-OR" [Pearl, 1988] of the causes that influence that variable (identified by the corresponding row of $\mathbf{Z}$) and are active for that case or trial (expressed in $\mathbf{Y}$). The probability that $x_{i,d}$ takes the value 1 is thus

$$P(x_{i,d} = 1|\mathbf{Z},\mathbf{Y}) = 1 - (1-\lambda)^{\mathbf{z}_{i,:}\cdot\mathbf{y}_{:,d}}(1-\epsilon) \tag{4.3}$$

where $\mathbf{z}_{i,:}$ is the i^{th} row of $\mathbf{Z}$, $\mathbf{y}_{:,d}$ is the d^{th} column of $\mathbf{Y}$, and $\mathbf{z}_{i,:} \cdot \mathbf{y}_{:,d} = \sum_{k=1}^{K} z_{i,k}y_{k,d}$. The parameter ϵ sets the probability that $x_{i,d} = 1$ when no relevant causes are active, and λ determines how this probability changes as the number of relevant active hidden causes increases. To complete the model, we assume that the entries of $\mathbf{Y}$ are generated independently from a Bernoulli process with parameter p, to give $P(\mathbf{Y}) = \prod_{k,d} p^{y_{k,d}}(1-p)^{1-y_{k,d}}$, and use the IBP prior for $\mathbf{Z}$.

4.3.1 Gibbs sampler posterior estimation

To use Gibbs sampling in this model we need to be able to sample from the distributions $P(z_{i,k}|\mathbf{X},\mathbf{Z}_{-i,k},\mathbf{Y})$ and $P(y_{k,t}|\mathbf{X},\mathbf{Z},\mathbf{Y}_{-k,t})$, where $\mathbf{Z}_{-i,k}$ is all values of $\mathbf{Z}$ except for $z_{i,k}$ and $\mathbf{Y}_{-k,t}$ is all of the values of the matrix $\mathbf{Y}$ except for $y_{k,t}$. As both $z_{i,k}$ and $y_{k,t}$ are binary, these distributions can be computed by enumeration. We can find $P(z_{i,k} = 1|\mathbf{z}_{-i,k})$ by exploiting the exchangeability of the IBP. Since any ordering of the customers results in the same distribution, we can assume that the i^{th} customer is the last to enter the

restaurant. Accordingly, they should sample each dish that has previously been tasted by $m_{-i,k}$ customers with probability $\frac{m_{-i,k}}{N}$, and try a Poisson($\frac{\alpha}{N}$) number of new dishes, for which $m_{-i,k} = 0$. Thus, we need to consider two separate cases in sampling $z_{i,k}$ in our Gibbs sampler: the case where $m_{-i,k} > 0$, and the case where $m_{-i,k} = 0$.

When $m_{-i,k} > 0$, $P(z_{i,k} = 1|\mathbf{z}_{-i,k})$ is given by

$$P(z_{i,k} = 1|\mathbf{z}_{-i,k}) = \bar{\theta}_k = \frac{m_{-i,k}}{N}. \tag{4.4}$$

In this case, we obtain

$$\begin{aligned} & P(z_{i,k} = a|\mathbf{X}, \mathbf{Z}_{-i,k}, \mathbf{Y}) \\ & \propto \quad \bar{\theta}_k^a (1 - \bar{\theta}_k)^{(1-a)} \prod_{t=1}^{T} (1 - (1 - \lambda)^{\mathbf{z}_{i,:} \cdot \mathbf{y}_{:,t}} (1 - \epsilon))|_{z_{i,k}=a} \end{aligned} \tag{4.5}$$

where $|_{z_{i,k}=a}$ means to replace $z_{i,k}$ with a in the preceding expression.

The case where $m_{-i,k} = 0$ requires more careful treatment. In our non-parametric approach, $\mathbf{Z}$ is a matrix with infinitely many columns (and $\mathbf{Y}$ is a matrix with infinitely many rows). In practice, only the non-zero columns of the matrix can be held in memory, but we still need to sample those columns. To do so let K_i^{new} be the number of columns of $\mathbf{Z}$ which contain a 1 only in row i. Then we have

$$\begin{aligned} & P(K_i^{\text{new}}|\mathbf{X}_{i,1:T}, \mathbf{Z}_{i,1:K+K_i^{\text{new}}}, \mathbf{Y}) \\ & \propto \quad P(\mathbf{X}_{i,1:T}|\mathbf{Z}_{i,1:K+K_i^{\text{new}}}, \mathbf{Y}, K_i^{\text{new}}) P(K_i^{\text{new}}) \end{aligned} \tag{4.6}$$

where the prior $P(K_i^{\text{new}})$ is Poisson($\frac{\alpha}{N}$), and we find $P(\mathbf{X}_{i,1:T}|\mathbf{Z}_{i,1:K+K_i^{\text{new}}}, \mathbf{Y}, K_i^{\text{new}})$ by marginalizing over $\mathbf{Y}^{\text{new}} = \mathbf{Y}_{K+1:K+K_i^{\text{new}},t}$, the new rows of $\mathbf{Y}$. As each entry of $\mathbf{X}$ is independent, we have

$$\begin{aligned} & P(\mathbf{X}_{i,1:T}|\mathbf{Z}_{i,1:K+K_i^{\text{new}}}, \mathbf{Y}, K_i^{\text{new}}) \\ & = \quad \prod_{t=1:T} P(x_{i,t}|\mathbf{Z}_{i,1:K+K_i^{\text{new}}}, \mathbf{Y}, K_i^{\text{new}}) \end{aligned} \tag{4.7}$$

marginalizing over $\mathbf{Y}^{\text{new}}$ gives

$$\begin{aligned}
&P(x_{i,t} = 1|\mathbf{Z}^{\text{new}}, \mathbf{Y}, K_i^{\text{new}}) \\
&= \sum_{\mathbf{Y}^{\text{new}}} P(x_{it} = 1|\mathbf{Z}^{\text{new}}, \mathbf{Y}^{\text{new}}) P(\mathbf{Y}^{\text{new}}) \\
&= \sum_{m=0}^{K_i^{\text{new}}} [1 - \eta(1-\lambda)^m(1-\epsilon)] p^m (1-p)^{K_i^{\text{new}} - m} \binom{K_i^{\text{new}}}{m} \\
&= 1 - (1-\epsilon)\eta \sum_{m=0}^{K_i^{\text{new}}} [(1-\lambda)p]^m (1-p)^{K_i^{\text{new}} - m} \binom{K_i^{\text{new}}}{m} \\
&= 1 - (1-\epsilon)\eta(1-\lambda p)^{K_i^{\text{new}}} \qquad (4.8)
\end{aligned}$$

where $\eta = (1-\lambda)^{\mathbf{z}_{i,1:K} \cdot \mathbf{y}_{1:K,t}}$. The last step makes use of the binomial theorem. This gives us all we need to compute the conditional distribution over K_i^{new} defined in Eqn. 4.6. In theory, sampling from this distribution would require evaluating it at all possible values of K_i^{new}. We approximate this by sampling from the distribution over $K_i^{\text{new}} \leq 10$.

While $\mathbf{Y}$ technically has an infinite number of rows, in practice we need only sample $y_{k,t}$ for those rows that correspond to non-zero columns of $\mathbf{Z}$. Proceeding similarly for $y_{k,t}$ gives

$$\begin{aligned}
&P(y_{k,t} = a|\mathbf{Z}, \mathbf{X}, \mathbf{Y}_{-k,t}) \qquad (4.9) \\
&\propto \; p^a (1-p)^{1-a} \prod_{i=1}^{N} (1 - (1-\lambda)^{\mathbf{z}_{i,:} \mathbf{y}_{:,t}} (1-\epsilon))|_{y_{k,t}=a}.
\end{aligned}$$

The pseudocode for our Gibbs sampler is given in Algorithm 4.

4.3.2 Particle filter posterior estimation

Our novel approach to posterior estimation in NPB matrix factorization models based on the IBP addresses the problems faced in Gibbs sampling by exploiting the fact that the prior on $\mathbf{Z}$ is recursively decomposable. To explain this we need to introduce new notation, let $\mathbf{X}^{(i)}$ be the i^{th} row of $\mathbf{X}$, and $\mathbf{X}^{(1:i)}$ and $\mathbf{Z}^{(1:i)}$ be all the rows of $\mathbf{X}$ and $\mathbf{Z}$ up to i respectively. Note that because the IBP prior is recursively decomposable it is easy to sample from $P(\mathbf{Z}^{(1:i)}|\mathbf{Z}^{(1:i-1)})$; to do so simply follow the IBP in choosing dishes for the i^{th} customer given the record of which dishes were chosen by the first $i-1$ customers (see Algorithm 5). Applying Bayes' rule, we can write the posterior on $\mathbf{Z}^{(1:i)}$ and $\mathbf{Y}$ given $\mathbf{X}^{(1:i)}$

Algorithm 4 Gibbs sampler for hidden causes

for $r = 1, \ldots,$ number of iterations **do**
  for $i = 1, \ldots, N$ **do**
    for $k = 1, \ldots, K$ **do**
      if $m_{-i,k} > 0$ **then**
        sample $z_{i,k}$ according to Eqn. 5.11
      else
        mark $z_{i,k}$ to be zeroed
      end if
    end for
    zero marked $z_{i,k}$'s
    sample K_i^{new} according to Eqn. 4.6
    $\mathbf{Z}_{i,K+1,K+K_i^{\text{new}}} \leftarrow 1$
    for all $y_{j,t} \in \mathbf{Y}_{K+1:K+K_i^{\text{new}},1:T}$ **do**
      sample $y_{j,t}$ according to Eqn. 5.12
    end for
    $K \leftarrow K + K_i^{\text{new}}$
  end for
  for all $y_{k,t} \in \mathbf{Y}$ **do**
    sample $y_{k,t}$ according to Eqn. 5.12
  end for
  remove columns with $\sum_i z_i, k = 0$ from $\mathbf{Z}$
  remove corresponding rows from $\mathbf{Y}$
end for

in the following form

$$P(\mathbf{Z}^{(1:i)}, \mathbf{Y}|\mathbf{X}^{(1:i)}) \propto P(\mathbf{X}^{(i)}|\mathbf{Z}^{(1:i)}, \mathbf{Y}, \mathbf{X}^{(1:i-1)}) P(\mathbf{Z}^{(1:i)}, \mathbf{Y}|\mathbf{X}^{(1:i-1)}). \tag{4.10}$$

Here we do not index $\mathbf{Y}$ as it is always an infinite matrix.[1]

If we could evaluate $P(\mathbf{Z}^{(1:i-1)}, \mathbf{Y}|\mathbf{X}^{(1:i-1)})$, we could obtain weighted samples (or "particles") from $P(\mathbf{Z}^{(1:i)}, \mathbf{Y}|\mathbf{X}^{(1:i)})$ using importance sampling with a proposal distribution of

$$P(\mathbf{Z}^{(1:i)}, \mathbf{Y}|\mathbf{X}^{(1:i-1)}) = \sum_{\mathbf{Z}^{(1:i-1)}} P(\mathbf{Z}^{(1:i)}|\mathbf{Z}^{(1:i-1)}) P(\mathbf{Z}^{(1:i-1)}, \mathbf{Y}|\mathbf{X}^{(1:i-1)}) \tag{4.11}$$

and taking

$$w_\ell \propto P(\mathbf{X}^{(i)}|\mathbf{Z}_{(\ell)}^{(1:i)}, \mathbf{Y}_{(\ell)}, \mathbf{X}^{(1:i-1)}) \tag{4.12}$$

as the weight associated with the ℓ^{th} particle. However, we could also use a similar scheme

[1] In practice, we need only keep track of the rows of $\mathbf{Y}$ that correspond to the non-empty columns of $\mathbf{Z}$, as the posterior distribution for the remaining entries is just the prior. Thus, if new non-empty columns are added in moving from $\mathbf{Z}^{(i-1)}$ to $\mathbf{Z}^{(i)}$, we need to expand the number of rows of $\mathbf{Y}$ that we represent accordingly.

Algorithm 5 Sample $P(\mathbf{Z}^{(1:i)}|\mathbf{Z}^{(1:i-1)}, \alpha)$ using the Indian Buffet process

$\mathbf{Z} \leftarrow \mathbf{Z}^{(1:i-1)}$
if $i = 1$ **then**
  sample $K_i^{\text{new}} \sim \text{Poisson}(\alpha)$
  $\mathbf{Z}_{i,1:K_i^{\text{new}}} \leftarrow 1$
else
  $K_+ \leftarrow$ number of non-zero columns in $\mathbf{Z}$
  for $k = 1, \ldots, K_+$ **do**
    sample $z_{i,k}$ according to $P(z_{i,k} = 1) \sim \text{Bernoulli}(\frac{m_{-i,k}}{i})$
  end for
  sample $K_i^{\text{new}} \sim \text{Poisson}(\frac{\alpha}{i})$
  $\mathbf{Z}_{i,K_+ + 1:K_+ + K_i^{\text{new}}} \leftarrow 1$
end if
$\mathbf{Z}^{(1:i)} \leftarrow \mathbf{Z}$

to approximate $P(\mathbf{Z}^{(1:i-1)}, \mathbf{Y}|\mathbf{X}^{(1:i-1)})$ if we could evaluate $P(\mathbf{Z}^{(1:i-2)}, \mathbf{Y}|\mathbf{X}^{(1:i-2)})$. Following Eq. 4.11, we could then approximately generate a set of weighted particles from $P(\mathbf{Z}^{(1:i)}, \mathbf{Y}|\mathbf{X}^{(1:i-1)})$ by using the IBP to sample a value from $P(\mathbf{Z}^{(1:i)}|\mathbf{Z}_{(\ell)}^{(1:i-1)})$ for each particle from $P(\mathbf{Z}^{(1:i-1)}, \mathbf{Y}|\mathbf{X}^{(1:i-1)})$ and carrying forward the weights associated with those particles. This "particle filtering" procedure defines a recursive importance sampling scheme for the full posterior $P(\mathbf{Z}, \mathbf{Y}|\mathbf{X})$, and is known as sequential importance sampling [Doucet et al., 2001]. When applied in its basic form this procedure can produce particles with extreme weights, so we resample the particles at each iteration of the recursion from the distribution given by their normalized weights and set $w_\ell = 1/L$ for all ℓ, which is a standard method known as sequential importance resampling [Doucet et al., 2001].

The procedure defined in the previous paragraphs is a general-purpose particle filter for matrix-factorization models based on the IBP. This procedure will work when the prior defined on $\mathbf{Y}$ is not conjugate to the likelihood (and is much simpler than other algorithms for using the IBP with non-conjugate priors, e.g. [Görür et al., 2006]). However, as we will soon demonstrate the procedure can be simplified further in special cases.

In this model the prior over $\mathbf{Y}$ is not conjugate to the likelihood, so we are forced to explicitly represent $\mathbf{Y}$ in our particle filter state, as outlined in Equations 4.10 and 4.11. However, we can define a more efficient algorithm than the basic particle filter due to the tractability of some integrals. This is why we call this model a "semi-conjugate" model.

The basic particle filter defined in Section 4.3.2 requires drawing the new rows of $\mathbf{Y}$ from the prior when we generate new columns of $\mathbf{Z}$. This can be problematic since the chance of producing an assignment of values to $\mathbf{Y}$ that has high probability under the likelihood can be quite low, in effect wasting many particles. However, if we can analytically marginalize out the new rows of $\mathbf{Y}$, we can avoid sampling those values from the prior and instead

Algorithm 6 Particle filter for Infinite Binary Matrix Factorization

initialize L particles $[\mathbf{Z}_\ell^{(0)}, \mathbf{Y}_\ell^{(0)}]$, $\ell = 1, \ldots, L$
for $i = 1, \ldots, N$ **do**
  for $\ell = 1, \ldots, L$ **do**
    sample $\mathbf{Z}_\ell^{(i)}$ from $\mathbf{Z}_\ell^{(i-1)}$ using Algorithm 5
    calculate w_ℓ using Eqns. 4.12 and 4.3
  end for
  normalize particle weights
  resample particles according to weight CDF
  for $\ell = 1, \ldots, L$ **do**
    sample $\mathbf{Y}_\ell^{(i)}$ from $P(\mathbf{Y}_\ell^{(i)}|\mathbf{Z}_\ell^{(1:i)}, \mathbf{Y}_\ell^{(1:i-1)}, \mathbf{X}^{(1:i)})$
  end for
end for

sample them from the posterior, in effect saving many of the potentially wasted particles. If we let $\mathbf{Y}^{(1:i)}$ denote the rows of $\mathbf{Y}$ that correspond to the first i columns of $\mathbf{Z}$ and $\mathbf{Y}^{(i)}$ denote the rows (potentially more than 1) of $\mathbf{Y}$ that are introduced to match the new columns appearing in $\mathbf{Z}^{(i)}$, then we can write

$$P(\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i)}|\mathbf{X}^{(1:i)}) = P(\mathbf{Y}^{(i)}|\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)}, \mathbf{X}^{(1:i)})P(\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)}|\mathbf{X}^{(1:i)}) \tag{4.13}$$

where

$$P(\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)}|\mathbf{X}^{(1:i)}) \propto P(\mathbf{X}^{(i)}|\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)}, \mathbf{X}^{(1:i-1)})P(\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)}|\mathbf{X}^{(1:i-1)}). \tag{4.14}$$

Thus, we can use the particle filter to estimate the distribution $P(\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)}|\mathbf{X}^{(1:i)})$ (vs. $P(\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i)}|\mathbf{X}^{(1:i)})$) provided that we can find a way to compute $P(\mathbf{X}^{(i)}|\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)})$ and sample from the distribution $P(\mathbf{Y}^{(i)}|\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)}, \mathbf{X}^{(1:i)})$ to complete our particles.

The procedure described in the previous paragraph is possible in this model because, while our prior on $\mathbf{Y}$ is not conjugate to the likelihood, it is still possible to compute $P(\mathbf{X}^{(i)}|\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)})$. The entries of $\mathbf{X}^{(i)}$ are independent given $\mathbf{Z}^{(1:i)}$ and $\mathbf{Y}^{(1:i)}$. Since the entries in each column of $\mathbf{Y}^{(i)}$ will influence only a single entry in $\mathbf{X}^{(i)}$, this independence is maintained when we sum out $\mathbf{Y}^{(i)}$. This we have already seen in deriving an an analytic solution to $P(\mathbf{X}^{(i)}|\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)}) = \prod_d P(x_{i,d}|\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)})$ where $P(x_{i,d} = 1|\mathbf{Z}^{(1:i)}, \mathbf{Y}^{(1:i-1)})$ was given before in Equation 4.8. This gives us the likelihood we need for reweighting particles $\mathbf{Z}^{(1:i)}$ and $\mathbf{Y}^{(1:i-1)}$. The posterior distribution on $\mathbf{Y}^{(i)}$ is straightforward to compute by combining the likelihood in Equation 4.3 with the prior $P(\mathbf{Y})$. The particle filtering algorithm for this model is given in Algorithm 6.

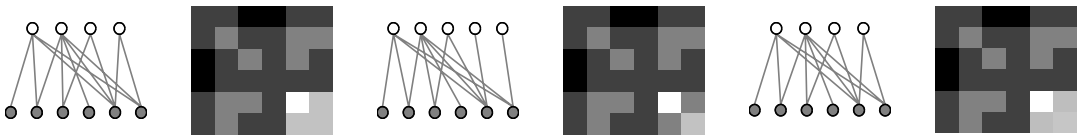

Figure 4.2: Infinite binary matrix factorization results. On the left is ground truth, the causal graph representation of $\mathbf{Z}$ and $\mathbf{Z}\mathbf{Z}^T$. The middle and right are particle filtering results; a single random particle $\mathbf{Z}$ and $E[\mathbf{Z}\mathbf{Z}^T]$ from a 100 and 1000 particle run middle and right respectively.

4.3.3 Experiments

We compared the particle filter in Algorithm 6 with Gibbs sampling on a dataset generated from the model described above, using the same Gibbs sampling algorithm and data generation procedure as developed in [Wood et al., 2006b]. We took $K_+ = 6$ and $N = 12$, running the IBP multiple times with $\alpha = 3$ until a matrix $\mathbf{Z}$ of correct dimensionality (12×6) was produced. This matrix is shown in Fig. 4.2 as a bipartite graph, where the observed variables are shaded. A 6×500 random matrix $\mathbf{Y}$ was generated with $p = 0.2$. The observed matrix $\mathbf{X}$ was then sampled from Eqn. 4.3 with parameters $\lambda = .9$ and $\epsilon = .01$.

Fig. 4.3 compares results from the particle filter and Gibbs sampler for this model. The performance of the models was measured by comparing a general error metric computed over the posterior distributions estimated by each approach. The error metric (the vertical axis in Figs. 4.5 and 4.3) was computed by taking the expectation of the matrix $\mathbf{Z}\mathbf{Z}^T$ over the posterior samples produced by each algorithm and taking the summed absolute difference (i.e. L_1 norm) between the upper triangular portion of $E[\mathbf{Z}\mathbf{Z}^T]$ computed over the samples and the upper triangular portion of the true $\mathbf{Z}\mathbf{Z}^T$ (including the diagonal). See Fig. 4.2 for an illustration of the information conveyed by $\mathbf{Z}\mathbf{Z}^T$. This error metric measures the distance of the mean of the posterior to the ground-truth. It is zero if the mean of the distribution matches the ground truth. It grows as a function of the difference between the ground truth and the posterior mean, accounting both for any difference in the number of latent factors that are present in each observation and for any difference in the number of latent factors that are shared between all pairs of observations.

The particle filter was run using many different numbers of particles, P. For each value of P, the particle filter was run 10 times. The horizontal axis location of each errorbar in the plot is the mean CPU time as reported by Matlab for the corresponding number of particles P while the error bars indicate the standard deviation of the error. The Gibbs sampler was run for varying numbers of sweeps, with the initial 10% of samples being discarded. The

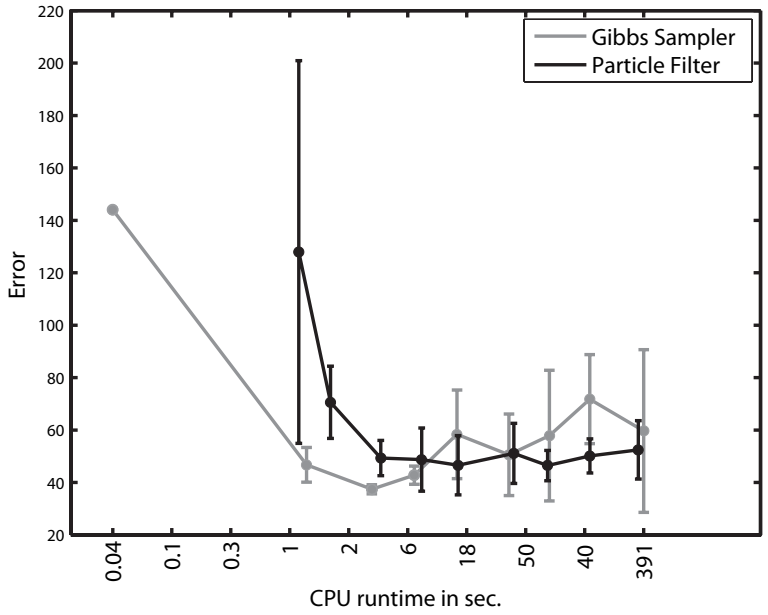

Figure 4.3: Performance results for particle filter vs. Gibbs sampling posterior estimation for the infinite binary matrix factorization model. Each point is an average over 10 runs with a particular number of particles or sweeps of the sampler $P = [1, 5, 10, 50, 100, 500, 1000, 2500, 5000]$ from left to right, and error bars indicate the standard deviation of the error.

number of Gibbs sampler sweeps was varied and the results are displayed in the same way as described for the particle filter above. The results show that the particle filter attains similar error to the Gibbs sampler in roughly the same amount of computational time. The difference in computational time here is due to implementation constants as the asymptotic complexity of each approach is equivalent. The advantage of the particle filtering approach here is that it constructs the posterior as observations are obtained; a characteristic that may be desireable for some applications.

4.4 A conjugate model: infinite linear-Gaussian matrix factorization

In this model, explained in detail in Griffiths and Ghahramani [2005], the entries of both $\mathbf{X}$ and $\mathbf{Y}$ are continuous. We report results on the modeling of image data of the same kind as was originally used to demonstrate the model in Griffiths and Ghahramani [2005]. Here each row of $\mathbf{X}$ is an image, each row of $\mathbf{Z}$ indicates the "latent features" present in

that image, such as the objects it contains, and each column of $\mathbf{Y}$ indicates the pixel values associated with a latent feature.

The likelihood for this image model is matrix Gaussian

$$P(\mathbf{X}|\mathbf{Z},\mathbf{Y},\sigma_x) = \frac{1}{(2\pi\sigma_X^2)^{ND/2}}\exp\{-\frac{1}{2\sigma_X^2}\mathrm{tr}((\mathbf{X}-\mathbf{ZY})^T(\mathbf{X}-\mathbf{ZY}))\}$$

where σ_X^2 is the noise variance. The prior on the parameters of the latent features is also Gaussian

$$P(\mathbf{Y}|\sigma_Y) = \frac{1}{(2\pi\sigma_Y^2)^{KD/2}}\exp\{-\frac{1}{2\sigma_Y^2}\mathrm{tr}(\mathbf{Y}^T\mathbf{Y})\}$$

with each element having variance σ_Y^2. Because both the likelihood and the prior are matrix Gaussian, they form a conjugate pair and $\mathbf{Y}$ can be integrated out to yield the collapsed likelihood,

$$P(\mathbf{X}|\mathbf{Z},\sigma_x) = \frac{1}{(2\pi)^{ND/2}\sigma_X^{(N-K_+)D}\sigma_Y^{K_+D}|\mathbf{Z}_+^T\mathbf{Z}_+\frac{\sigma_X^2}{\sigma_Y^2}\mathbf{I}_{K_+}|^{D/2}}\exp\{-\frac{1}{2\sigma_X^2}\mathrm{tr}(\mathbf{X}^T\mathbf{\Sigma}^{-1}\mathbf{X})\} \tag{4.15}$$

which is matrix Gaussian with covariance $\mathbf{\Sigma}^{-1} = \mathbf{I} - \mathbf{Z}_+(\mathbf{Z}_+^T\mathbf{Z} + \frac{\sigma_X^2}{\sigma_Y^2}\mathbf{I}_{K_+})^{-1}\mathbf{Z}_+^T$. Here $\mathbf{Z}_+ = \mathbf{Z}_{1:i,1:K+}$ is the first K_+ columns of $\mathbf{Z}$ and K_+ is the number of non-zero columns of $\mathbf{Z}$.

4.4.1 Particle filter posterior estimation

The use of a conjugate prior means that we do not need to represent $\mathbf{Y}$ explicitly in our particle filter. In this case the particle filter recursion shown in Eqns. 4.10 and 4.11 reduces to

$$P(\mathbf{Z}^{(1:i)}|\mathbf{X}^{(1:i)}) \propto P(\mathbf{X}^{(i)}|\mathbf{Z}^{(1:i)},\mathbf{X}^{(1:i-1)}) \sum_{\mathbf{Z}^{(1:i-1)}} P(\mathbf{Z}^{(1:i)}|\mathbf{Z}^{(1:i-1)})P(\mathbf{Z}^{(1:i-1)}|\mathbf{X}^{(1:i-1)})$$

and may be implemented as shown in Algorithm 7.

Reweighting the particles requires computing $P(\mathbf{X}^{(i)}|\mathbf{Z}^{(1:i)},\mathbf{X}^{(1:i-1)})$, the conditional probability of the most recent row of $\mathbf{X}$ given all the previous rows and $\mathbf{Z}$. Since $P(\mathbf{X}^{(1:i)}|\mathbf{Z}^{(1:i)})$ is matrix Gaussian we can find the required conditional distribution by following the standard rules for conditioning in Gaussians. Letting $\mathbf{\Sigma}_*^{-1} = \mathbf{\Sigma}^{-1}/\sigma_X^2$ be the covariance matrix for $\mathbf{X}^{(1:i)}$ given $\mathbf{Z}^{(1:i)}$, we can partition this matrix into four parts

$$\mathbf{\Sigma}_*^{-1} = \left[\begin{array}{c:c} \mathbf{A} & \mathbf{c} \\ \hdashline \mathbf{c}^T & b \end{array}\right]$$

Algorithm 7 Particle filter for Infinite Linear Gaussian Model

1: initialize L particles $[\mathbf{Z}_\ell^{(0)}]$, $\ell = 1, \ldots, L$
2: **for** $i = 1, \ldots, N$ **do**
3: **for** $\ell = 1, \ldots, L$ **do**
4: sample $\mathbf{Z}_\ell^{(1:i)}$ from $\mathbf{Z}_\ell^{(1:i-1)}$ using Algorithm 5
5: calculate w_ℓ using Eqns. 4.12 and 4.16
6: **end for**
7: normalize particle weights
8: resample particles according to weight cumulative distribution
9: **end for**

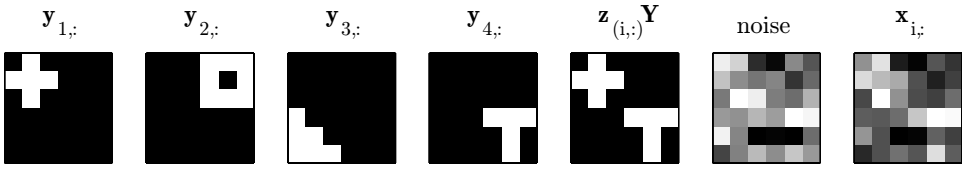

Figure 4.4: Generation of $\mathbf{X}$ under the linear Gaussian model. The first four images (left to right) correspond to the true latent features, i.e. rows of $\mathbf{Y}$. The fifth shows how the images get combined, with two source images added together by multiplying by a single row of $\mathbf{Z}$, $\mathbf{z}_{i,:} = [1\ 0\ 0\ 1]$. The sixth is Gaussian noise. The seventh image is the resulting row of $\mathbf{X}$.

where $\mathbf{A}$ is a matrix, $\mathbf{c}$ is a vector, and b is a scalar. Then the conditional distribution of $\mathbf{X}^{(i)}$ is

$$\mathbf{X}^{(i)}|\mathbf{Z}^{(1:i)}, \mathbf{X}^{(1:i-1)} \sim \text{Gaussian}(\mathbf{c}^T\mathbf{A}^{-1}\mathbf{X}^{(1:i-1)}, b - \mathbf{c}^T\mathbf{A}^{-1}\mathbf{c}). \tag{4.16}$$

This requires inverting a matrix $\mathbf{A}$ which grows linearly with the size of the data; however, $\mathbf{A}$ is highly structured and this can be exploited to reduce the cost of this inversion Barnett [1979].

4.4.2 Experiments

We compared the particle filter in Algorithm 7 with Gibbs sampling on an image dataset similar to that used in Griffiths and Ghahramani [2005]. We refer the reader to Griffiths and Ghahramani [2005] for the details of the Gibbs sampler for this model. As illustrated in Fig. 4.4, our ground-truth $\mathbf{Y}$ consisted of four different 6×6 latent images. A 100×4 binary ground-truth matrix $\mathbf{Z}$ was generated with by sampling from $P(z_{i,k} = 1) = 0.5$. The observed matrix $\mathbf{X}$ was generated by adding Gaussian noise with $\sigma_X = 0.5$ to each entry of $\mathbf{ZY}$.

Fig. 4.5 compares results from the particle filter and Gibbs sampler for this model. Comparison of the particle filter and Gibbs sampling was done using the procedure outlined in Section 4.3.3, producing somewhat different results: here the particle filter gave a better

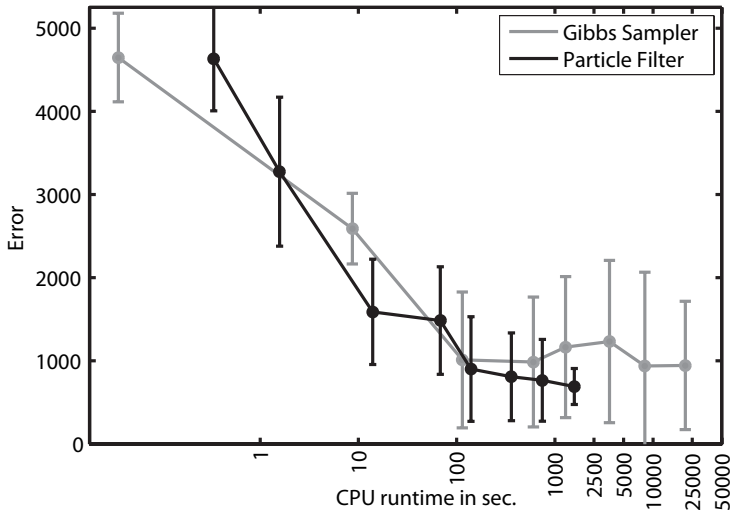

Figure 4.5: Performance results for particle filter vs. Gibbs sampling posterior estimation for the infinite linear Gaussian matrix factorization. Each point is an average over 10 runs with a particular number of particles or sweeps of the sampler $P = [1, 10, 100, 500, 1000, 2500, 5000]$ left to right, and error bars indicate the standard deviation of the error.

approximation to the posterior distribution in less time, as shown in Fig. 4.5. The results show that the particle filter attains low error in significantly less time than the Gibbs sampler, with the difference being an order or magnitude or more in most cases. This is a result of the fact that in the particle filter the number of new ones to add to a row of $\mathbf{Z}$ is proposed from the prior requiring only one evaluation of the likelihood. In the Gibbs sampler determining how many new ones to add to a row of $\mathbf{Z}$ requires evaluating the likelihood many times, up to the truncation level in computing the distribution over K^{new}.

4.5 Conclusion

In this chapter we have introduced particle filter posterior estimation for non-parametric Bayesian matrix factorization models based on the Indian buffet process. This approach is applicable to any Bayesian matrix factorization model with a sparse recursively decomposable prior. We have applied this approach with two different models, one with a conjugate prior and one with a non-conjugate prior, finding both comparable to Gibbs sampling.

However, more work needs to be done to explore the strengths and weakneses of these algorithms. In particular, simple sequential importance resampling is known to break down when applied to datasets with many observations, although we are optimistic that methods for addressing this problem that have been developed for Dirichlet process mixture models (e.g., [Fearnhead, 2004]) will also be applicable in this setting. By exploring the strengths and weaknesses of different methods for approximate inference in these models, we hope to come closer to our ultimate goal of making nonparametric Bayesian matrix factorization into a tool that can be applied on the scale of real world problems.

Chapter 5

Application: Stroke databank modeling

In this chapter we demonstrate our NPB binary matrix factorization model in a causal structure learning context. In particular we estimate a causal model for stroke signs exhibited by a small population of patients. From this model we can answer questions like, "How many types of strokes are there?" and "Are these signs manifestations of the same stroke localization?" all in a way that remains maximally agnostic in the same way as for all the previous NPB models presented in this dissertation.

5.1 Causal structure learning

A variety of methods from Bayesian statistics have been applied to the problem of learning the dependencies among a set of observed variables [Friedman and Koller, 2000; Heckerman, 1998]. However, in many settings, the dependencies of interest are not those that exist among the observed variables, but those that are produced by "hidden causes". For example, in medicine, the symptoms of patients are explained as the result of diseases that are not themselves directly observable – an assumption that is embodied in graphical models for medical diagnosis, such as QMR-DT [Shwe et al., 1991]. Here we consider how Bayesian methods can be used to infer both the existence of such hidden causes and how they influence observed variables. In our medical example, this would mean discovering diseases from the symptoms of patients.

Learning the structure of graphical models containing hidden causes presents a significant challenge, since the number of hidden causes is unknown and potentially unbounded.

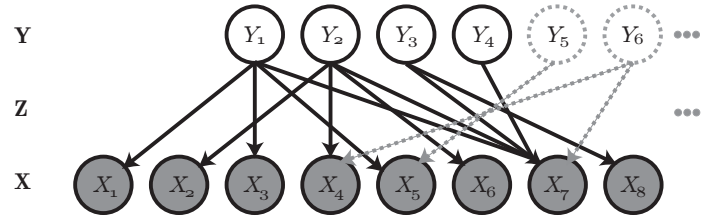

Figure 5.1: A hypothetical Bayesian network connecting hidden causes $Y_1, \ldots, Y_K$ to observed variables $X_1, \ldots, X_N$. We consider the case where the number of hidden causes, K, is unbounded. The state of the hidden causes, the observed variables, and the dependencies between them can all be summarized using binary matrices, being $\mathbf{Y}$, $\mathbf{X}$, and $\mathbf{Z}$ respectively.

Researchers have explored several approaches to this problem. One approach uses statistical criteria to identify when hidden causes might be present (e.g., [Elidan and Friedman, 2005]). While these algorithms are effective, they do not reflect our aim of developing a Bayesian approach to solving this problem. Other more closely related work defines a prior over the number of hidden causes, and uses "reversible jump" Markov chain Monte Carlo (RJMCMC) [Green, 1995] algorithms to move between structures with different numbers of hidden causes (e.g., [Courville et al., 2005; Orban et al., 2006]). These methods satisfy our desire for a Bayesian solution, but designing well-mixing RJMCMC algorithms can be difficult.

Previous Bayesian approaches to inferring hidden causal structure assume that the number of hidden causes is finite. In many cases, it is more accurate to assume instead that the number of hidden causes is infinite. Rather than seeking to determine the number of hidden causes, we instead seek to find and count the finite subset of these hidden causes that manifest in a particular finite dataset. This perspective on the dimensionality of models is common in non-parametric Bayesian statistics. For example, Dirichlet process mixture models assume that data come from a potentially infinite number of clusters, of which only a finite subset are observed [Antoniak, 1974; Neal, 2000; Rasmussen, 2000]. However, non-parametric Bayesian methods have not previously been applied to the problem of learning causal structure from data.

In this chapter we develop a non-parametric Bayesian approach to structure learning with an unbounded number of hidden causes. Specifically, we define a prior over causal structures using the Indian buffet process (IBP) [Griffiths and Ghahramani, 2005], a distribution over infinite binary matrices. Using the properties of the IBP, we derive a Gibbs sampling algorithm that can be used to sample from the posterior distribution over causal structures. We compare this approach to standard RJMCMC methods, and use it to infer

the hidden causes behind the symptoms of stroke patients.

5.2 Modeling hidden causal structure

Assume we have observed T instances (trials) of a set of N binary variables. Using $x_{i,t}$ to denote the value of the i^{th} observed variable on the t^{th} trial, we can summarize these observations with an $N \times T$ binary matrix $\mathbf{X}$. A standard structure learning task would be to learn a Bayesian network representation of the dependencies among the variables $X_1, \ldots, X_N$. These dependencies can be expressed using the $N \times N$ adjacency matrix $\mathbf{A}$ of a directed graph in which the nodes correspond to the N variables, with $a_{ij} = 1$ if an edge exists from node j to node i, and 0 otherwise.

To extend this problem to include hidden causes, assume that there are a further K binary variables which are never observed. Using $y_{k,t}$ to denote the value of the k^{th} hidden cause on the t^{th} trial, we can summarize the state of the hidden causes with a $K \times T$ binary matrix $\mathbf{Y}$. If we let $z_{i,k} = 1$ if an edge exists from node k to node i, and 0 otherwise, we can represent the dependencies between hidden causes and observable variables with the $N \times K$ binary matrix $\mathbf{Z}$. The full adjacency matrix $\mathbf{G}$ for the Bayesian network defined on the variables $X_1, \ldots, X_N$ and $Y_1, \ldots, Y_K$ is

$$\mathbf{G} = \begin{bmatrix} \mathbf{A} & \mathbf{Z} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{5.1}$$

where $\mathbf{0}$ denotes a matrix of zeros of the appropriate size to make $\mathbf{G}$ a square $(N + K) \times (N + K)$ matrix.

Our focus on this chapter will be on learning $\mathbf{Z}$, as Bayesian methods for learning $\mathbf{A}$ (e.g., [Friedman and Koller, 2000]) can easily be combined with the methods described here to infer $\mathbf{G}$ as a whole. Our problem, then, reduces to learning the structure of a bipartite graph (see Figure 5.1).

5.3 Generative model

Our goal is to infer the dependencies between the hidden causes and the observed variables, $\mathbf{Z}$, and the values of those hidden causes on each trial, $\mathbf{Y}$, from the values of the observed variables, $\mathbf{X}$. If we define a generative model specifying a distribution over $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$,

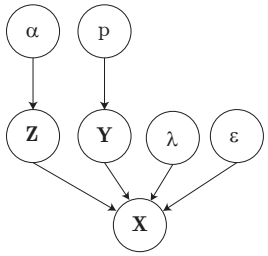

Figure 5.2: Graphical model for hidden cause matrix factorization model

we can compute the posterior distribution over $\mathbf{Z}$ and $\mathbf{Y}$ given $\mathbf{X}$ using Bayes' rule

$$P(\mathbf{Z}, \mathbf{Y}|\mathbf{X}) = \frac{P(\mathbf{X}|\mathbf{Y}, \mathbf{Z})P(\mathbf{Y})P(\mathbf{Z})}{\sum_{\mathbf{Y},\mathbf{Z}} P(\mathbf{X}|\mathbf{Y}, \mathbf{Z})P(\mathbf{Y})P(\mathbf{Z})} \tag{5.2}$$

where we make the independence assumptions illustrated in Figure 5.2. We will start by assuming K is finite, and then consider the case where $K \to \infty$.

5.3.1 A finite model

We assume that the entries of $\mathbf{X}$ are conditionally independent given $\mathbf{Z}$ and $\mathbf{Y}$, and are generated from a *noisy-OR* [Pearl, 1988] distribution, with

$$P(x_{i,t} = 1|\mathbf{Z}, \mathbf{Y}) = 1 - (1 - \lambda)^{\mathbf{z}_{i,:}\mathbf{y}_{:,t}}(1 - \epsilon) \tag{5.3}$$

where $\mathbf{z}_{i,:}$ is the i^{th} row of $\mathbf{Z}$, $\mathbf{y}_{i,:}$ is the i^{th} column of $\mathbf{Y}$, and $\mathbf{z}_{i,:}\mathbf{y}_{:,t} = \sum_{k=1}^{K} z_{i,k}y_{k,t}$. The baseline probability that $x_{i,t} = 1$ is ϵ, and λ is the probability with which any of the hidden causes is effective. This model makes sense in applications where many causes can elicit an effect, and the likelihood of observing an effect is increased by the number of hidden causes that are active. The medical diagnosis application we consider later in the chapter fits this description well.

We assume that the entries of $\mathbf{Y}$ are generated independently from a Bernoulli(p) distribution,

$$P(\mathbf{Y}) = \prod_{k,t} p^{y_{k,t}}(1-p)^{1-y_{k,t}} \tag{5.4}$$

where the product ranges over all values of k from 1 to K and all values of t from 1 to T. This model makes only the very general assumption that the baseline "prevalence" of all hidden causes is roughly the same. This assumption may not be appropriate for some applications, and can be relaxed if necessary.

In specifying a distribution on $\mathbf{Z}$, our goal is to generate matrices that allow multiple hidden causes to affect the same observed variable. This characteristic is desirable in many settings, and is exemplified by the case of medical diagnosis, where multiple diseases can cause the same symptom. A simple process for generating such a $\mathbf{Z}$ would be to assume that each hidden cause k (corresponding to a column of the matrix) is associated with a parameter θ_k, and then sample the values of z_{ik} from a Bernoulli(θ_k) distribution for i ranging from 1 to N. If we make the further assumption that each θ_k is generated from a Beta($\frac{\alpha}{K}$, 1) distribution and integrate out the θ_k, the probability of $\mathbf{Z}$ is

$$P(\mathbf{Z}) = \prod_{k=1}^{K} \frac{\frac{\alpha}{K}\Gamma(m_k + \frac{\alpha}{K})\Gamma(N - m_k + 1)}{\Gamma(N + 1 + \frac{\alpha}{K})} \tag{5.5}$$

where $\Gamma(\cdot)$ is the generalized factorial function, with $\Gamma(x) = (x-1)\Gamma(x-1)$, and $m_k = \sum_i z_{i,k}$.

5.3.2 Taking the infinite limit

In non-parametric Bayesian statistics, it is common to define models with unbounded dimensionality by taking the infinite limit of models with finite dimensionality [Neal, 2000; Rasmussen, 2000]. In this spirit, we can consider what happens to the model defined above as $K \to \infty$. The distribution on $\mathbf{X}$ remains well-defined, and the only values of $\mathbf{Y}$ with which we need be concerned are those in rows that correspond to columns of $\mathbf{Z}$ for which $m_k > 0$. Thus, we need only consider what happens to Equation 5.5 as $K \to \infty$. If we attend only to the columns for which $m_k > 0$ and define a scheme for ordering those columns, we obtain the distribution

$$P(\mathbf{Z}) = \frac{\alpha^{K_+}}{\prod_{h=1}^{2^N-1} K_h!} \exp\{-\alpha H_N\} \prod_{k=1}^{K_+} \frac{(N - m_k)!(m_k - 1)!}{N!}$$

where K_+ is the number of columns for which $m_k > 0$, K_h is the number of columns whose entries correspond to the binary number h, and $H_N = \sum_{i=1}^{N} \frac{1}{i}$ [Griffiths and Ghahramani, 2005].

This distribution can be shown to result from the *Indian buffet process*, defined in terms of a sequence of N customers entering a restaurant and choosing from an infinite array of dishes (corresponding to columns of $\mathbf{Z}$). The first customer tries the first Poisson(α) dishes (placing 1's in the appropriate columns). The remaining customers then enter one by one and pick previously sampled dishes with probability $\frac{m_{-i,k}}{i}$, where $m_{-i,k}$ is the number of customers who have already chosen the k^{th} dish. After trying the shared dishes, each customer also then tries the next Poisson($\frac{\alpha}{i}$) new dishes. The distribution that results from this process is *exchangeable*: the probability of each binary matrix $\mathbf{Z}$ is unaffected by the order of the customers.

5.4 Inference algorithms

Having defined a generative model, we can use Bayesian inference to infer $\mathbf{Z}$ and $\mathbf{Y}$ from $\mathbf{X}$. However, since the denominator of Equation 5.2 is an intractable sum, an approximate inference algorithm must be used. We present two such algorithms: a RJMCMC algorithm for the model with finite but unknown dimensionality and a Gibbs sampler for the model with an unbounded number of hidden causes.

5.4.1 Reversible jump MCMC posterior estimation

Reversible jump MCMC is a variant of the Metropolis-Hastings algorithm that allows moves between models of different dimensionality [Green, 1995]. The central idea is to augment a sampler for a finite model with a "dimension-shifting" move. In our case, a standard "birth/death" proposal would be of the following form: pick a single hidden cause (column k of $\mathbf{Z}$) and check the number of incident edges m_k. If $m_k = 0$, then remove that cause and decrement K. If $m_k > 0$, then add a new cause with no links, and generate the new values of $\mathbf{Y}$ that will correspond to it according to the prior. Letting ξ denote the values of $\mathbf{Z}$, $\mathbf{Y}$, and K, the move is accepted with probability

$$A(\xi^*, \xi) = \min\left[1, \frac{P(\mathbf{X}, \xi^*)}{P(\mathbf{X}, \xi)} \frac{Q(\xi|\xi^*)}{Q(\xi^*|\xi)}\right] \tag{5.6}$$

where ξ^* is the proposed value, ξ is the current value, and $Q(\xi^*|\xi)$ is the probability of proposing ξ^* given ξ. Making the dependency on K in our finite model explicit and defining

$P(K)$ to be the prior probability of K, we can factorize $P(\mathbf{X}, \xi)$ into the known probabilities $P(\mathbf{X}|\mathbf{Z}, \mathbf{Y})P(\mathbf{Y}|K)P(\mathbf{Z}|K)P(K)$.

Using this proposal, a hidden cause is added with probability K_+/K. An empty column is added to $\mathbf{Z}$, and a corresponding row of $\mathbf{Y}$ is generated by sampling according to Equation 5.4. The probability of proposing this new configuration of K, $\mathbf{Z}$, and $\mathbf{Y}$ is thus $(K_+/K)p^{\sum_t y_{k,t}}(1-p)^{\sum_t (1-y_{k,t})}$ where k is the index of the new column. To return to the previous configuration we may delete any hidden cause with the same value as our proposed new row of $\mathbf{Y}$. The probability of choosing such a row is $\delta/(K+1)$ where δ is the number of rows of $\mathbf{Y}$ (including the new row) which are identical to the proposed new row. Consequently, the ratio of proposal probabilities is

$$\frac{Q(\xi|\xi^*)}{Q(\xi^*|\xi)} = \frac{\frac{\delta}{K+1}}{\frac{K_+}{K}p^{\sum_t y_{k,t}}(1-p)^{\sum_t (1-y_{k,t})}}. \tag{5.7}$$

The ratio of the probabilities of the resulting configurations needs to take into account the difference in the probability of $\mathbf{Y}$, which is just the probability of the new row of $\mathbf{Y}$, $p^{\sum_t y_{k,t}}(1-p)^{\sum_t (1-y_{k,t})}$, the different probabilities of $\mathbf{Z}$ with and without the new column (with the corresponding changes in K), and the different probabilities of K. This gives the ratio

$$\frac{P(\mathbf{X}, \xi^*)}{P(\mathbf{X}, \xi)} = \frac{p^{\sum_t y_{k,t}}(1-p)^{\sum_t (1-y_{k,t})}P(\mathbf{Z}|K+1)P(K+1)}{P(\mathbf{Z}|K)P(K)}$$

Putting together this together with Equation 5.7 gives

$$A(\xi^*, \xi) = \min\left[1, \frac{\frac{\delta}{K+1}P(\mathbf{Z}|K+1)P(K+1)}{\frac{K_+}{K}P(\mathbf{Z}|K)P(K)}\right]. \tag{5.8}$$

A similar argument yields the acceptance probability for the proposal to delete a hidden cause with no links

$$A(\xi^*, \xi) = \min\left[1, \frac{\frac{K_+}{K-1}P(\mathbf{Z}|K-1)P(K-1)}{\frac{\delta}{K}P(\mathbf{Z}|K)P(K)}\right]. \tag{5.9}$$

To complete the specification of the algorithm, we need a scheme for sampling $\mathbf{Z}$ and $\mathbf{Y}$. We use Gibbs sampling, drawing each component of the two matrices from the distributions $P(z_{i,k}|\mathbf{X}, \mathbf{Z}_{-i,k}, \mathbf{Y})$ and $P(y_{k,t}|\mathbf{X}, \mathbf{Z}, \mathbf{Y}_{-k,t})$, where $\mathbf{Z}_{-i,k}$ is all values of $\mathbf{Z}$ except for $z_{i,k}$ and $\mathbf{Y}_{-k,t}$ is all of the values of the matrix $\mathbf{Y}$ except for $y_{k,t}$. As both $z_{i,k}$ and $y_{k,t}$ are

binary, these probabilities can be computed by enumeration. From our generative model and Bayes' rule we have

$$\begin{aligned} & P(z_{i,k} = a|\mathbf{X}, \mathbf{Z}_{-i,k}, \mathbf{Y}) \\ & \quad \propto \quad P(\mathbf{X}|\mathbf{Z}_{-i,k}, \mathbf{Y}, z_{i,k} = a) P(z_{i,k} = a|\mathbf{z}_{-i,k}) \end{aligned}$$

where $P(\mathbf{X}|\mathbf{Z}_{-i,k}, \mathbf{Y}, z_{i,k} = a)$ is specified by Equation 5.3, and $P(z_{i,k} = a|\mathbf{z}_{-i,k})$ results from our prior on $\mathbf{Z}$. It follows from Equation 5.5 that

$$P(z_{i,k} = 1|\mathbf{z}_{-i,k}) = \bar{\theta}_k = \frac{m_{-i,k} + \frac{\alpha}{K}}{N} \tag{5.10}$$

where $\mathbf{z}_{-i,k}$ is all of the entries of column k of $\mathbf{Z}$ except row i and $m_{-i,k} = \sum_{j \neq i}^{N} z_{j,k}$. Consequently, we obtain

$$\begin{aligned} & P(z_{i,k} = a|\mathbf{X}, \mathbf{Z}_{-i,k}, \mathbf{Y}) \\ & \quad \propto \quad \bar{\theta}_k^a (1 - \bar{\theta}_k)^{(1-a)} \prod_{t=1}^{T} (1 - (1 - \lambda)^{\mathbf{z}_{i,:} \cdot \mathbf{y}_{:,t}} (1 - \epsilon))|_{z_{i,k}=a} \end{aligned} \tag{5.11}$$

where $|_{z_{i,k}=a}$ means to replace $z_{i,k}$ with a in the preceding expression. Proceeding similarly for $y_{k,t}$ gives

$$\begin{aligned} & P(y_{k,t} = a|\mathbf{Z}, \mathbf{X}, \mathbf{Y}_{-k,t}) \\ & \quad \propto \quad p^a (1 - p)^{1-a} \prod_{i=1}^{N} (1 - (1 - \lambda)^{\mathbf{z}_{i,:} \cdot \mathbf{y}_{:,t}} (1 - \epsilon))|_{y_{k,t}=a}. \end{aligned} \tag{5.12}$$

This yields the RJMCMC sampler in Algorithm 8.

5.4.2 Gibbs sampler posterior estimation

When $K \to \infty$, we no longer require dimension-jumping moves, and can simply use a Gibbs sampler to infer $\mathbf{Y}$ and $\mathbf{Z}$. The only difference between the Gibbs sampler and the RJMCMC algorithm outlined above is the scheme for sampling $z_{i,k}$. For the the distributions required for the Gibbs sampler refer to Equations 4.9, 4.5, and 4.6 in Chapter 4.

Algorithm 8 RJMCMC sampler for hidden causes

```
for r = 1, ..., number of iterations do
  for i = 1, ..., N do
    randomly select column k of Z
    if m_{i,k} > 0 then
      propose adding a new cause
      accept according to Eqn. 5.8
    else
      propose deleting this unlinked cause
      accept according to Eqn. 5.9
    end if
    for k = 1, ..., K do
      sample z_{i,k} according to Eqn. 5.11
    end for
    for all y_{k,t} ∈ Y do
      sample y_{k,t} according to Eqn. 5.12
    end for
  end for
end for
```

5.5 Experiments

5.5.1 Synthetic data

We evaluated the RJMCMC algorithm for the finite model and the Gibbs sampler for the infinite model on two tasks using simulated data. First, we examined the ability of both algorithms to recover the true number of hidden causes used to generate a dataset. The data were generated by fixing the number of observations, N, and varying the number of hidden causes, K. For each value of K, 10 different datasets were generated by using rejection sampling to draw a matrix $\mathbf{Z}$ of the appropriate dimensionality from the IBP, drawing $\mathbf{Y}$ according to Equation 5.4, and then drawing $\mathbf{X}$ according to Equation 5.3. RJMCMC and Gibbs were both initialized with either an empty $\mathbf{Z}$ matrix ($K = 1$ for RJMCMC) or random $\mathbf{Z}$ and $\mathbf{Y}$ matrices with $K = K_+ = 10$, and then run for 500 iterations on each dataset. The other model parameters were fixed at $T = 500$, $\alpha = 3$, $\epsilon = 0.01$, $\lambda = 0.9$, and $p = 0.1$.

Our results are shown in Figure 5.3. The Gibbs sampler slightly over-estimates the number of hidden causes, but generally produces results that are close to the true dimensionality regardless of initialization. In contrast, RJMCMC appears to be affected by initialization. In particular, it systematically under-estimates the true dimensionality when initialized with $K = 1$. This is a result of poor mixing: while proposals to add hidden causes are often accepted, the new values of $\mathbf{Y}$ associated with those causes are typically inconsistent with the structure of $\mathbf{X}$, and consequently the new causes do not obtain links to observable

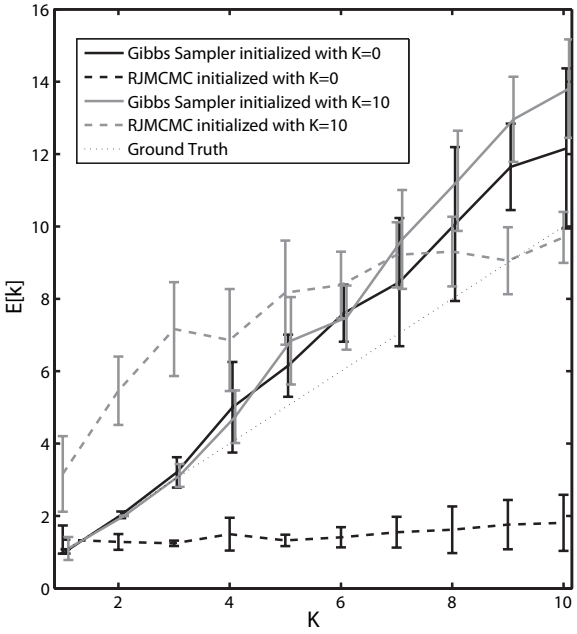

Figure 5.3: Learning the number of hidden causes using both RJMCMC and Gibbs sampling. Each line show the mean and standard deviation of the expected value of the dimensionality of the model (K for RJMCMC, and K_+ for Gibbs) taken over 500 iterations of sampling for each of 10 datasets.

nodes. The new causes are thus quickly deleted. A new cause might be generated with an appropriate set of $\mathbf{Y}$ values given sufficiently many sampling iterations (see Figure 5.4), but in a short run like that used here, slow mixing results in a strong influence of initialization.

Our second evaluation compared the ability of the two algorithms to recover specific structures. We manually specified four $\mathbf{Z}$ matrices, and then generated 10 datasets for each using the procedure outlined above with $T = 500$ $\alpha = 3$, $\epsilon = 0.01$, $\lambda = 0.9$, and $p = 0.1$. Both RJMCMC and Gibbs were initialized with an empty $\mathbf{Z}$ matrix ($K = 1$ for RJMCMC). We used two measures to evaluate performance. First, *in-degree error*, which we define to be the difference between the true in-degree and the expected in-degree of the observed nodes computed over samples. This is computed by taking the sum absolute difference between $\text{diag}(\mathbf{Z}\mathbf{Z}^T)$, the in-degree of the observable nodes, and $\text{diag}(E[\mathbf{Z}\mathbf{Z}^T])$, the expected in-degree computed over the samples. And second, the *structure error*, which we define to be the sum absolute difference between the upper triangular portion of $\mathbf{Z}\mathbf{Z}^T$ and $E[\mathbf{Z}\mathbf{Z}^T]$. Each element of the upper triangular portion of $\mathbf{Z}\mathbf{Z}^T$ is a count of the number of hidden causes shared by a pair of observable variables, the sum difference is a general measure of

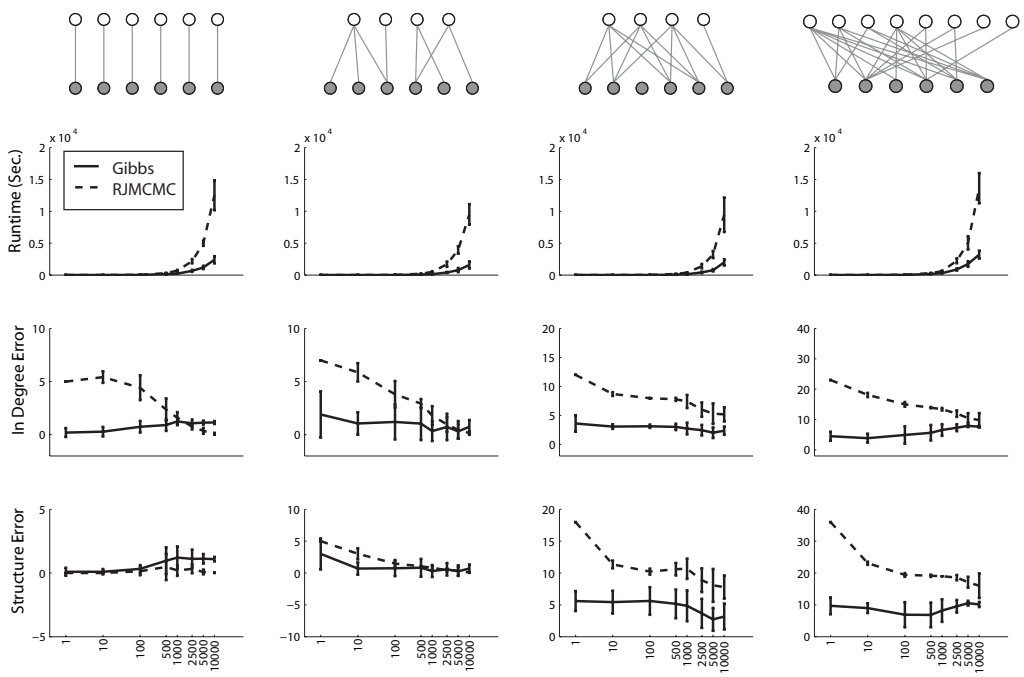

Figure 5.4: Recovering causal structure with RJMCMC and Gibbs sampling. From left to right the columns are results for a degree 1 bipartite graph ($K = 6$), disconnected graph ($K = 4$), an "undercomplete" random graph with fewer causes than observations ($K = 4$), and an "overcomplete" random graph with more causes than observations ($K = 8$). The top row shows the true structures, the second row shows mean runtime in wall clock seconds as a function of the number of iterations of sampling. The third and fourth rows show the in-degree error and structure error for each algorithm, as defined in the main text, on the same axis of number of iterations. Error bars are symmetric, and indicate one standard deviation over 10 datasets.

graph dissimilarity.

The results are shown in Figure 5.4. The Gibbs sampler consistently recovers a structure close to the truth, and does so in surprisingly few iterations. This reflects a tendency to move quickly to a good solution, and then minimally explore the space around that solution. The variance of the results grow slightly with more iterations, reflecting greater exploration of the space of structures. In contrast, RJMCMC performs poorly for all but the largest number of iterations. This is a reflection of the fact that it mixes slowly, taking a long time to increase the dimension of a model. One other feature of the plots bears explanation: the poor performance on the overcomplete graph is not so much a problem with the algorithms as an indication of an unavoidable problem with identifiability in overcomplete models. For instance, in this particular graph there is no information, short of the prior on $\mathbf{Y}$, that can

be used to distinguish causal nodes 5 and 6 from a hypothetical single combined node.

5.5.2 Mt. Sinai stroke databank

Here we learn a model of a subset of the Mount Sinai Stroke Data Bank [Tuhrim et al., 1991]. This data bank consists of stroke signs exhibited by patients admitted to an acute stroke unit at Mount Sinai Hospital, together with lesion localization evaluations made by neurologist with special stroke expertise (the data were collected from a standardized neurological assessment including a detailed neurologic examination). In the language of the preceding chapter, the signs are our observed variables, and the localizations are our hidden causes. The raw data bank consisted of 38 signs and 14 localizations. Some signs were left-right variables and some were graded in degrees of severity. For each patient, signs were binarized in two steps. First, graded signs like *decreased level of consciousness* and *comprehension deficit severity* were assigned a 1 if any level was indicated at all and 0 if no indication was made. Second, for "sided" signs like *visual field deficit* and *abnormal deep tendon reflex* we created two variables, one each for left and right, and assigned a 1 to the variable corresponding to the side on which the sign was observed. Of the resulting 56 sign variables, only 42 were expressed by at least one patient. The mean number of signs per patient was 8.24 and the mean number of stroke localizations was 1.96. Every localization was found in at least one patient, although there were five localizations that were found in only one patient.

Stroke databank modeling results

Although the ground truth localizations were known, we inferred the localizations, $\mathbf{Y}$, and their causal relationships to the signs, $\mathbf{Z}$, directly from the signs exhibited by the patients, $\mathbf{X}$, using our Gibbs sampler. In addition, we placed a Beta$(1,1)$ prior on λ, ϵ, and p, and a Gamma$(1,1)$ prior on α, and sampled them as well using Metropolis updates for λ and ϵ and Gibbs for p and α. These hyperparameters have interpretations: p measures of the incidence rate of localizations (the approximate ground truth for the data bank is $p = 0.14$), ϵ is the rate of spontaneous sign expression (noise), λ is a measure of how reliably a localization (or combination of localizations) gives rise to a sign, and α and K together measure of the number of hidden localizations (ground truth is $K = 14$).

Trace plots for the hyperparameters over 20,000 iterations of Gibbs sampling appear in Figure 5.5. Interpretation of these results should be considered under the caveat that there are few datapoints in this data bank, and that all the data is stroke-specific. The

posterior distribution on ϵ favored low values, suggesting that the prevalence of these signs in the absence of a particular stroke localization in these patients is low. That values of λ are reasonably high reflects the fact that the localizations responsible for producing a particular sign produce that sign with high probability. The posterior distributions on p and K favored values slightly higher and lower than the ground truth, respectively, but these parameters should be expected to be coupled, since they influence the overall prevalence of signs. The under-estimate of K is not unexpected, due to the paucity of data for many localizations. In fact, there were only nine localizations exhibited by at least two patients, providing a closer correspondence to the values favored by the sampler.

The causal structure with the highest posterior probability in our set of samples is shown in Figure 5.6. We can attempt to interpret the hidden causes by examining the signs to which they are connected. We showed the clusters of signs corresponding to the hidden causes found by the algorithm to a clinical neurologist familiar with the domain, who concluded that the localizations were somewhat general but not inappropriately confounded. These observations came with the caveat that the loss of degree information and the abridgement of the typical clinical sign and localization domain made precise localizations difficult. As a further encouraging sign, we note that the data bank contained no bilateral stroke sufferers, and the recovered graph reflects this by correctly separating signs that are caused by infarcts in each hemisphere.

5.6 Discussion

Our results suggest a number of future directions. First, while our focus here was on the case where p and λ were shared by all hidden causes, variations on the algorithm we describe could be applied in the case where these hyperparameters vary across causes, extending the applicability of the model. Second, the slow mixing exhibited by RJMCMC requires further investigation. While birth/death proposals of the kind we used here are common, it may be possible to develop a faster-mixing proposal by drawing inspiration from the Gibbs sampler for our non-parametric model, adding and deleting nodes with a single link attached.

The non-parametric Bayesian methods explored in this chapter make it possible to learn Bayesian networks with infinitely many nodes. While this might seem intractable at first glance, assuming that the number of nodes is unbounded actually removes the formal problems involved in inferring hidden causes, and leads to a simple algorithm with broad applicability.

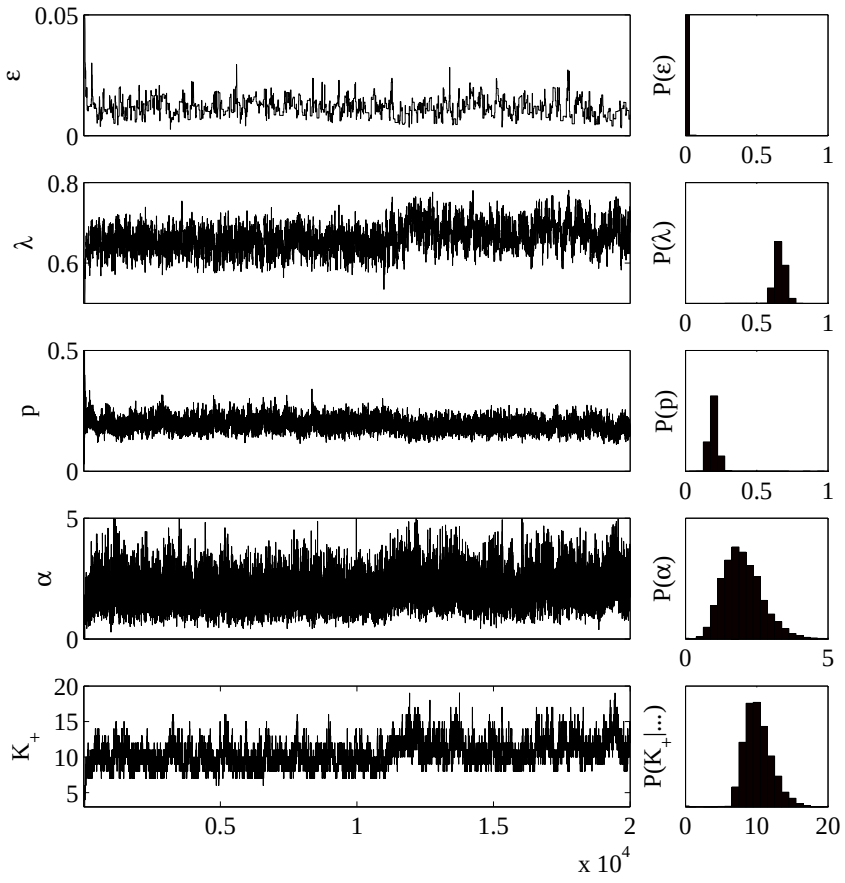

Figure 5.5: Trace plots and histograms for the Gibbs sampler applied to the signs exhibited by 50 stroke patients. The left column shows the current value of $\epsilon, \lambda, p, \alpha$, and K_+ as the sampler progressed, where K_+ is obtained by examining the current $\mathbf{Z}$ sample. The right column shows histograms of the same variables computed over the samples.

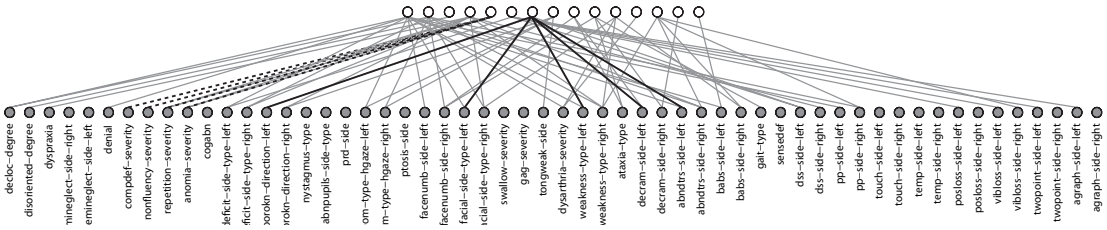

Figure 5.6: Causal structure with highest posterior probability. Two grouping of signs are highlighted. In solid black, we find a grouping of poor optokinetic nystagmus, lack of facial control, weakness, decreased rapid alternating movements, abnormal deep tendon reflexes all on the left side, consistent with a right frontal/parietal infarct. In dashed black, we find a grouping of comprehension deficit, non-fluency, repetition, and anomia generally consistent, in part, with a left temporal infarct.

Chapter 6

Conclusion and Future Work

Latent variable modeling is natural in settings where causal influences are not directly observable. Nonparametric Bayesian modeling is natural when the complexity of the model cannot be prespecified or should grow with the data. In this dissertation we have illustrated the utility of these approaches by demonstrating theoretical and practical improvements to spike sorting and subsequent neural data analyses. We also have developed a new NPB hidden causal structure model and demonstrated it in modeling neurological data.

There are several exciting avenues of future research that stem from the findings in this dissertation. The first are practical improvements to the spike sorting approach outlined in Chapter 3. Among them, accounting for refractory periods, modeling action potential waveform shape change over time, deconvolving overlapping spikes, and handling the appearance and disappearance of neurons. Specific approaches to each of these were mentioned in Chapter 3. On a more theoretical note, our comparison to reversible jump Markov chain Monte Carlo approaches was cursory. In the broader literature there is unfortunately little in the way of direct comparison between RJMCMC approaches and nonparametric approaches, excepting [Green and Richardson, 2001]. A more careful empirical and theoretical comparison of these techniques is warranted.

With respect to the binary hidden causal structure model and estimation algorithms of Chapter 4 there are a number of avenues of improvement to explore. The most immediate is the adoption of the particle resampling approach of Fearnhead and Clifford [2003] to the particle filter we developed for our nonparametric Bayesian matrix factorization model. This is straightforward to do and necessary for practical real-world application of our model. Additionally, demonstrating the model in more complex and data intensive application domains is of importance.

Thinking more broadly there is a large class of nonparametric Bayesian latent variable models that can be developed on the foundation of this and related work. A nonparametric Bayesian extension to nonnegative matrix factorization (NMF) [Lee and Seung, 2000] would address the model selection problem in NMF.

Additionally, a more full investigation of the interplay between sequential and batch posterior estimation in nonparametric Bayesian modeling is warranted. There is little prior work on moving from one posterior representation to the other [MacEachern et al., 1999]. Stick breaking representations and split merge samplers add yet another dimension of flexibility to mixed estimation schemes. By combining and leveraging the best of all of these sampling schemes it should be possible to greatly increase the practical applicability of NPB modeling approaches to real world problems.

One issue in particular confounds all models based on the Dirichlet process which is the tendency for such models to overestimate the number of hidden classes, or more generally to strongly prefer a logarithmic distribution over class cardinalities. While this may be appropriate for a large number of applications, it is certainly not so for all. Examing this feature of models based on the Dirichlet process and thinking about ways to impose different assumptions about the distribution of class cardinalities (perhaps uniform) may be important for successful development of NPB approaches for different application domains.

Finally, in this dissertation we have not considered in any great depth when and if NPB modeling is the right thing to do; instead we have demonstrated that for some applications such as modeling neural data doing so has its benefits. Unfortunately NPB modeling remains computationally expensive and many of the underlying assumptions may be inappropriate in some application domains, so this dissertation should not be interpretted as an unabashed advocation of NPB modeling.

That said we believe that NPB modeling approaches represent the best way forward for attacking many of the most interesting problems faced by the machine learning community today. For instance humans somehow learn in an unsupervised way how to identify both words from sound waves and objects from light impinging on the retina, but both of these tasks remain beyond the milieu of machine learning. It is probable that each of these processes requires automatic identification of groups of statistically similar phenomena, whether spoken words that usually sound nearly the same or objects that exhibit similar visual characteristics. Development of NPB modeling approaches for these applications is in its earliest stages and a great deal of work remains to be done; however, each problem exhibits characteristics that make them excellent candidates for NPB modeling.

Complimentarily NPB modeling is an interesting and elegant framework for addressing

questions about how biological computation achieves some of the things we often take for granted. For instance attempts to figure out a neurophysiological mechanism for robustly identifying groups of statistically similar observations might benefit from related work in the theory of nonparametric Bayesian modeling. As an example, good progress has been made in showing how inference algorithms such as belief propagation might be implemented in neural circuitry, but, for instance, neurophysiologically plausible mechanisms for learning the underlying graphical models remain elusive. Might nonparametric Bayesian approaches to this problem (i.e. inferring the existence and cardinality of latent variables) help guide our search for such a physical mechanism? We believe that this is quite likely to be the case.

Bibliography

T. N. Aflalo and M. S A. Graziano. Possible origins of the complex topographic organization of motor cortex: Reduction of a multidimensional space onto a two-dimensional array. *Journal of Neuroscience*, 26(23):6288–6297, 2006.

C. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2:1152–1174, 1974.

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

D. Blackwell and J. MacQueen. Ferguson distributions via Polya urn schemes. *The Annals of Statistics*, 1:353–355, 1973.

T. J. Blanche, M. A Spacek, J. F. Hetke, and N. V. Swindale. Polytrodes: High-density silicon electrode arrays for large-scale multiunit recording. *Journal of Neurophysiology*, 93:2987–3000, 2005.

D. Blei and M. Jordan. Variational inference for Dirichlet process mixtures. *Journal of Bayesian Analysis*, 1(1):121–144, 2005.

E. Brown, R. Kass, and P. Mitra. Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature Neuroscience*, 7:456–461, 2004.

C. A. Bush and S. N. MacEachern. A semi-parametric Bayesian model for randomized block designs. *Biometrika*, 83:275–286, 1996.

CKI. `http://www.cyberkineticsinc.com/`, 2005.

A. C. Courville, N. D. Daw, and D. S. Touretzky. Similarity and discrimination in classical conditioning: A latent variable account. In *Advances in Neural Information Processing Systems*, Cambridge, MA, 2005. MIT Press.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39:1–38, 1977.

A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. Wiley, New York, 2000.

G. Elidan and N. Friedman. Learning hidden variable networks: The information bottleneck approach. *Journal of Machine Learning Research*, 6:81–127, 2005.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, 1995.

P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Journal of Statistics and Computing*, 14:11–21, 2004.

P. Fearnhead and P. Clifford. Online inference for well-log data. *Journal of the Royal Statistics Society Series B*, 65:887–899, 2003.

M. S. Fee, P. P. Mitra, and D. Kleinfeld. Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-gaussian variability. *J. Neuroscience Methods*, 69: 175–188, 1996.

T. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1:209–230, 1973.

T. S. Ferguson. Bayesian density estimation by mixtures of normal distributions. In M. Rizvi, J. Rustagi, and D. Siegmund, editors, *Recent advances in statistics*, pages 287–302. Academic Press, New York, 1983.

C. Fraley and A. E. Raftery. Bayesian regularization for normal mixture estimation and model-based clustering. Technical Report 05/486, Department of Statistics, University of Washington, Seattle, Washington, 2005.

N. Friedman and D. Koller. Being Bayesian about network structure. In *Proceedings of the 16th Annual conference on uncertainty in AI*, pages 201–210. Morgan Kaufmann, Stanford, CA, 2000.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis*. Chapman & Hall, New York, 1995.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

A. Georgopoulos, J. Kalaska, R. Caminiti, and J. Massey. On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *Journal of Neuroscience*, 2:1527–1537, 1982.

A. Georgopoulos, A. Schwartz, and R. Kettner. Neuronal population coding of movement direction. *Science*, 233:1416–1419, 1986.

D. Görür, C. R. Rasmussen, A. S. Tolias, F. Sinz, and N.K. Logothetis. Modeling spikes with mixtures of factor analyzers. In *Proceeding of the DAGM Symposium*, pages 391–398. Springer, 2004.

D. Görür, F. Jäkel, and C. R. Rasmussen. A choice model with infinitely many latent features. In *Proceeding of the 23rd International Conference on Machine Learning*, 2006.

P. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732, 1995.

P. Green and S. Richardson. Modelling heterogeneity with and without the Dirichlet process. *Scandinavian Journal of Statistics*, 28:355–377, 2001.

J.E. Griffin and M.F.J. Steel. Order-based dependent Dirichlet processes. *Journal of the American Association of Statistics*, 101(473):179–194, 2006.

T. L. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. Technical Report 2005-001, Gatsby Computational Neuroscience Unit, 2005.

Tom Griffiths and Zoubin Ghahramani. Infinite latent feature models and the Indian buffet process. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA, 2006.

Peter Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

K. D. Harris, D. A. Henze, J. Csicsvari, H. Hirase, and G. Buzsáki. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *Journal of Neurophysiology*, 81(1):401–414, 2000.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

D. Heckerman. A tutorial on learning with Bayesian networks. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 301–354. MIT Press, Cambridge, MA, 1998.

E. Hulata, R. Segev, and E. Ben-Jacob. A method for spike sorting and detection based on wavelet packets and Shannon's mutual information. *Journal of Neuroscience Methods*, 117:1–12, 2002.

S. Jain and R. M. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1):158–182, March 2004.

I. T. Jolliffe. *Principal component analysis*. Springer, New York, 1986.

Michael I. Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the American Society of Mechanical Engineers, Journal of Basic Engineering*, 82:35–45, 1960.

KlustaKwik. `http://klustakwik.sourceforge.net/`, 2000.

K. P. Koerding and D. M. Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427:244–248, 2004.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 2000.

M. S. Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network*, 9(4):R53–78, 1998.

J. S. Liu. *Monte Carlo strategies in scienfitic computing*. Springer Verlag, New York, NY, 2001.

S. MacEachern and P. Muller. Estimating mixture of Dirichlet process models. *Journal of Computational and Graphical Statistics*, 7:223–238, 1998.

S. N. MacEachern, M. Clyde, and J. Liu. Sequential importance sampling for nonparametric Bayes models: the next generation. *The Canadian Journal of Statistics*, 27:251–267, 1999.

M. Meila. Comparing clusterings. Technical Report 418, Department of Statistics, University of Washington, Seattle, Washington, 2002.

A. W. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21: 1087–1092, 1953.

B. Mirkin. *Mathematical classification and clustering*. Kluwer Academic Publishers, New York, 1996.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000.

R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. Technical Report 9815, Department of Statistics, University of Toronto, 1998.

R. M. Neal and G. E. Hinton. A view EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in graphical models*. MIT Press, Cambridge, MA, 1998.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2001.

D.P. Nguyen, L.M. Frank, and E.N. Brown. An application of reversible-jump Markov chain Monte carlo to spike classification of multi-unit extracellular recordings. *Network*, 14(1): 61–82, 2003.

G. Orban, J. Fiser, R. N. Aslin, and M. Lengyel. Bayesian model learning in human visual perception. In *Advances in Neural Information Processing Systems 18*, pages 1043–1050, Cambridge, MA, 2006. MIT Press.

J. Pearl. *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann, San Francisco, CA, 1988.

J. Pitman. Combinatorial stochastic processes, 2002. Notes for Saint Flour Summer School.

Plexon Inc. `http://www.plexoninc.com/OFS.htm`, 2003.

G. Radons, J. D. Becker, B. Dülfer, and J. Krüger. Analysis, classification, and coding of multielectrode spike trains with hidden markov models. *Biological Cybernetics*, 71:359, 1994.

C. Rasmussen. The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems 12*. MIT Press, Cambridge, MA, 2000.

S. Richardson and P. J. Green. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society*, 59(4):731–792, 1997.

M. Sahani, J. S. Pezaris, and R. A. Andersen. On the separation of signals from neighboring cells in tetrode recordings. In *Advances in Neural Information Processing Systems 10*, pages 222–228. MIT Press, 1998.

A. N. Sanborn, T. L. Griffiths, and D. J. Navarro. A more rational model of categorization. In *Proceedings of the 28th Annual Conference of the on Cognitive Science Socity*, 2006.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

M. Serruya, N. Hatsopoulos, M. Fellows, L. Paninski, and J. Donoghue. Robustness of neuroprosthetic decoding algorithms. *Biological Cybernetics*, 88(3):201–209, 2003.

M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue. Brain-machine interface: Instant neural control of a movement signal. *Nature*, 416:141–142, 2002.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

S. Shoham, M. R. Fellows, and R. A. Normann. Robust, automatic spike sorting using mixtures of multivariate t-distributions. *Journal of Neuroscience Methods*, 127(2):111–122, 2003.

M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base I. the probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–255, 1991.

N. Srebro and S. Roweis. Time-varying topic models using dependent Dirichlet processes. Technical Report UTML TR 2005-003, University of Toronto, 2005.

E. Sudderth, A. Torralba, W. Freeman, and A. Willsky. Describing visual scenes using transformed Dirichlet processes. In *Advances in Neural Information Processing Systems*, pages 1297–1304, Cambridge, MA, 2005. MIT Press.

S. Takahashi, Y. Anzai, and Y. Sakurai. Automatic sorting for multi-neuronal activity recorded with tetrodes in the presence of overlapping spikes. *Journal of Neurophysiology*, 89:2245–2258, 2003.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. Technical Report 653, Department of Statistics, University of California at Berkeley, 2004.

S. Tuhrim, J. Reggia, and S. Goodall. An experimental study of criteria for hypothesis plausibility. *Journal of Experimental and Theoretical Artificial Intelligence*, 3:129–144, 1991.

Y. Weiss, E. P. Simoncelli, and E. H. Adelson. Motion illusions as optimal percepts. *Nature Neuroscience*, 5:598–604, 2002.

M. West, P. Muller, and M. Escobar. Hierarchical priors and mixture models, with application in regression and density estimation. In P. Freeman and A. Smith, editors, *Aspects of Uncertainty*, pages 363–386. Wiley, New York, 1994.

F. Wood and T. L. Griffiths. Particle filtering for nonparametric Bayesian matrix factorization. In *Advances in Neural Information Processing Systems*, page to appear, Cambridge, MA, 2007. MIT Press.

F. Wood, M. J. Black, C. Vargas-Irwin, M. Fellows, and J. P. Donoghue. On the variability of manual spike sorting. *IEEE Transactions on Biomedical Engineering*, 51(6):912–918, June 2004a.

F. Wood, M. Fellows, J. P. Donoghue, and M. J. Black. Automatic spike sorting for neural decoding. In *Proceedings of the 27th IEEE Conference on Engineering in Medicine and Biological Systems*, pages 4126–4129, 2004b.

F. Wood, S. Goldwater, and M. J. Black. A non-parametric Bayesian approach to spike sorting. In *Proceedings of the 28th IEEE Conference on Engineering in Medicine and Biological Systems*, pages 1165–1169, 2006a.

F. Wood, T. L. Griffiths, and Z. Ghahramani. A non-parametric Bayesian method for inferring hidden causes. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 536–543, 2006b.

W. Wu, M. J. Black, Y. Gao, E. Bienenstock, M. Serruya, A. Shaikhouni, and J. P. Donoghue. Neural decoding of cursor motion using a Kalman filter. In *Advances in Neural Information Processing Systems 15*, pages 133–140. MIT Press, 2003.

W. Wu, Y. Gao, E. Bienenstock, J. P. Donoghue, and M. J. Black. Bayesian population coding of motor cortical activity using a kalman filter. *Neural Computation*, 18:80–118, 2005.