ABSTRACT

Stochastic Inference and Bayesian Nonparametric Models in Electrophysiological Time Series

by

David Carlson

Department of Electrical and Computer Engineering
Duke University

Date: ______________________

Approved:

___________________________
Lawrence Carin, Supervisor

___________________________
Guillermo Sapiro

___________________________
Galen Reeves

___________________________
Katherine Heller

___________________________
Kafui Dzirasa

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University 2015
Copyright © 2015 by David Carlson
All rights reserved except the rights granted by the
Creative Commons Attribution-Noncommercial Licence
Abstract

This thesis presents novel methods for processing electrophysiological time-series from simultaneously recorded electrodes in a brain, as well as providing new inference techniques that are more generally applicable. On spike sorting, I introduce Bayesian nonparametric methods to process multiple electrodes simultaneously, which improves performance when the electrode spacing is less than 100 microns. Furthermore, by treating the spike sorting problem as a single deconvolutional model instead of the conventional 2-step procedure with detection and clustering steps, the overlapping spike problem is ameliorated. I then show that these detected neurons and their spike trains have dynamic relationships with local field potentials in distinct brain regions, and that the number of distinct relationships appears to cluster.

While these models approach an important scientific problem, it is necessary to have efficient inference in computationally-intensive models. To this end, I introduce novel methods for Variational Bayesian inference, as well as introducing a new stochastic inference algorithm called “Stochastic Spectral Descent,” which mimics Stochastic Gradient Descent but operates in the Shatten-infinity norm. I show that several common machine learning problems naturally operate in the Shatten-infinity norm, and that this descent method mimics the natural geometry and greatly improves learning efficiency.
## Contents

Abstract iv  
List of Tables xi  
List of Figures xii  
List of Abbreviations and Symbols xviii  
Acknowledgements xx  

1 Introduction 1  
  1.1 Overview ............................................. 1  
  1.2 Graphical Models for Deep Learning and Stochastic Inference ..... 3  
    1.2.1 Restricted Boltzmann Machines ............................ 3  
    1.2.2 Sigmoid Belief Networks ............................... 5  
    1.2.3 Stochastic Gradient Descent for RBM and SBN .............. 6  
  1.3 Spike Sorting .......................................... 8  
    1.3.1 Spike Detection ................................... 9  
    1.3.2 Feature extraction .................................. 13  
    1.3.3 Spike Sorting ..................................... 14  
    1.3.4 Combining spike detection and clustering .................. 19  

2 Stochastic Spectral Descent for Restricted Boltzmann Machines 25  
  2.1 Introduction .......................................... 25  
  2.2 Preliminaries and Model Definitions .......................... 27
4 The Focused Mixture Model for Spike Sorting

4.1 Introduction ......................................................... 63
4.2 Models and Analysis ............................................. 65
  4.2.1 Model Concept ................................................. 65
  4.2.2 Bayesian dictionary learning ................................. 67
  4.2.3 Mixture modeling .............................................. 69
  4.2.4 Focused Mixture Model ....................................... 69
  4.2.5 Computations .................................................. 71
  4.2.6 Data Acquisition and Pre-processing ....................... 73
  4.2.7 Evaluation Criteria .......................................... 74
4.3 Results .............................................................. 75
  4.3.1 Real data with partial ground truth ......................... 75
  4.3.2 Longitudinal analysis of electrophysiological data .......... 77
  4.3.3 Handling missing data ........................................ 83
  4.3.4 Model tuning .................................................. 85
  4.3.5 Sparsely Firing Neurons ..................................... 87
  4.3.6 Computational requirements .................................. 89
4.4 Discussion .......................................................... 90
  4.4.1 Summary ...................................................... 90
  4.4.2 Future Directions .............................................. 91
5 OPASS: Spike Sorting via a Bayesian Nonparametric Deconvolutional Process

5.1 Introduction ......................................................... 93
5.2 Model .............................................................. 95
  5.2.1 Modeling the continuous-time output of a single neuron .... 95
  5.2.2 A nonparametric model of population activity ............... 97
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1.2</td>
<td>Proof of Theorem 2</td>
<td>132</td>
</tr>
<tr>
<td>A.1.3</td>
<td>Proof of Theorem 3</td>
<td>134</td>
</tr>
<tr>
<td>A.2</td>
<td>Derivation of optimal steps</td>
<td>135</td>
</tr>
<tr>
<td>A.3</td>
<td>Discussion of using $\ell_2$ bound instead of $\ell_\infty$ bound on $lse$ function</td>
<td>136</td>
</tr>
<tr>
<td>B</td>
<td>Proofs and Supplemental Material of Descent Algorithms of Deep Discrete Graphical Models</td>
<td>139</td>
</tr>
<tr>
<td>B.1</td>
<td>Proof of Theorem 4</td>
<td>139</td>
</tr>
<tr>
<td>B.2</td>
<td>Proof of Theorem 5</td>
<td>141</td>
</tr>
<tr>
<td>B.3</td>
<td>Proof of Equation 3.4</td>
<td>142</td>
</tr>
<tr>
<td>B.4</td>
<td>Proofs of Sigmoid Belief Net Results</td>
<td>143</td>
</tr>
<tr>
<td>B.5</td>
<td>Proofs of Replicated Softmax-Restricted Boltzmann Machine Results</td>
<td>143</td>
</tr>
<tr>
<td>B.5.1</td>
<td>Derivation of optimal steps with a maximum $\ell_1$ row penalty</td>
<td>144</td>
</tr>
<tr>
<td>B.6</td>
<td>Proofs of Replicated Softmax Belief Net Results</td>
<td>145</td>
</tr>
<tr>
<td>B.7</td>
<td>Proofs of Deep Belief Net</td>
<td>146</td>
</tr>
<tr>
<td>C</td>
<td>Supplemental Material for the Focused Mixture Model</td>
<td>147</td>
</tr>
<tr>
<td>C.1</td>
<td>Relationship between the FMM and other Bayesian Nonparametric Models</td>
<td>147</td>
</tr>
<tr>
<td>C.2</td>
<td>Relationship between the FMM and Dirichlet priors</td>
<td>148</td>
</tr>
<tr>
<td>C.3</td>
<td>Other Formulations of the FMM</td>
<td>149</td>
</tr>
<tr>
<td>C.4</td>
<td>Additional Connections to Other Bayesian Models</td>
<td>151</td>
</tr>
<tr>
<td>C.5</td>
<td>Proof of Lemma 6</td>
<td>153</td>
</tr>
<tr>
<td>D</td>
<td>Supplemental Material for the OPASS Algorithm for Spike-Sorting</td>
<td>154</td>
</tr>
<tr>
<td>D.1</td>
<td>Supplemental Materials</td>
<td>154</td>
</tr>
<tr>
<td>E</td>
<td>Supplemental Material on the Relationships between Sorted Spikes and Local Field Potentials</td>
<td>160</td>
</tr>
<tr>
<td>E.1</td>
<td>Supplemental VB Updates and Equations</td>
<td>160</td>
</tr>
</tbody>
</table>
E.2 INLA for the non-clustering dynamic model . . . . . . . . . . . . . . 161

Bibliography 166

Biography 177
## List of Tables

2.1 A DBN with 100 hidden nodes in the first layer and 200 hidden nodes in the second layer. The left column denotes the number of training samples on the first layer, and then the corresponding lower bounds on the log likelihood given for both the training and the testing set. 42

6.1 Mean held-out RFE of the multi-cell models predicting the Hippocampus LFP. “Invariant” denotes the time-invariant model, “Non-cluster” and “clustering” denote the dynamic model without and with clustering. 122

6.2 Mean held-out RFE of the animal going through sleep cycles in each region. 126

E.1 Hold-out RFE predicting the Nucleus Accumbens region with the novel environment dataset 163

E.2 Hold-out RFE predicting the Prelimbix Cortex region with the novel environment dataset 164

E.3 Hold-out RFE predicting the Thalamus region with the novel environment dataset 164

E.4 Hold-out RFE predicting the Ventral Tegmental Area region with the novel environment dataset 165
List of Figures

1.1 The different graph structures between the RBM (a) and the SBN (b). The RBM is defined as a joint probabilistic model over the hidden units and the visible units, and so is undirected. The SBN first defines a prior over the hidden units, and then conditionally generates the visible units. This gives a directed graph structure.

2.1 Results on simulated $M = 100$, $J = 25$ datasets, all plots shown in the log scale on iterations (Left) $W$ set to a full matrix. (Middle) $W$ set to a matrix with rank 5. All algorithms converge to the same approximate objective value, but SSD gets to that level in fewer iterations. (Right) $W$ set to a full matrix, but the batch size is set to 10.

2.2 Learning curves for a single-layer RBM on MNIST showing the mean log $p(v)$ versus the number of iterations. (Left) $J=25$ (Middle) $J=100$ (Right) $J=500$. SSD shows improved performance over competing algorithms across all levels of $J$ for this dataset.

2.3 Train (solid line) and test (dashed line) model likelihood after 10,000 iterations for different values of $J$ with different optimization methods.

2.4 Per-unit empirical entropy for a $J = 500$ RBM training on the MNIST data set after 50,000 iterations. SSD shows higher entropies.

3.1 Learning curves for on the SBN model. The curves where the step-sizes are set to the rates in Section 3.3.1 are solids, and the tuned algorithms have dashed lines (Left) Training curves for toy data with $M=250$, $J=25$. (Middle, Right) Training curve for the MNIST data set with $M=784$, and $J=25$ or $J=100$, respectively. Untuned methods for all but SSD are not shown because they are uncompetitive.

3.2 Results on RSBN (Left and Middle) training reporting the perplexity metric. (Left) Synthetic data with $J = 15$ using exact gradients and log-likelihood measurements (Middle) $J = 50$, using variational approximation to obtain gradients and log-likelihood. (Right) Results on RS-RBM for a synthetic dataset of size $J = 25$. 
3.3 (Left) Learning curves for 20 Newsgroups with the RS-RBM. (Middle) Learning curves for 20 newsgroups with the RSBN (Right) Learning curves for Reuters with the RSBN.

4.1 Accuracy of the various methods on d533101 data (Henze et al., 2000). All abbreviations are explained in the main text (Section 4.3.1). Note that dictionary learning dominates performance over principal components. Moreover, modeling multiple channels (as in MDP and FMM) dominates performance over operating on each channel separately.

4.2 Clustering results shown in the 2 PC space of the various methods on d533101 data (Henze et al., 2000). All abbreviations are explained in the main text (Section 4.3.1). “Known neuron” denotes waveforms associated with the neuron from the cell with the intracellular recording, and “Unknown neuron” refers to all other detected waveforms. Note that all methods are shown in the first two PCs for visualization, but that the FMM-DL shown in (d) is jointly learning the feature space and clustering.

4.3 Longitudinal data analysis of the rat motor cortex data. (a) Schematic of the neural recording array that was placed in the rat motor cortex. The red numbers identify the sensors, and a zoom-in of the bottom-eight sensors is shown. The sensors are ordered by the order of the read-out pads, at left. The presented data are for sensors numbered 1 to 8, corresponding to the zoomed-in region. (b) From the maximum-likelihood collection sample, the apportionment of data among mixture components (clusters). Results are shown for 45 sec recording periods, on each of 8 days. For example, D-4 reflects data on day 4. Note that while the truncation level is such that there are 20 candidate clusters (vertical axis in (b)), only an inferred subset of clusters are actually used on any given day. (c) Predictive likelihood of held-out data. The horizontal axis represents the fraction of data held out during training. FMM-DL dominates NFMM-DL on these data.

4.4 Posteriors and dictionaries from rat motor cortex data (the same data as in Figure 4.3). (a) Approximate posterior distribution on the number of global clusters (mixture components). (b) Approximate posterior distribution of the number of dictionary elements. (c) Examples of inferred dictionary elements; amplitudes of dictionary elements are unit less.
4.5 Example clusters inferred for data on the bottom 8 channels of Fig.
4.3(a). (a)-(b) Example of single-unit events. (c) Example of a cluster
not attributed to a single-unit-event. The 8 signals are ordered from
left to right consistent with the numbering of the 8 channels at the
bottom of Figure 4.3(a). The black curves represent the mean, and
the error bars are one standard deviation. 82

4.6 Our generative model elegantly addresses missing data. (a) Example
of a clipped waveform from the publicly available data (blue), original
waveform (gray) and recovery waveform (black); the error bars reflect
one standard deviation from the posterior distribution on the under-
lying signal. (b) Relative errors (with respect to the mean estimated
signal). Note that we only show part of the waveform for visualization
purposes. 85

4.7 Effect of manually tuning \( \omega_0 \) to obtain a different number of features
for the rat motor cortex data. (a) Waveforms projected down onto two
learned features based on cluster result with \( \omega_0 = 10^6 \), the number of
inferred clusters is two. (b) Same as (a) with \( \omega_0 = 10^8 \); the number of
inferred clusters is seven. 87

4.8 Sparse firing results on synthetic data based on the Pittsburgh dataset.
The three rows correspond to three different signal-to noise ratio
(SNR) levels: (a) 1, (b) 1.5, and (c) 2.5. The four columns correspond
to: (1) cluster results of spike waveforms with colors representing dif-
ferent clusters, (2) plots of learned features based on cluster result, (3)
approximate posterior distribution of cluster numbers, and (4) confu-
sion matrix heatmap. Note that we accurately recover all the sparsely
spiking neurons except the sparsest one in the noisiest regime. 88

4.9 Performance analysis in the sparsely firing neuron case on synthetic
data based on the Pittsburgh dataset. (a) Accuracy comparisons
based on the cluster results under the various SNR. (b) Approximate
posterior distributions of error rate for FMM-DL in the different SNR
levels. (c) Approximate posterior distributions of spike waveform num-
ber for the unit 2, unit 3, and unit 4 under the various SNR regimes.
89

5.1 Opass achieves improved sensitivity and specificity over all competing
methods on partial ground truth data. (a) True positive and false
positive rates for all variants of Opass and several competing algo-

5.1 Opass achieves improved sensitivity and specificity over all competing
methods on partial ground truth data. (a) True positive and false
positive rates for all variants of Opass and several competing algo-

5.2 Opass detects multiple overlapping waveforms (Top Left) The observed voltage (solid black), MAP waveform 1 (red), MAP waveform 2 (blue), and waveform from the sum (dashed-black). (Bottom Left) Residuals from same example snippet, showing a clear improvement in residuals. ................................................................. 111

5.3 The IC waveform changes over time, which our posterior parameters track. (a) Mean IC waveforms over time. Each colored line represents the mean of the waveform averaged over 24 seconds with color denoting the time interval. This neuron decreases in amplitude over the period of the recording. (b) The same waveforms plotted in PC space still captures the temporal variance. (c) The mean and standard deviation of the waveforms at three time points for the auto-regressive prior on the mean waveform (top) and static prior (bottom). While the auto-regressive prior admits adaptation to the time-varying mean, the posterior of the static prior simply increases its variance. ............................. 112

5.4 Improving Opass by incorporating multiple channels. The top 2 most prevalent waveforms from the NeuroNexus dataset with three channels. Note that the left panel has a waveform that appears on both channel 2 and channel 3, whereas the waveform in the right panel only appears in channel 3. If only channel 3 was used, it would be difficult to separate these waveform. ................................................................. 112

6.1 (Left) Mean single-cell holdout RFE predicting mouse 3’s Nucleus Accumbens LFP comparing the dynamic and time-invariant model. Each point is a single neuron. (Middle) Convolutional dictionary for a VTA cell predicting mouse 3’s Nucleus Accumbens LFP at 5 minutes, 15 minutes, and 38 minutes after the experiment start. (Right) Hold-out RFE over experiment time with the time-invariant, non-clustering, and the clustering model to predict mouse 3’s Hippocampus LFP. ................................................................. 123

6.2 Example clusters predicting mouse 3’s Hippocampus LFP. The top part shows the convolutional factor throughout the duration of the experiment, and the bottom part shows the location of the cells in the cluster. Some of the clusters are dynamic whereas others were consistent through the duration of the experiment. ................................. 123

6.3 (Left) RFE as a function of time bin and frequency bin for all Hippocampus cells predicting the Thalamus LFP. There is a change in the predictive properties around 10 minutes. (Middle) Total energy versus the unexplained residual for the Hippocampus cells predicting the Thalamus LFP for the frequency band 25-35 Hz. (Right) RFE using only the cluster of cells shown in Figure 6.2(right). ................................................................. 124
6.4 The predictive patterns of individual neurons predicting multiple regions. (Left) A Hippocampus cell is the best single cell predictor of the V1 LFP. (Middle) A V1 cell with a relationship only to the V1 LFP. (Right) A Nucleus Accumbens Shell cell that is equivalent in predictive ability to the best V1 cell.

6.5 (Left) The cluster predicting the V1 region of the brain, matching known pattern for individual V1 cells Rasch et al. (2009); Nauhaus et al. (2009). (Middle, Right) Clusters predicting the motor cortex that show positive (pro) and negative (anti) relationships between amplitude and sleep.

6.6 Mean RFE when the animal is awake and when it is asleep. (Left) Cluster’s convolution factor is stable, and shows only minor differences between sleep and awake prediction. (Middle and Right) Clusters shown in Figure 6.5 (left and right), depicting varying patterns with the mouse’s sleep state.

D.1 (a) Dictionary learned from the first 5 seconds of data from the HC1 dataset. (b) Percentage of variance explained by each PCA component.

D.2 (a) This shows the average number of true positives versus the average number of false positives in the intracellular cluster for 2 minute segments of the 4 minutes of the experiment. OPASS does better than all the competitors.

D.3 Improving OPASS by incorporating multiple channels. (a) Three electrode device showing local proximity of electrodes with channel indexes in large, red numbers. (b) The representation of detected spikes on the 3rd channel in PCA space. This cluster does not seem separable here.

D.4 False and true positive detections have the same first-order statistics, making detection using only these statistics quite difficult. (a) Error-bar plots of the true positives, false positives, and missed positives in the IC cluster. While the false positives have slightly more variability, the mean shape for the false positives and the true positives is nearly identical. The true misses have a significantly lower amplitude as well as high variability. (b) All waveforms from the IC neuron as well as those we estimated from the IC neuron projected onto the first two PC space.

D.5 Pairs plot of true positives (black), false positives (blue x’s), and missed positives (red +’s). It does not seem like clustering in this space could yield much improvement.
D.6 $\mathcal{O}_{\text{pass}}$ multielectrode performance. (a) 8 electrode device showing local proximity of electrodes with channel indexes in large, red numbers. (b,c,d) Top three most prevalent waveforms. Each waveform shape is 2 ms long.

D.7 $\mathcal{O}_{\text{pass}}$ scales linearly with amount of data, with a slope smaller than one, meaning that $\mathcal{O}_{\text{pass}}$ can operate in real-time.

E.1 This is the posterior found from a single cell in INLA.
List of Abbreviations and Symbols

General Symbols
I have attempted to maintain consistent notation throughout this document. First, there are general representations of vectors, matrices, and math notation:

\begin{itemize}
  \item $n$, $N$ Unbolded symbols represent scalars.
  \item $\mathbf{x}$ A bold lowercase letter represents a column-wise vector.
  \item $\mathbf{X}$ A bold uppercase letter represents a matrix.
  \item $\mathbf{X}_{m,:}$ Represents the $m^{th}$ row of $\mathbf{X}$.
  \item $\mathbf{X}_{:,n}$ Represents the $n^{th}$ column of $\mathbf{X}$.
  \item $\odot$ Represents an element-wise (Hadamard) product.
  \item $\otimes$ Represents the Kronecker product.
  \item $\mathbf{x} \ast \mathbf{y}$ Represents the convolution of signals $\mathbf{x}$ and $\mathbf{y}$
  \item $\langle \cdot , \cdot \rangle$ Represents an inner product.
  \item $\mathbb{R}$ The real domain.
  \item $\mathbb{Z}$ The domain of integers.
\end{itemize}

Norms and Functions
Furthermore, we utilize definitions of norms and functions that are commonly used in this thesis. These definitions are largely standard, but are included here for completeness:

\begin{itemize}
  \item $||\mathbf{x}||_p$ Represents the $\ell_p$ norm of a vector $\mathbf{x}$
  \item $||\mathbf{X}||_F$ Represents the Frobenius norm of a matrix $\mathbf{X}$
\end{itemize}
\[ \|X\|_{Sp} \] Represents the Shatten-p norm a matrix, which is \( \|\lambda\|_p \) on the singular values \( \lambda \) of \( X \).

\[ \sigma(\cdot) \] Represents the sigmoid function \( \sigma(x) = \frac{1}{1+\exp(-x)} \).

\[ \text{softmax}(\cdot) \] Represents the softmax function, \( [\text{softmax}(x)]_n = \frac{\exp(x_n)}{\sum_{i=1}^{N} \exp(x_i)} \).

Lipschitz properties

The gradient of a function \( f \) is Lipschitz continuous with parameter \( L > 0 \) if
\[ \|\nabla f(x) - \nabla f(y)\|_p \leq L \|x - y\|_q, \]
where \( q \) is the dual norm to \( p \). Functions that are Lipschitz gradient on norm \( p \) have an upper bound
\[ f(y) \leq f(x) + \langle \nabla f(x), (y - x) \rangle + \frac{L}{2} \|x - y\|_q^2. \]
When this is the \( \ell_2 \) norm, minimizing the upper surrogate leads to gradient descent, which the SGD mirrors. In this work, we focus only on upper surrogate function, and use the relationship of the dual norm to derive parameter updates. Bounds written as only a subset of the parameters consider all unwritten parameters to be held constant.
Acknowledgements

I would like to thank everyone who has positively impacted my education, which is quite a long list. I want to thank my fellow group members who either through discussions or collaborations have helped me in my graduate career. Amongst many others, this includes Minhua Chen, Eric Wang, Jorge Silva, Esther Salazar, Xianxing Zhang, Bo Chen, Mingyuan Zhou, Liming Wang, David Lue, Kyle Ulrich, Zhe Gan, Ricardo Henao, Lingbo Li, Piyush Rai, and Yan Kaganovsky. As well, I am grateful to my committee and collaborators, including David Dunson, Kafui Dzirasa, and Volkan Cevher, as well as Liam Paninski for hosting me as a visiting scholar at Columbia. I’d especially like to thank my advisor Lawrence Carin, who allowed me the freedom to explore a wide variety of topics.

Furthermore, I’d like to acknowledge the support of my family and friends, especially my parents for supporting me. As well, I’d like to thank Laura Macaulay for her support and encouragement.

I’d also like to thank the many institutions and universities have hosted me during my graduate career, including Columbia University, the University of Michigan, the University of Pittsburgh, Carnegie Mellon University, Huntington Medical Research Institute, and the Mathematical Sciences Research Institute, as well as the many organizations that have provided funding, including ARO, DARPA, DOE, NGA and ONR.
1

Introduction

1.1 Overview

This thesis presents novel models with applications to neural time-series, as well as presenting novel methods for parameter learning. This introduction contains background information on problems explored in this thesis, including models and learning methods, and Chapters 2-6 detail my contributions. The appendix contains details on specific model implementation, as well as proofs of Theorems and Lemmas contained in the main manuscript.

First, I introduce novel methods for stochastic learning in classes of probabilistic models, which include widely used models such as the Restricted Boltzmann Machine (Hinton, 2002) and the Sigmoid Belief Net (Neal, 1992). These models are commonly used as building blocks for deep learning models. Deep learning attempts to provide general purpose algorithms and models for data by using multiple non-linear transforms of the data, and has shown state-of-the-art performance in many different domains (Bengio, 2012). The basic ideas of these models and standard inference schemes are introduced in Section 1.2. Typically, these models are learned using stochastic gradient descent, and background details are given in Section 1.2.3. In
Chapter 2 (Carlson et al., 2015b), a new bound on the log-sum-exp function is introduced that bounds the difference between two log-sum-exp functions on the $\ell_\infty$ norm, and this form is applied to the Restricted Boltzmann Machine objective function on the Shatten-$\infty$ norm. In this work, I propose a stochastic algorithm to optimize the objective function that operates in the Shatten-$\infty$ norm, and inference speed is dramatically improved. This relates to the concept of geometry in optimization problems, and better matching of norms via Steepest Descent gives improved optimization in many problems (Nesterov, 2012). In Chapter 3 (Carlson et al., 2015a), this idea is extended to other discrete graphical models, and general theorems are introduced to create bounds that are useful for bounding objective functions in both MAP estimation and Variational methods.

Next, I discuss generative models for processing neural electrophysiological data in Chapters 4, 5, and 6. This work focuses first on spike sorting; spike sorting methods are a class of methods that attempt to extract spike trains of individual neurons from a voltage time-series signal from an extracellular voltage signal (Lewicki, 1998a). There has been a great amount of recent work into this problem, including Bayesian nonparametric clustering methods (Wood et al., 2006). I give a background about commonly used models in Section 1.3. For my contributions, I demonstrate the benefits of using a joint clustering and feature learning approach in Chapter 4 (Carlson et al., 2013b), as well as demonstrating an online algorithm for deconvolutional processes with a Bayesian nonparametric process to infer the number of active neurons in Chapter 5 (Carlson et al., 2013a). Additionally, in Chapter 6, I discuss using inferred spike trains to reveal features about the relationships of the neurons and local field potentials (Carlson et al., 2014).

During my doctoral work, I have also explored other topics, including the use of information theory for compressive sensing design (Wang et al., 2013, 2014), model averaging methods (Hu and Carlson, 2014), statistical methods for mRNA data
(Chen et al., 2011c), and Gaussian process methods for local fields potentials (Ulrich et al., 2014). This document makes note of this work, but focuses on contributions to stochastic methods and electrophysiological models. Furthermore, it should be noted that (Hopke, 1950; Carlson, 1969) have been helpful.

1.2 Graphical Models for Deep Learning and Stochastic Inference

The popularity of deep learning is increasing yearly, as it becomes state-of-the-art in many different learning tasks (Bengio, 2012). Building these deep models is a computationally demanding task, often undertaken by large and specialized computer systems. One method for training these models is to use a “pre-training” technique, where 2-layer building blocks are repeatedly learned to develop a many-layer network. Many different graphical models are used as building blocks for deep learning, including both undirected and directed graphical models. A simple undirected graphical model is the Restricted Boltzmann Machine (Hinton, 2002), and background details on this model are included in Section 1.2.1. Alternatively, a directed graphical model could be used, such as the Sigmoid Belief Network (Neal, 1992) that is introduced in Section 1.2.2.

1.2.1 Restricted Boltzmann Machines

Undirected graphical models, which are often referred to as Markov Random Fields, are used in many different applications. As an initial example, we consider the Restricted Boltzmann Machine (RBM) (Hinton, 2002). The RBM is a bipartite graph between layers where each node is binary \( \{0, 1\} \), a special case of the Ising model in physics. This graph structure can been seen in Figure 1.1(a). Instead of modeling correlations between observations \( \mathbf{v}_n \in \{0, 1\}^M \) for \( n = 1, \ldots, N \) directly, the RBM links observed nodes through hidden unobserved nodes \( \mathbf{h}_n \in \{0, 1\}^J \). The
model is defined by choosing an energy function, which is

\[-E(v, h) = v^T c + v^T W h + h^T b\]  \hspace{1cm} (1.1)

However, the parameters \(\theta = \{c, W, b\}\) are not known a priori, and must be learned. Typically, the problem is set up as a maximum a posteriori (MAP) problem, with the model likelihood defined as:

\[
\frac{1}{N} \log p_{\theta}(v_1, \ldots, v_N) = \frac{1}{N} \sum_n \log \sum_h \exp(-E_{\theta}(v_n, h)) - \log Z(\theta) \tag{1.2}
\]

\[
\log Z(\theta) = \log \sum_v \sum_h \exp(-E_{\theta}(v, h)) \tag{1.3}
\]

This gives a MAP estimation problem of

\[
\theta = \arg \max_{\theta} \log p_{\theta}(v_1, \ldots, v_N) + g(\theta) \tag{1.4}
\]

where \(g(\theta)\) comes from the prior on the parameters, which is usually defined as Gaussian or Laplacian in practice, giving an \(\ell_2\) or \(\ell_1\) penalized maximum likelihood problem. The gradient on the model likelihood is easily written,

\[
\nabla_{\theta} \frac{1}{N} \log p_{\theta}(v_1, \ldots, v_N) = \frac{1}{N} \mathbb{E}_{p(h|v_n)}[\nabla_{\theta}(-E(v_n, h))] - \mathbb{E}_{p(v,h)}[\nabla_{\theta}(-E(v, h))] \tag{1.5}
\]

Unfortunately, for all but the smallest problems this gradient is not tractable to calculate exactly because the expectation over the model cannot be calculated analytically. Instead, an approximate Monte Carlo Integration scheme is used, which is inspired by the standard Monte Carlo Integration scheme,

\[
\mathbb{E}_{p(v,h)}[\nabla_{\theta}(-E(v, h))] \approx \frac{1}{K} \sum_{k=1}^{K} \nabla_{\theta}(-E(v_k, h_k), \{v_k, h_k\} \sim p(v, h) \tag{1.6}
\]

Unfortunately, in the RBM it is not known how to efficiently sample from the generative model \(p(v, h)\). Instead, the generation scheme is approximated by Contrastive
Divergence (CD) (Hinton, 2002), which uses Gibbs sampling in a Markov chain to approximate true samples. A chain starts at a randomly sampled data point \( v_0 = v_n \), and then:

\[
v_0 \rightarrow h_0 \sim p(h|v_0) \rightarrow v_1 \sim p(v|h_0) \rightarrow h_1 \sim p(h|v_1) \rightarrow v_2 \sim p(v|h_1) \rightarrow \ldots \quad (1.7)
\]

As the CD order \( k \) (number of Gibbs samples) increases, \( v_k \) approaches the true probability distribution \( p(v) \). Additionally, further improvements have been made to Contrastive Divergence in recent years, including Persistent Contrastive Divergence (Tieleman and Hinton, 2009) and the Enhanced Gradient Method (Cho et al., 2013). However, the contrastive divergence process is both noisy and computationally expensive.

After the gradient has been estimated, this quantity is used in a stochastic gradient descent, which is introduced in section 1.2.3.

1.2.2 Sigmoid Belief Networks

Similar to the RBM, Sigmoid Belief Networks (SBN) (Neal, 1992) are connected graphs between hidden units \( h_n \in \{0, 1\}^J \) and visible units \( v_n \in \{0, 1\}^J \), and is often described with the same parameters as the RBM, with \( \theta = \{c, b, W\} \). However, the SBN first has a generative model on the prior, with

\[
p(h_j) = \text{Bernoulli}(h_j; \sigma(b_j))
\]

(1.8)

Then, the observations are dependent on the hidden units,

\[
p(v_m|h) = \text{Bernoulli}(v_m; \sigma([c + Wh]_m))
\]

(1.9)

Here, the RBM is described only as a joint generative model over \( \{v, h\} \), where the SBN is described as a generative process for \( h \) first, and then \( v \) is drawn after \( h \) is sampled. This is represented as a directed graphical model, which is shown in Figure
1.1(b). Because this has a different method of generation, the SBN has a different energy function,

\[-E(v, h) = v^T c + v^T Wh + h^T b - \sum_m \log(1 + \exp([c + Wh]_m)) - \sum_j \log(1 + \exp(b_j))\]  

(1.10)

Although this energy function has additional nonlinear terms compared to the RBM, it leads to a nice property of the partition function that

\[Z(\theta) = \sum_{v, h} \exp(-E(v, h)) = 1\]  

(1.11)

Likewise to the RBM, the primary mode of model learning is via gradient methods, although there have been methods that use variational methods and auxiliary variable samplers (Mnih and Gregor, 2014; Gan et al., 2015). Like the RBM, the gradient is easy to write, with

\[\nabla_\theta \log p(v_n) = E_{p(h|v_n)}[\nabla_\theta (-E(v_n, h))]\]  

(1.12)

For the penalized maximum likelihood or MAP estimate, this quantity is estimated by using Gibbs sampling on \(h|v\) to get approximate posterior samples, and then Monte Carlo Integration. Traditionally, stochastic gradient descent has been used for parameter learning, which is described in Section 1.2.3.

1.2.3 Stochastic Gradient Descent for RBM and SBN

Stochastic gradient descent (SGD) is very popular in modern machine learning methods because it can be computationally more efficient for practical problems. As demonstrated in (Bousquet and Bottou, 2008), the objective function only needs to be calculated within statistical error. With large datasets, the minimum computational time methods can often be SGD instead of classical methods such as gradient descent or Newton’s method, which have a better per-iteration rate of convergence compared to SGD.
The idea of SGD is that the cost function is separable into $N$ distinct data points,

$$f(\theta) = \frac{1}{N} \sum_{n=1}^{N} f_n(\theta)$$  \hspace{1cm} (1.13)

then an optimization procedure proceeds by choosing a data point $i$ at random, and updating

$$\theta^k = \theta^{k-1} - s_k \nabla f_i(\theta^{k-1})$$  \hspace{1cm} (1.14)

where $s_k$ is the step size at iteration $k$. Denoting $\theta^*$ as the optimal solution, we can examine the convergence of the series $\theta_k \to \theta_{k+1} \to \ldots \theta^*$ in a convex function. The error decreases stochastically with rate $E[f(\theta^k) - f(\theta^*)] = \mathcal{O}(\frac{1}{k})$ for step sizes $s_k = \frac{s_0}{k}$ and $s_0 \leq 2L$, where $L$ is the Lipschitz constant of the gradient with respect to the $\ell_2$ norm.

Unlike batch methods, such as gradient descent or Newton’s method, stochastic gradient descent does not adapt to strong convexity to improve the convergence rate. However, in many applications a single step of SGD can be $N$ times faster than a single step of GD, and optimization only needs to get the parameters within the statistical error. For modern huge data sets, SGD will reach this level orders of magnitude faster than deterministic methods, although the per-iteration rate is worse.

For the SBN and the RBM, the Lipschitz constants (or similar quantities) for the parameters were not previously known. For the RBM, the Lipschitz constant on $W$ is $\frac{1}{MJ}$ with respect to the Frobenius norm, and for the SBN, the Lipschitz constant is $\frac{4}{J}$ with respect to the Frobenius norm. Curiously, both of these models have tighter bounds with respect to the Shatter-$\infty$ norm, and stochastic schemes based off of these alternative geometries is presented in Chapters 2 and 3.
1.3 Spike Sorting

“Spike sorting” refers to the process of extracting neural spike trains from raw electrophysiological voltage recordings, and has applications ranging from brain-machine interfaces to neural coding and beyond. Given a time-series of voltage data from implanted electrodes, we would like to be able to accurately determine:

- The timestamp of all action potentials
- The number of neurons and how the action potentials are sorted into their corresponding neurons
- The dynamic firing rate of each neuron

These goals are not independent tasks, nor is this an exhaustive set of goals. Further applications of spike sorting can be found in (Gibson et al., 2008). Detailed analysis of the interaction between tuning curves and the bias of spike sorting can be found in (Ventura, 2009b; Ventura and Gerkin, 2012). This document concerns itself with primarily the first two items given, which are referred to as “Spike Detection,” extracting the action potentials from the background time series, and the “clustering
step,” which deals with separating the detected action potentials into different clusters (ideally, each cluster would accurately represent a different neuron). The results from these two steps are used in many downstream processing applications. Lewicki (Lewicki, 1998a) gives a straightforward review that splits the spike sorting problem into 4 distinct steps:

1. Measuring neural activity and filtering the raw signal
2. Spike detection
3. Feature extraction
4. Cluster analysis

Each of these separate steps has been an area of dedicated research, and a brief overview of steps 2-4 follows.

1.3.1 Spike Detection

Goals, evaluation and trade-offs

The general goal of spike detection algorithms is to detect a neural action potentials (true positives) and not to detect background signals as action potentials (false positives). The issue of false positives is complicated by the nature of the background signal, which is made up of both thermal noise and further-away neural activity (Buzsaki, 2004). As in most detection methods, for any given detection method we have a threshold that is tunable to tradeoff between the true positive rate and the false positive rate. In addition, for offline analysis computation complexity isn’t usually a large concern, but in applications such as BMI (Obeid and Wolf, 2004; Gibson et al., 2008) algorithmic computational efficiency is a concern as well.

Typical methods to evaluate the detection algorithms include using fully synthetic datasets (Mukhopadhyay and Ray, 1998; Obeid and Wolf, 2004; Gibson et al., 2008)
or to use data with partial ground truth and compare the algorithms on the known detections (Shahid et al., 2010; Carlson et al., 2013a). The synthetic datasets are made up of realistic background noise (often taken from another recording or a superposition of white noise and small-amplitude waveforms) and scaled versions of well-isolated waveforms, where the scaling is chosen to mimic varying SNR.

The evaluation metrics are as follows. First, at each time step (or window), we will either call the data a “spike” or “noise.” If we call the data a spike, and it is in fact a spike, this is a true positive. If we call it a spike when it is just background signal, then we call this a false positive. Likewise, if we label something noise and it was a spike, this is a “missed positive.” If we label a section noise when it is noise, this is a “correct rejection.”

To evaluate the performance of a detection algorithm, the most often metric is the true positive-false positive (TP-FP) rate, which is going to be dependent on the threshold of the detection algorithm. A higher threshold will give a lower true positive rate, but also a lower false positive rate. Because of this tradeoff based on the given threshold for a detection algorithm, the people often compare the full (TP-FP) curve, which is called the ROC (Receiver Operator Characteristics) curve. In many applications the AUC (Area Under the Curve) metric is used, but due to scaling problems the AUC metric is not used to evaluate detection performance.

If we are using a Bayesian probabilistic model as in (Carlson et al., 2013a), it is straightforward to calculate the posterior probability of a spike on a voltage snippet \(v_t\) and a model \(\mathcal{M}\), which we can write:

\[
p(\text{spike}|v_t, \mathcal{M}) = \frac{p(v_t|\text{spike}, \mathcal{M})p(\text{spike}|\mathcal{M})}{p(v_t|\text{spike}, \mathcal{M})p(\text{spike}|\mathcal{M}) + p(v_t|\text{background}, \mathcal{M})(1 - p(\text{spike}|\mathcal{M}))}
\]

(1.15)

We can use this equation determine the probability that the waveform snippet is a spike (assuming the model is correct, no small assumption) and use a tunable
threshold to determine whether the model will call the snippet an action potential or background signal. In practice, many of the simple detection methods don’t use the Bayesian optimal method or the likelihood ratio, instead opting for when the voltage sample is significantly unlikely under the model for the background noise, usually for convenience and computational efficiency.

*Simple threshold detection*

One of the simplest and most widespread methods of detection is to use a simple threshold detection (Lewicki, 1998a). In this method, a threshold is set at a level $\tau$, and whenever the voltage trace passes the threshold level, a spike is detected. The researcher has to determine the appropriate level $\tau$ for the voltage threshold, which we will briefly discuss. Additionally, typically after the detection there is a period before another spike can be detected (Obeid and Wolf, 2004), which is to prevent spurious detections of the same waveform twice. Note that this practice can become an issue when spike density is high and there are a significant number of overlapping spikes, and (Obeid and Wolf, 2004) advocates for this method based on their experiments.

Succinctly, if we let our voltage time series be represented by $x(t)$ and the lag between spikes is $L$, then we can find time of the $n^{th}$ spike $s_n$ by:

$$s_n = \arg \min_{t > s_{n-1} + L} x(t) > \tau$$

Depending on the situation, it may be beneficial to use the absolute value of negative of the voltage signal.

In practice, $\tau$ is often set to 3 or 4 standard deviations of the *noise signal* away from the mean. Estimating the standard deviation of the noise can be done in a variety of ways.
Nonlinear Energy Operator

The Nonlinear Energy Operator (NEO) traditionally gives an estimate of the energy content of a linear oscillator, which was extended to spike detection by (Mukhopadhyay and Ray, 1998). The general form of the operator a continuous signal $x(t)$ is

$$\psi_{NEO}[x(t)] = x^2(t) - x(t)x''(t),$$

where $x''(t)$ is the second derivative of the signal. The operator on discrete signals is given by:

$$\psi_{NEO}[x[p]] = x^2[p] - x[p]x''[p].$$

To decide whether an event is a spike or not, we now use the output of the NEO operator with a threshold. Once again, an appropriate value of $\tau$ must be given. In the original paper (Mukhopadhyay and Ray, 1998), they advocated for setting the threshold by scaling the mean of the NEO operator:

$$\tau = C \frac{1}{N} \sum_{t=1}^{N} \psi_{NEO}[x(t)]$$

Where $C$ is determined experimentally in pilot studies and then set to a constant for later experiments. $C$ was set to 1.75 in (Mukhopadhyay and Ray, 1998).

Additionally, because of the nature of the NEO operator, it is not robust to noise in the high frequency regime. One idea to improve it’s robustness is to smooth with estimator with a window to get the Smoothed Nonlinear Energy Operator (SNEO). Letting $*$ denote convolution and $w(t)$ be a smoothing window function:

$$\psi_{SNEO}[x(t)] = \psi_{NEO}[x(t)] * w(t)$$

Overall, the NEO and SNEO get mixed performance over datasets versus the simple threshold (Obeid and Wolf, 2004; Gibson et al., 2008; Shahid et al., 2010), but is energy efficient and performs well in some situations.
Other methods

Other commonly used methods included wavelet-based detection methods and matched filter/template matching methods. In theory, a matched filter is the best detection algorithm you can use, but it is computationally more demanding and the spikes are often of an unknown shape, making the user guess at the waveform shape and the algorithm becomes suboptimal.

Detection for BMI

The ideal method of spike detection also depends on the application, where offline analysis can do a more computationally intensive detection scheme whereas in a real-time Brain-Machine interface applications computationally efficient schemes are highly desirable. Both (Obeid and Wolf, 2004) and (Gibson et al., 2008) provide a review of largely the same spike sorting methods for the BMI problem. (Obeid and Wolf, 2004) recommends sending the raw signal through an absolute value and then using a simple threshold whereas (Gibson et al., 2008) advocates the use of the Nonlinear Energy Operator (NEO).

1.3.2 Feature extraction

As a preprocessing step to the spike-sorting algorithm, some method of feature extraction will be applied to the extracted snippets \( \{x_1, \ldots, x_n\} \) to get a vector of features \( y_i \) for each snippet.

Principal component analysis (PCA)

By far, the most commonly used method of feature extraction is PCA, which reduces the dimensionality of the problem from \( L \) dimensions to \( p \) dimensions, where \( p < L \), and \( p \) is set by the user. It’s common practice to use a low number of principal
components, usually between 2-5. The common Plexon Offline Sorter\(^1\) defaults to using the top 3 principal components.

*Probabilistic principal component analysis (PPCA)*

In many situations the spike detection is performed online and the original time-series is not stored for either space or energy constraints. If the detection algorithm works as planned, then PCA can be used well to estimate the subspace. However, often the detection algorithm can result in badly misaligned spikes, and realigning the spikes will cause a missing data problem. In this situation we would still like to be able to reduce the dimensionality of the problem, but the PCA algorithm cannot be applied in this case due to the missing. One method of attempting to solve this is the PPCA algorithm (Tipping and Bishop, 1999), or any of the other Bayesian dictionary learning techniques, or any of the robust matrix factorization methods, for that matter.

The PPCA algorithm attempts to solve:

\[
y_i \sim \mathcal{N}(\mu, WW^T + \sigma^2 I)
\]  

(1.20)

This is possible to solve this in many different ways, such as the EM algorithm, will allow you to learn a feature space with missing data. Typically, it will closely align with the PCA result if the data was completely observed.

Additionally, this feature extraction (with or without missing data) can be combined with the spike sorting algorithm, as in (Carlson et al., 2013b).

1.3.3 **Spike Sorting**

Assuming that we already have the waveform voltage traces from the detection algorithm, we now have a collection of \(N\) vectors \(p\)-dimensional waveforms. Given this collection, we want to perform unsupervised clustering to:

\(^1\) http://www.plexon.com/products/offline-sorter
• Learn the number of active neurons

• Sort each waveform into appropriate clusters (or label it as noise), so that each cluster corresponds to an active neuron

Additionally, it may be of interest to:

• Track waveform dynamics over time

• Track varying neural firing rates over time

• Correlate neural activity with stimuli

In order to do this, we have to come up with effective unsupervised algorithms. One of the biggest problems is model evaluation. In particular, how do we know how many neurons (how many clusters) there are? Additionally, if we know the number of clusters, what is the best way to split the data into appropriate clusters.

One approach to model evaluation is to use synthetic data. To create synthetic data, most people take a similar approach to (Quiroga et al., 2004). In this approach, the background signal is made up of a superposition of waveforms from previous experiments at random amplitudes and times, as to mimic real neural background activity. Then several spike trains are added using waveforms estimated from previous experiments that match specific SNR profiles. These synthetic datasets allow full comparisons of both detection algorithms as well as the accurate clustering, both in terms of the number of active neurons and the data splits.

Alternatively, there are datasets where partial ground truth is known, such as the case when an extracellular probe and an intracellular probe are recorded from jointly. The extracellular probe is set up as normal for spike sorting, whereas the intracellular probe places the electrode inside an actual neuron (this is hard to do and I don’t know of any lengthy datasets). The intracellular recording gives a very clean
estimate of spike times, so that we expect to see a corresponding spike in the noisy extracellular dataset at the same time. Details of experiments like these can be seen in (Harris et al., 2000), which implanted an extracellular tetrode and an intracellular probe in the hippocampus of anesthetized rats. The data from that experiment is publicly available, and can be found at http://crcns.org/data-sets/hc/hc-1.

Generally, we model the background noise as some sort of Gaussian distribution, or ignore it after feature selection. If we’re dealing with snippets, we assume that each snippet comes from a single cluster and there are no overlapping spikes (i.e. a snippet with waveforms from 2 or more clusters). We can write a parametric mixture model by writing the a sum over \( K \) different distributions:

\[
y_i \sim \sum_{k=1}^{K} \pi_k f(y_i; \theta_k)
\]  
(1.21)

In spike sorting, we can specifically about which cluster each waveform belongs to, so we can write the auxiliary variables, and in this case we are going to use the common Gaussian Mixture Model for the data (Lewicki, 1998a):

\[
y_i | z_i \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})
\]  
(1.22)

\[
z_i \sim \text{categorical}(\pi_1, \ldots, \pi_K)
\]  
(1.23)

Where our parameters \( \Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1,\ldots,K} \) define the model. There are several different ways to optimize this model, such as the Expectation Maximization (EM) algorithm and Markov Chain Monte Carlo (MCMC) methods. Additionally, we there has been work looking into extending this model to incorporate a Bayesian nonparametric framework (Wood et al., 2006; Gasthaus et al., 2008; Wood and Black, 2008b; Chen et al., 2011b; Carlson et al., 2013b). The nonparametric Bayes framework provides one framework to infer the number of clusters (or neurons) in the dataset. Alternative methods have been used which choose the number of clusters via AIC (Ventura, 2009b) or BIC (Pouzat et al., 2002).
Action potentials are often modeled as voltages propagating over a lossy medium, i.e. brain tissue. Even though the signals are lossy, it seems like if electrode sites are close enough, then any action potential will appear on multiple channels. (Buzsaki, 2004) put a multi-electrode array in the brain tissue with electrode sites very nearby, and then detected spikes. After spike detection, you can use the signal over multiple channels to triangulate the distance. (Buzsaki, 2004) found that neurons within 50\(\mu m\) are strong enough to sort accurately, and spikes within 140\(\mu m\) can be detected, and this is pretty well accepted.

This type of triangularization only holds when you’re in a medium with homogeneous electromagnetic properties. The brain certainly does not satisfy this due to the non-homogeneity, and especially so when there is an implant in the brain. It’s close enough immediately after implant that these figures are pretty reasonable estimates, but during chronic implants the homogeneous medium breaks down. I’ve rerun these types of experiments on datasets with chronic implants (which are known to have very non-homogeneous properties, namely the tissue around the implant site is full of scar tissue and microglia and astrocytes), and triangulated neurons out to an extreme case of 330 \(\mu m\). That is an extreme figure, and is illustrative just that the assumptions break down, especially at longer distances and for chronic implants.

There are three main approaches to modeling waveforms over multiple channels. The first is just to concatenate all of the channels together before the feature extraction step. This can work very well when all waveform shapes appear on all channels simultaneously, such as in a tetrode, (Wood et al., 2006). An alternative is to use Independent Component Analysis type methods (Lewicki, 1998a). ICA works well when the waveform on each channel is a linear scaling of a base waveform shape, which holds in many cases but not in all.
Additionally, it is possible to use matrix methods, (Carlson et al., 2013b). Before, we had a simple vector of features for each waveform, now we have a vector of raw recordings for each channel $c = \{1, \ldots, C\}$, where $C$ is the total number of channels. We can describe the set of features at time $t$ for channel $c$ as $v_{tc}$ and write the set of features at time $t$ as a matrix $V_t = [v_{t1}, \ldots, v_{tC}]$. Just like when we are analyzing a single channel, we need to reduce the feature set.

The simplest thing to do is just to use SVD on the vectorized form of the matrix (Wood et al., 2006) (i.e. vec($V_t$) = [$v^T_{t1}, \ldots, v^T_{tC}$]). This works quite well spikes appear across all channels, such as in a tetrode. When we have a multi-electrode arrays that are more spread out, it doesn’t make sense (and can lead to over-fitting) to concatenate the channels. For example, if we have $C = 16$, then we would need at least 1 factor for channel (and more realistically 2+), causing us to estimate 30 factors, which is harder to do. Instead, we can attempt to find a lower-dimensional linear subspace (represented by $D$ with dimensions $N$ by $P$, where $N$ is the number of samples and $P$ is the number of features) that well explains all the channels (Carlson et al., 2013b), such that:

$$V_t = DY_t + E_t$$

This implies that $Y$ is a $P$ by $C$ matrix. Fitting this feature decomposition can be done through a variety of methods, most simply by just calculating the SVD on the collection of vectors \{${v_{11}, \ldots, v_{T1}, \ldots, v_{1C}, \ldots, v_{TC}}$\} and reforming the data into $D$ and $Y$. As a cautionary note, doing SVD in this way makes some big assumptions. Namely, the error is assumed to be at the same level across channels, which may or may not be a reasonable assumption. This can be simply handled by rescaling the channels so that the error variance is the same across channels.

After we have a matrix of features for each spike detection, $Y_t \in \mathbb{R}^{N \times C}$, we can run the EM algorithm on it. This can be set up in a few different ways. The way
with the least amount of structure is to say:

\[
\text{vec}(Y_t) \sim \sum_{k=1}^{K} \pi_k N(\mu_k, \Sigma_k)
\]  

(1.25)

Alternatively, we can set this problem up as a matrix normal, which will reduce the number of parameters needed to estimate:

\[
Y_t \sim \sum_{k=1}^{K} \pi_k MN(M_k, U_k, V_k)
\]  

(1.26)

Where \(MN\) represents the matrix normal where \(U_k \in \mathbb{R}^{N \times N}\) is the covariance matrix between rows, \(V_k \in \mathbb{R}^{C \times C}\) is the covariance matrix between columns, and \(M_k \in \mathbb{R}^{N \times C}\). The likelihood for a matrix normal is given by:

\[
p(Y|M, U, V) = \frac{\exp(-\frac{1}{2} \text{tr}[V^{-1}(Y - M)^T U^{-1}(Y - M)])}{(2\pi)^{np/2}|V|^{n/2}|U|^{p/2}}
\]  

(1.27)

Additionally, it is also easy to assume independence between channels given the cluster index, which we can write as:

\[
Y_t \sim \sum_{k=1}^{K} \pi_k \prod_{c=1}^{C} N(y_{tc}; \mu_{kc}, \Sigma_{kc})
\]  

(1.28)

It is straightforward to adapt any of these base distributions into a Bayesian non-parametric setup, or to use AIC or BIC on these models.

1.3.4 Combining spike detection and clustering

As the number of channels and the density of neurons increase, the number of cases where we have overlapping spikes increases. As discussed in many papers, overlapping spikes are difficult to work with when we only have waveform snippets and are a very difficult problem. There are ways of addressing this, but we have several recent
papers have attempted to jointly detect and cluster waveforms (Ekanadham et al., 2011; Pillow et al., 2013; Carlson et al., 2013a; Prentice et al., 2011) by processing the time-series directly. The time-series allows these methods to find overlapping spikes fairly well. The general model in these approaches is that the recorded time series is the superposition of each waveform and a Gaussian background signal. If we are recording from a single channel, then we can view the signal at time $t$ as the sum over all the active neurons. Letting $v(t)$ be the recorded signal at time $t$, $x_k(t)$ be the signal from (active) neuron $k$, and $\epsilon(t)$ be the background noise signal.

$$v(t) = \sum_{k=1}^{K} x_k(t) + \epsilon(t)$$

(1.29)

The model selection over $K$ is a nontrivial problem, and is approached in different ways. (Ekanadham et al., 2011; Pillow et al., 2013) choose $K$ by using a set of classically detected snippets to determine $K$, whereas (Carlson et al., 2013a) approaches the problem via a Poisson Process-Gamma Process to infer the number of clusters.

Since we are now dealing with the entire time series instead of just the detected snippets, the models are going to have to be modified. Because the amount of data that needs to be process has increased, there is an increased focus on computational efficiency. The noise is going to be modeled as (colored) Gaussian noise, which provides a computationally efficient and reasonable model for the noise ((Ekanadham et al., 2011) first whitens the signal and treats the signal as white Gaussian noise).

The time-series for each neuron $k$ is generated in two steps. First, the spike times \{s_1, \ldots, s_{N_k}\}, where $N_k$ is the number of events in the time-interval for neuron $k$. After the spike times are generated, a waveform is generated for each of the spike times and added to the signal.

Typically, for the generative process of the contribution from neuron $k$, we first draw the spike times from a Poisson process (a Poisson process is almost certainly
wrong due to refractory issues, but is computationally efficient) with rate \( \lambda_k \), so that the total number of spikes that we have for neuron \( k \) over a total interval \( T \) is \( n_k \sim \text{Poisson}(T\lambda_k) \) and corresponding spike times \( \{s_{k1}, \ldots, s_{k n_k}\} \). (We can make this a non-homogenous Poisson Process with a time-varying \( \lambda_k(t) \) where \( \lambda_k \) is now dependent on the current time.)

For each spike in neuron \( k \), \( i = 1, \ldots, n_k \), we draw a spike shape from some distribution over spike shapes \( d_{ki} \sim f_k \), where \( f_k \) is the distribution over spike shapes (assumedly of bounded length \( L \)), and then we can write the time series for neuron \( k \) as:

\[
x_k = \sum_{i=1}^{n_k} d_{ki} \star \delta_{sk_i}
\]

(1.30)

where \( \star \) denotes convolution.

Model fitting

Model fitting in these types of models can be difficult. One such approach is that of (Carlson et al., 2013a; Pillow et al., 2013) where greedy methods were used. Suppose that we have a dictionary \( \mathbf{A} \in \mathbb{R}^{L \times J} \) (which can be found in practice from PCA on a subset of spikes) and that for each spike time \( i \) in neuron \( k \) we draw the weights on these factors from a normal distribution:

\[
y_{ki} \sim \mathcal{N}(\mu_k, \Sigma_k)
\]

(1.31)

\[
x_k = \sum_{i=1}^{n_k} (\mathbf{A} y_{ki}) \star \delta_{sk_i}
\]

(1.32)

We additionally need to have a model on the noise distribution. Assume for now that we have a white noise distribution (which can be generalized later), so that \( \epsilon(t) \sim \mathcal{N}(0, \sigma^2) \).

Suppose for now that we only have a single neuron. If that is the case, we want to calculate the posterior probability of a set of spike times \( \{s_1, \ldots, s_n\} \), which we
can write as:

$$p(\{s_1, \ldots, s_n\}|v, \mu, \Sigma) = \frac{p(v|\{s_1, \ldots, s_n\}, \mu, \Sigma)p(\{s_1, \ldots, s_n\})}{p(v|\mu, \Sigma)}$$

$$\propto p(v|\{s_1, \ldots, s_n\}, \mu, \Sigma)p(\{s_1, \ldots, s_n\}) \quad (1.33)$$

This is intractable to calculate. One of the simplest and computationally feasible to get a local MAP (maximum a posteriori) estimate is to use greedy methods. The idea is to start with 0 spikes, and add a single spike at a time while those spikes will increase the posterior estimate.

One very common approximation that is helpful here is that we can model the Poisson process as a Bernoulli process. In this case, we define a vector of length $T$ full of binary variables, and $\{s_1, \ldots, s_n\} = \{t : z_t = 1\}$, and is given the pdf:

$$p(z) = \prod_{t=1}^{T} p(z_t) = (\lambda \Delta T)^{z_t} (1 - \lambda \Delta T)^{1-z_t} \quad (1.34)$$

where $\nu = \Delta T$ is defined as the width of each bin. If we increase the total number of spikes detected by 1, then:

$$\frac{p(\{s_1, \ldots, s_n, s_n + 1\})}{p(\{s_1, \ldots, s_n\})} = \frac{\nu}{1 - \nu} \quad (1.35)$$

If we want to increase our posterior distribution, we need to find somewhere where:

$$\frac{p(v|\{s_1, \ldots, s_n, s_n + 1\}, \mu, \Sigma)}{p(v|\{s_1, \ldots, s_n\}, \mu, \Sigma)} > \frac{1 - \nu}{\nu} \quad (1.36)$$

It makes sense to choose the best additional spike at each time. Because our spikes are localized, if we let $v_t^{t+L}$ be the voltage trace at time $t$ to $t + L$, where $L$ is the length of the spike train, if we assume no overlapping regions, then the change in the posterior value can be written:

$$\frac{\nu p(v_t^{t+L}|s_{n+1} = t, \mu, \Sigma)}{(1 - \nu)p(v_t^{t+L})} \quad (1.37)$$
\( p(v_t^{t+L}) \) comes from the noise model and is easy to calculate, whereas:

\[
p(v_t^{t+L}|z_t = 1, \mu, \Sigma) = \int p(v_t^{t+L}|z_t = 1, y_t)p(y_t|\mu, \Sigma)dy_t \quad (1.38)
\]

Either people evaluate this integral, or they will approximate it with the MAP value of \( y_t \). Note that using the MAP value often works well in practice, but leads to overfitting which is usually compensating by setting \( \nu \) much lower than its true value.

This integral can be simply evaluated by evaluating the normal distribution:

\[
p(v_t^{t+L}|z_t = 1, \mu, \Sigma) = \mathcal{N}(v_t^{t+L}; A\mu, \sigma^2 I + A\Sigma A^T) \quad (1.39)
\]

Note that this assumes a white noise model. It’s simple to transform to any Gaussian noise model with zero-mean and covariance \( V \):

\[
p(v_t^{t+L}|z_t = 1, \mu, \Sigma) = \mathcal{N}(v_t^{t+L}; A\mu, V + A\Sigma A^T) \quad (1.40)
\]

Often, however, we won’t know the true value of \( \{\mu, \Sigma\} \). We can continue integrating out the uncertainty in the model to get an estimate of the parameter with less dependence on the underlying parameters.

Now that we have a way of calculating the expected gain in the posterior, we have a well-principled way to add additional spikes in a greedy manner. One way is to pass completely over the data each time and find the single location \( t \) with the highest likelihood and add it if it gives an increase to the posterior. Alternatively, we can run a single pass over the data, processing it in an online manner, and at each time point, while checking briefly into the future for a better spike, adding a spike if it increases the posterior. Because we are performing a deconvolutional process, after adding a spike we have to remove its contribution to the voltage signal \( v \) so that we don’t have a second waveform attempt to explain that same signal.

Thus, we can infer the spike train in an online manner for a single cluster with a known mean and covariance. However, we have several issues with this setup –
we often don’t know $\mu$ and $\Sigma$ and there are multiple clusters. To handle multiple clusters, suppose that $y$ is drawn instead from a mixture model:

$$y_i \sim \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k)$$

or:

$$y_i | \gamma_i \sim \mathcal{N}(\mu_{\gamma_i}, \Sigma_{\gamma_i})$$
$$\gamma_i \sim \text{categorical}(\pi)$$

This changes the probability of whether $z_i = 1$ by changing our model likelihood, but we can simply evaluate:

$$p(v_i^{t+L}|z_t = 1, \mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K) = \sum_{k=1}^{K} \pi_k \mathcal{N}(v_i^{t+L}; \mathbf{A}\mu_k, \mathbf{V} + \mathbf{A}\Sigma_k\mathbf{A}^T)$$

If there is a spike at time $t$, we can choose which cluster it comes from by setting:

$$\gamma_i = \arg \max_{\gamma_i \in \{1, \ldots, K\}} \pi_{\gamma_i} \mathcal{N}(v_i^{t+L}; \mathbf{A}\mu_k, \mathbf{V} + \mathbf{A}\Sigma_k\mathbf{A}^T)$$

Thus, we only have the problem that we don’t completely know the parameters. If we want to maintain uncertainty in a Bayesian way, we can do this variational Bayesian (VB) methods, where we maintain an approximate separate posterior on the cluster parameters. Details on this can be found in (Carlson et al., 2013a).
Stochastic Spectral Descent for Restricted Boltzmann Machines

2.1 Introduction

Deep learning methods are becoming increasingly popular for feature extraction applications, having produced state-of-the-art results for many classification problems (Bengio, 2012). The impressiveness of the results are unfortunately matched by the (often) extraordinary amount of computation that goes into training them.

Instead of optimizing the cost function generated by a deep network model directly, we can use a divide and conquer strategy via Restricted Boltzmann Machines (RBMs). An RBM is a probabilistic generative model over binary observations and binary hidden nodes, with connections only between the observed visible nodes and unobserved hidden nodes. Indeed, one can exploit RBMs as a building block for learning Deep Restricted Boltzmann Machines (DRBMs) (Salakhutdinov and Hinton, 2009a), Deep Belief Networks (DBN) (Hinton et al., 2006) and Deep Sigmoid Belief Nets (Neal, 1992; Mnih and Gregor, 2014). To this end, this paper introduces a new method, motivated by convex optimization principles, to improve the efficiency...
To explain the novelty in our approach, we first elaborate on the specific computational challenge we address in this paper. The training objective of the RBM model is expressed as a minimization of a non-convex composite objective over a matrix, that connects the hidden and visible units as well as bias terms for both the visible and hidden units. The first term in the objective is convex and captures the partition function of the RBM. The second term is concave and encodes the influence of the observations. Since calculating the gradient of the individual terms imposes a significant computational burden, we leverage Monte Carlo integration via Contrastive Divergence (CD) (Hinton, 2002; Tieleman and Hinton, 2009) to obtain statistical estimates.

Because the cost of estimating the gradients is a major bottleneck, our key contention is that exploiting the information in the gradients more effectively can greatly speed up the overall inference. To achieve this desideratum, we change the space in which our gradient method operates to better match the geometry of the RBM problem. Indeed, we treat the matrix variable explicitly as a matrix by choosing a normed space based on the Shatten-$\infty$ norm (i.e., the spectral norm). We use a similar idea for the bias variables and operate in the $\ell_\infty$-space. Changing the normed space impacts the optimization radius, which can provably improve optimization efficiency in the deterministic setting.

Our algorithm, termed *stochastic spectral descent* (SSD) operates as follows: we obtain the stochastic gradient estimates using CD. We then use an appropriate *sharp*-operator, which applies a nonlinear transformation on the gradient. We update the putative solution using the *gradient-sharp* and a constant step-size. For the matrix variables, the sharp operator takes the singular value decomposition of the gradient and sets all the nontrivial singular values to the trace-norm of the gradient. For the bias terms, the sharp-operator replaces the vector coefficients simply with their sign.
(i.e., ±1) and multiplies the overall vector with the $\ell_1$-norm of the original gradient.

From a deterministic convex optimization perspective, the proposed algorithm is almost classic (Nesterov, 2012). In fact, we can derive the algorithm as a majorization-minimization method by minimizing an upper bound of the objective equation. However, to the best of our knowledge, its application to learning deep networks is novel, where the deterministic gradient estimates are replaced by their stochastic estimates. While we do not provide a rigorous convergence proof of our algorithm in this paper, we provide enough empirical evidence to justify its usage.

Recent work on improving the efficiency of training RBMs largely focus on stochastic gradient descent and its variations. These variations include autotuning of the step-size in SGD (Schaul et al., 2012), the Enhanced Gradient method (Cho et al., 2013), and using a variable metric (Duchi et al., 2010). For training RBMs, we can also change the cost function to improve training quality. Dropout RBM (Srivastava et al., 2014), for instance, empirically reduces the number of dead units. Here, the focus is on using the geometry for theory-based algorithms instead of improving step-size selection or gradient estimates. However, like the Dropout procedure, the SSD empirically learns an RBM that has higher usages of hidden nodes with fewer “dead units.”

2.2 Preliminaries and Model Definitions

2.2.1 Restricted Boltzmann Machines

The RBM is a two-layer binary Markov Random Field, where the observed binary stochastic visible units $v \in \{0, 1\}^M$ have pairwise connections to the binary stochastic hidden units $h \in \{0, 1\}^J$. There are no pairwise connections within the visible units, nor within hidden units. The negative energy for a state $\{v, h\}$ is

$$-E(v, h; \theta) = v^T c + h^T b + v^T W h \quad (2.1)$$
with $\theta = \{c, b, W\}$, $c \in \mathbb{R}^M$, $b \in \mathbb{R}^J$, and $W \in \mathbb{R}^{M \times J}$. The joint probability of the model is defined as $p_\theta(v, h) = \exp(-E(v, h; \theta))/Z_\theta$, where $Z_\theta$ is the partition function. The likelihood for an observation $v$ is:

$$p_\theta(v) = \frac{1}{Z_\theta} \sum_h \exp(-E(v, h; \theta))$$  \hspace{1cm} (2.2)$$

$$Z_\theta = \sum_v \sum_h \exp(-E(v, h; \theta))$$  \hspace{1cm} (2.3)$$

The sum over $h$ denotes the sum over all $2^J$ binary vectors, and likewise for $v$. The optimization goal is to minimize the negative log-likelihood of the model, defined for $N$ data samples $\{v_n\}_{n=1}^N$ as:

$$\arg\min_\theta F(\theta) = -\frac{1}{N} \sum_{n=1}^N \log p_\theta(v_n) = f(\theta) - g(\theta)$$  \hspace{1cm} (2.4)$$

$$f(\theta) = \log \sum_v \sum_h \exp(v^Tc + h^Tb + v^TWh)$$  \hspace{1cm} (2.5)$$

$$g(\theta) = \frac{1}{N} \sum_{n=1}^N \log \sum_h \exp(v_n^Tc + h^Tb + v_n^TWh)$$

The gradients of the objective function are dependent on expectations of the model:

$$\nabla_W F(\theta) = \mathbb{E}_{p_\theta(v, h)}[vh^T] - \mathbb{E}_{p_0}[vh^T]$$  \hspace{1cm} (2.6)$$

$$\nabla_b F(\theta) = \mathbb{E}_{p_\theta(v, h)}[h] - \mathbb{E}_{p_0}[h]$$  \hspace{1cm} (2.7)$$

$$\nabla_c F(\theta) = \mathbb{E}_{p_\theta(v, h)}[v] - \frac{1}{N} \sum_{n=1}^N v_n$$  \hspace{1cm} (2.8)$$

where $p_0$ denotes $\frac{1}{N} \sum_{n=1}^N p_\theta(h|v_n)$. The gradients cannot be directly calculated in all but the smallest problems, so Monte Carlo integration is used to estimate the gradients. Sampling from $p_\theta(v, h)$ is intractable, so Persistent Contrastive Divergence (PCD) sampling schemes (Hinton, 2002; Tieleman and Hinton, 2009) are used to generate approximate samples.
In general, explicit objective function evaluations are intractable, but Annealed Importance Sampling (Salakhutdinov and Murray, 2008) allows tight estimates of the log partition function. Given the estimate on $\log Z_\theta$, the objective function is analytic.

### 2.2.2 Deep Belief Nets

A Deep Belief Net (DBN) (Hinton et al., 2006) is an $(L + 1)$-layer deep model of binary nodes. The bottom layer consists of the visible units $v \in \{0, 1\}^M$, and $L$ hidden layers are stacked on top. Each hidden layer is a binary vector, $h^\ell \in \{0, 1\}^{J^\ell}$. The top two hidden layers are jointly drawn from a RBM:

$$
p(h^L, h^{L-1}) \propto \exp(-E(h^L, h^{L-1}))
$$

$$
-E(h^L, h^{L-1}) = (h^L)^T b^L + (h^{L-1})^T b^{L-1} + (h^{L-1})W^L h^L
$$

Given $h^{L-1}$, the DBN is a directed generative model with the same form of the Sigmoid Belief Net (Neal, 1992). The hidden layers, $h^1, \ldots, h^{L-2}$, and the visible units are generated by:

$$
p(h^\ell_j|h^{\ell+1}) = \text{Bern}(\sigma(b^\ell_j + \sum_{k=1}^{J^{\ell+1}} W^\ell_{jk} h^{\ell+1}_k))
$$

$$
p(v_m|h^1) = \text{Bern}(\sigma(c^1_m + \sum_{j=1}^{J^1} W^1_{mj} h^1_j))
$$

Learning a Sigmoid Belief Network or a DBN is a challenging problem, and much recent work has explored approximate learning using recognition models (Hinton et al., 1995; Mnih and Gregor, 2014; Gregor et al., 2014). In (Hinton et al., 2006), it was shown that greedy layer-wise training of RBMs would provide an effective “pre-training” initialization of the DBN.
The pre-training method starts by training $v$ and $h^1$ as an RBM. Then, using samples from the first-layer RBM for $h^1$, $h^1$ and $h^2$ are trained as an RBM. This procedure continues until the last layer is trained. “Fine-tuning” updates using the cost function of the DBN show only minor performance gains (Hinton et al., 2006).

The model is evaluated by using variational evidence lower bounds. Considering a 3-layer DBN, the model likelihood is lower bounded by:

$$\log p(\theta, v) \geq E_{q_1}[\log p(v|h^1; W^1, c)] - E_{q_2}[\log q(h^1)] + E_{\theta}[\log p(h^1; W^2, b^1, b^2)]$$  \hspace{1cm} (2.13)

The second hidden layer, $h^2$, is analytically integrated out, and the variational distributions $q$ are set during the greedy layer-wise training. Recent methods have explored methods to learn better variational distributions (Mnih and Gregor, 2014; Gregor et al., 2014) in these models. To evaluate the term $E_{q_2}[p(h^1; W^2, b^1, b^2)]$, first the log partition function is estimated by Annealed Importance Sampling (AIS) and then $\exp(-E_{q}(v))$ is estimated through Monte-Carlo integration. Further details can be found in (Salakhutdinov and Murray, 2008).

2.3 Related Work

Other optimization procedures have attempted to adapt to geometry. ADAgrad (Duchi et al., 2010) provides a element-wise step size scheme that dynamically adapts to the geometry of the data with theoretical guarantees on the regret bound. In machine learning problems with very noisy gradients, ADAdelta (Zeiler, 2012) and RMSprop (Tieleman and LeCun, 2012) were introduced to provide schemes that adapt to the geometry of the problem with alternative step-size reduction. These methods have shown empirically good performance in autoencoder models, and here we will show results on both ADAgrad and RMSprop in RBMs. These methods all attempt to learn the geometry, where
the proposed algorithm is given the geometry of the problem.

Recent work has also focused on the SGD step-size. (Schulz et al., 2010) explored the convergence of RBM when a constant step size is used, as well as model divergence when using smaller numbers of Contrastive Divergence steps. (Schaul et al., 2012) explored automatic step-size selection for deep models, but did not specifically consider RBMs. (Yuille, 2004) explored when the CD based descent would converge.

Dropout (Srivastava et al., 2014) exploits model averaging to improve the gradient descent algorithms by preventing co-adaptation. Empirically, using a dropout RBM models decreases the number of dead units as compared to traditional RBMs. However, Dropout RBMs differ from traditional RBMs due to the penalization scheme on the model, and does not give maximum likelihood estimates of the model, so is not compared to here.

2.4 Learning Restricted Boltzmann Machines

To the best of our knowledge, virtually all RBM training approaches operate in the Euclidean space, which leads to stochastic gradient descent and its variations. Because the gradients are expensive to estimate, exploiting the information in the gradients can dramatically speed up inference.

The non-convex objective function of the RBM is

\[ F(\theta) = f(\theta) - g(\theta) \quad (2.14) \]

where \( f(\theta) \) comes from the log partition function and \( g(\theta) \) depends on the data. Both \( f(\theta) \) and \( g(\theta) \) are convex functions, so the objective function can be analyzed as the sum of a convex and concave function. The key algorithmic idea is the use of majorization-minimization to update the parameters. We first find an upper bound on the function \( F(\theta) \), and then steps are taken to minimize this upper bound. The function \( f(\theta) \) is examined first.
2.4.1 Upper bounds on the log partition function

Lipschitz-gradient functions have a quadratic upper bound with a function-dependent parameter $L$:

$$f(u) \leq f(v) + \langle \nabla f(v), u - v \rangle + \frac{L}{2} ||u - v||^2$$ (2.15)

If the norm is $\ell_2$, this bound is minimized by taking a step in the negative direction of the gradient, with $u = v - L^{-1}\nabla f(v)$. Our function $f(\theta)$ is not dependent on the $\ell_2$ and Frobenius norms, but instead is bound by the $\ell_\infty$ and the $S_\infty$ norms.

The analysis in this paper hinges on the form of this inequality for functions of the form $\log \sum_j \omega_j \exp(u_j)$:

**Theorem 1.** Define a function $lse_\omega(u) = \log \sum_{j=1}^J \omega_j \exp(u_j)$, then this function has an upper bound independent of $\omega$:

$$lse_\omega(v) \leq lse_\omega(u) + \langle \nabla lse_\omega(u), v - u \rangle + \frac{1}{2} ||v - u||^2_{\infty}$$ (2.16)

**Proof.** See Supplemental Section A.1. \qed

Critically, this inequality is independent of the constant vector $\omega$.

The bound given in Theorem 1 is different than the $\ell_2$ bound on the $lse$ function in the literature (Böhning, 1992), which in the RBM case is

$$lse_\omega(v) \leq lse_\omega(u) + \langle \nabla lse_\omega(u), v - u \rangle + \frac{1}{4} ||v - u||^2_{2}$$ (2.17)

However, in the RBM the $lse$ function has a dimensionality of the number of possible states, which is $2^{M+J}$. Using the established $\ell_2$ bound instead of the $\ell_\infty$ bound would lead to exponentially looser estimates in realistic problem sizes, as well as not correctly capturing the geometry. See Supplemental Section A.3 for further details.

Theorem 1 is directly utilized to bound the function $f(\theta)$. Let $\theta_k = \{b^k, c^k, W^k\}$, then:
Theorem 2. The bias terms have an upper bound

\[
f(b, c^k, W^k) \leq f(\theta^k) + \langle \nabla_b f(\theta^k), b - b^k \rangle + \frac{J}{2} ||b - b^k||^2 
\]

\[
f(b^k, c, W^k) \leq f(\theta^k) + \langle \nabla_c f(\theta^k), c - c^k \rangle + \frac{M}{2} ||c - c^k||^2 
\]

(2.18) (2.19)

Proof. See Supplemental Section A.1.

Given the gradient with respect to \(b\), Equation 2.18 is bound independently of \(c^k\) and \(W^k\), but the gradient is dependent on the current values of these parameters.

Next, the majorization function on \(W\) for \(f\) is bound by the Shatten-\(\infty\) norm:

Theorem 3. \(f(b^k, W^k)\) has an upper bound:

\[
f(b^k, c^k, W) \leq f(\theta^k) + \text{tr}((W - W^k)^T \nabla_W f(\theta^k)) + \frac{MJ}{2} ||W - W^k||_{\infty}^2 
\]

(2.20)

Proof. See Supplemental Section A.1.

2.4.2 Majorization bounds on the learning objective

Because of the convexity of \(g(\theta)\), the bounds on \(g(\theta)\) are:

\[
g(b, c^k, W^k) \geq g(\theta^k) + \langle \nabla_b g(\theta^k), b - b^k \rangle 
\]

(2.21)

\[
g(b^k, c, W^k) \geq g(\theta^k) + \langle \nabla_c g(\theta^k), c - c^k \rangle
\]

(2.22)

\[
g(b^k, c^k, W) \geq g(\theta^k) + \langle \nabla_w g(\theta^k), W - W^k \rangle
\]

Combining these inequalities with the inequalities on \(f(\theta)\) for the bias terms \(b\)
and \( c \) gives

\[
F({b, c^k, W^k}) \leq F(\theta^k) + \langle \nabla_b F(\theta^k), b - b^k \rangle + \frac{J}{2} \|b - b^k\|_\infty^2
\]

(2.23)

\[
F({b^k, c, W^k}) \leq F(\theta^k) + \langle \nabla_c F(\theta^k), c - c^k \rangle + \frac{M}{2} \|c - c^k\|_\infty^2
\]

(2.24)

These majorization bounds for the bias terms on \( f \) are not minimized in the direction of the gradient. Instead the minimizers are

\[
b^* = b^k - \frac{1}{f} \|\nabla_b f(\theta^k)\|_1 \times \text{sign}(\nabla_b F(\theta^k))
\]

(2.25)

\[
c^* = c^k - \frac{1}{M} \|\nabla_c f(\theta^k)\|_1 \times \text{sign}(\nabla_c F(\theta^k))
\]

(2.26)

Derivations for Equations 2.25 and 2.26 can be found in Supplemental Section A.2.

The bound on the objective function for \( W \) is:

\[
F({b^k, c^k, W}) \leq F(\theta^k) + \text{tr}(\nabla_W F(\theta^k)(W - W^k)) + \frac{MJ}{2} \|W - W^k\|^2_{S_\infty}
\]

(2.27)

Because the term \( \|W - W^k\|^2_{S_\infty} \) is dependent on the largest eigenvalue, the minimization of this majorization function is not maximized in the direction of the gradient. Representing the gradient by its singular value decomposition \( \nabla_W F(\theta) = \sum_{k=1}^{\min(M,J)} \lambda_k a_k b_k^T \), this bound is minimized at:

\[
W^* = W - \frac{\|\lambda\|_1}{MJ} \sum_{k=1}^{\text{rank}(\nabla_W F(\theta^k))} a_k b_k^T
\]

(2.28)

The derivation for Equation 2.28 can be found in Supplemental Section A.2.
Algorithm 1 RBM Stochastic Spectral Descent

1: Inputs: \( v_1, \ldots, v_N, J, \alpha \)
2: Initialize: \( b = 0, c = 0, W_{mj} \sim \mathcal{N}(0, 1) \)
3: for \( i = 1, \ldots \)
4: Sampling a minibatch \( v_1, \ldots, B \)
5: \([dW, db, dc] = \text{CDGradEstimate}(v_1, \ldots, B, \theta)\)
6: \( b = b - (\text{sum}(\text{abs}(db))/J)\text{sign}(db) \)
7: \( c = c - (\text{sum}(\text{abs}(dc))/M)\text{sign}(dc) \)
8: \([A, \lambda, B] = \text{svd}(dW)\)
9: \( W = W - \alpha(\|\lambda\|_1/MJ)AB^T \)
10: end for

2.4.3 Stochastic Spectral Descent

In Section 2.4.2 the majorization bounds on the objective function are given for each set of parameters \( b, c, \) and \( W \). In a gradient descent scheme, the gradient would be exactly calculated between each of the updates. In an RBM, the cost of estimating the gradient is high and mini-batches are necessary for large data to reasonably estimate the gradients. To set the step-size on \( W \) on the SSD, we can simply use \( 1/MJ \), which can exhibit slow convergence. However, the presence of local strong convexity in \( g(\theta) \) suggests that we can choose a higher-step-size, which might potentially depend on the minibatch size. Overall, with some experimentation, we found the following setting robust in many of the scenarios we tested, the step-size on \( W \) is set to \( 1/\sqrt{MJ} \), and \( b \) and \( c \) are set to \( 1/M \) and \( 1/J \) respectively. This procedure is summarized in Algorithm 1.

2.4.4 Discussion of potential issues

The proposed algorithm exploits the natural geometry of the logsumexp function and how it relates to the RBM cost function. There are a few concerns about using these projections instead of using the gradients directly, including the additional computational cost of the projection, the inexact nature of the estimating the gradient, and the effect of small mini-batches on the ability to estimate the eigenvectors of the gradient. These concerns are addressed below.
Computational Cost of the Spectral Norm

The cost of computing the singular value decomposition is in general not trivial, and requires $O(MJ\min(M, J))$ time to calculate. However, in the RBM, the cost of estimating the gradient is expensive. The cost of a single Gibbs sample from $v^{(k-1)} \rightarrow v^k$, the key operation of contrastive divergence, costs $O(MJ)$. A mini-batch of size $N_{batch}$ and Contrastive Divergence order $C$ (the number of Gibbs iterations) costs $O(MJN_{batch}C)$. Previous work has shown that a large Contrastive Divergence order is necessary to fit the generative model, with $C \geq 25$ (Salakhutdinov and Murray, 2008). In our experiments, the computational cost of calculating the SVD is essentially free compared to the computational cost of estimating the gradient.

Issues with Noisy Gradient Estimation

The method for estimating the gradient is to take a mini-batch of data and estimate the gradient on $g(\theta)$ and estimate the gradient on $f(\theta)$ with samples via persistent Contrastive Divergence. From SVD theory, if the small batch sizes cause additive white Gaussian noise, the expectation of the eigenvalues is biased upwards. However, the expectations of the subspaces does not change. In practice, these issues don’t seem to affect the performance.

Effect of mini-batches smaller than $\min(M, J)$

In many cases, it’s of interest to use very small mini-batches of size $r << \min(M, J)$. In this case rank$(\nabla_W F(\theta)) \leq 2r$, which gives a low rank matrix. Learning is noisier but still provides competitive results, as shown in Section 2.5.1.

2.5 Experiments

Experiments were performed on simulated data and the MNIST dataset. Estimation of the model likelihood $\frac{1}{N} \sum_n \log p(v_n) = -F(\theta)$ is calculated through AIS (Salakhut-
Figure 2.1: Results on simulated $M = 100$, $J = 25$ datasets, all plots shown in the log scale on iterations (Left) $W$ set to a full matrix. (Middle) $W$ set to a matrix with rank 5. All algorithms converge to the same approximate objective value, but SSD gets to that level in fewer iterations. (Right) $W$ set to a full matrix, but the batch size is set to 10.

Figure 2.2: Learning curves for a single-layer RBM on MNIST showing the mean log $p(v)$ versus the number of iterations. (Left) $J=25$ (Middle) $J=100$ (Right) $J=500$. SSD shows improved performance over competing algorithms across all levels of $J$ for this dataset.

dinov and Murray, 2008) with 10,000 temperature scales evenly spaced from 0 to 1 and 100 particles. The base distribution in AIS was set to independent binary draws at the mean of the observations. The performance of SSD was compared to stochastic gradient descent (SGD), ADAgrad (Duchi et al., 2010), and RMSprop (Tieleman and LeCun, 2012). The step-size in ADAgrad was set to 0.1, and the step size in RMSprop was set to 0.1 with a decay parameter of 0.999, which were optimal at both $J = 25$ and $J = 100$ in the MNIST data set. We compared to SGD with constant step sizes set at 0.1 and 0.01. The algorithms all shared the same initialization for
each problem with the bias terms set to zero vectors, and the pairwise weights $W$ initialized to random Gaussian draws with variance 0.01. The CD order was set to 25; this was chosen based on the results of (Salakhutdinov and Murray, 2008), that empirically showed a higher number was necessary for good model estimates. The batch size was set to $2J$ unless otherwise stated.

2.5.1 Synthetic Data

Experiments were run on synthetic data sets with $M = 100$ and $J = 25$. $N = 5000$ data samples were used. The observations were approximately generated from the model by first drawing random binary vectors, and then running 10,000 Gibbs iterations using the true parameters.

In the first synthetic data example, $W$ was set to random Gaussian weights with variance 0.5 to mimic a full rank matrix, and the biases were set to zeros. Figure 2.1 (left) shows the mean log $p(v)$ for all of the data sample versus the log of the number of iterations. Here, SSD reaches the saturating value with a full order of magnitude fewer iterations. Larger step-sizes on SGD were unstable.

The next simulated dataset was generated from a low-rank matrix on $W$. This is meant to address concerns on the ability of the algorithm to fit low-rank data when the steps on $W$ are full-rank. The pairwise matrix $W$ was set to $A^T B$ with $A \in \mathbb{R}^{M \times R}$ and $B \in \mathbb{R}^{J \times R}$, with $R = 5$. $A$ was generated from random Gaussians with variance 1, and $B$ was generated from random Gaussian with variance 0.5. The biases were set to zero. Here, the performance from SSD is still better than SGD, although the performance margin is smaller.

Finally, to examine the effects of small batch sizes, a random full-weight matrix was generated as in the first example. In this example, SSD still performs quite well. The improvement in performance is less than the cases where the gradient is better estimated, however.
Figure 2.3: Train (solid line) and test (dashed line) model likelihood after 10,000 iterations for different values of $J$ with different optimization methods.

Figure 2.4: Per-unit empirical entropy for a $J = 500$ RBM training on the MNIST data set after 50,000 iterations. SSD shows higher entropies.

2.5.2 MNIST

The MNIST digit dataset contains 60,000 training and 10,000 test images of handwritten digits (0 to 9) of size $28 \times 28$ pixels. Each image was vectorized and binarized as in (Salakhutdinov and Murray, 2008).

RBM

The learning curves for different levels of $J$ are first examined. In Figure 2.2(left), the learning curves for $J = 25$ are shown. Here, SSD gives dominant performance and
has a fairly smooth curve. ADAgrad and RMSprop are surprisingly uncompetitive on this problem size; further tuning parameters on these algorithms did not result in improved performance. After 30,000 iterations, SGD has yet to approach the performance given by SSD, and SSD is able to approach this level of performance quickly. While the focus of this paper is on RBM training, it is interesting to examine the testing performance as well. The mean test log likelihoods were -129.76 for SSD, -131.73 for SGD, -137.02 for RMSprop, and -135.97 for ADAgrad after 30,000 iterations. The MNIST dataset is unusual in this case because the test set has a bigger likelihood than the training dataset. This is consistent with the literature on networks of this size (Salakhutdinov and Murray, 2008). The learning curves on these algorithms are quite noisy, but this is consistent with observations of long runs of SGD in RBMs (Schulz et al., 2010).

In Figure 2.2(middle), the learning curves for $J = 100$ are shown. SSD once again shows dominant performance. After 30,000 iterations, the test performance is -97.11 for SSD, -101.78 for SGD, -99.82 for RMSprop, and -100.10 for ADAgrad.

In Figure 2.2(right), the learning curves for $J = 500$ are shown. Here, both ADAgrad and RMSprop show significant improvement over SGD, but still lag behind the performance of SSD. The test performance is -86.77 for SSD, -89.61 for RMSprop, -90.29 for SGD with parameter 0.1, and -90.36 for SGD with parameter 0.01. In fact, after 50,000 iterations, SGD is at roughly the same level from SSD at 5,000. Note that this is marginally below the value reported for the test set in (Salakhutdinov and Murray, 2008), but there a penalized likelihood scheme was used instead of maximum likelihood, giving less overfitting. Matching the penalized likelihood scheme with SSD gives a test performance of -85.65 with a training likelihood of -82.60 after 50,000 iterations.

To investigate the effect of using this algorithm on a given number of iterations, the number of iterations was set to 10,000 and run for $J$ values varying from 25 to 500.
The results of these simulations are shown in Figure 2.3. Because the computational costs of running the algorithm are similar for all algorithms (in our code, a single iteration took 3% longer per iteration in SSD than SGD for $J=100$), this provides a metric of how well each algorithm will do in a given amount of time.

Figure 2.3 shows the training mean log probability after 10,000 iterations for all algorithms considered. It’s interesting to note that both RMSprop and ADAgrad seem to be improving in comparative performance as the problem size increases, but SSD is provides the best performance over the experimental range. In fact, after 10,000 iterations, SSD has already reached the best reported training likelihood in the literature.

The usage of units is also considered. It was shown in (Srivastava et al. (2014)) that using a Dropout-RBM results in fewer “dead” units, which are units that are nearly completely on or nearly completely off. Here, the per-unit binary entropy of the empirical distribution of each unit is used to measure how much a unit is effectively used. This result for a $J = 500$ RBM is shown in Figure 2.4, which shows that SSD hidden units have higher entropies than SGD hidden units. Thus, each unit seems to be utilized to a higher extent by training with SSD instead of SGD.

**DBN**

Here a 3-layer Deep Belief Net is considered. To examine how the different learning speeds of optimization procedures propagate on the deep model, the first layer was trained with the different algorithms for 1000, 5000, 10000, 20000, and 30000 gradient steps for $J = 100$ hidden nodes. A DBN with 200 nodes in the second hidden layer was learn based on this output, and the lower bound was evaluated. 10,000 iterations were used in the second layer. Because the second layer network was much smaller, this was an effective number of samples to converge and was much quicker. The lower bounds are shown in Table 2.1. SSD shows a performance improvement at every
Table 2.1: A DBN with 100 hidden nodes in the first layer and 200 hidden nodes in the second layer. The left column denotes the number of training samples on the first layer, and then the corresponding lower bounds on the log likelihood given for both the training and the testing set.

<table>
<thead>
<tr>
<th>L1 Iters</th>
<th>SSD</th>
<th>ADAgad</th>
<th>RMSprop</th>
<th>SGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1k, Train</td>
<td>-118.3</td>
<td>-125.5</td>
<td>-124.9</td>
<td>-133.9</td>
</tr>
<tr>
<td>2k, Train</td>
<td>-106.2</td>
<td>-111.7</td>
<td>-110.0</td>
<td>-116.8</td>
</tr>
<tr>
<td>10k, Train</td>
<td>-104.7</td>
<td>-107.7</td>
<td>-106.2</td>
<td>-113.9</td>
</tr>
<tr>
<td>20k, Train</td>
<td>-101.6</td>
<td>-103.2</td>
<td>-102.0</td>
<td>-109.3</td>
</tr>
<tr>
<td>30k, Train</td>
<td>-101.0</td>
<td>-102.3</td>
<td>-101.6</td>
<td>-108.0</td>
</tr>
<tr>
<td>1k, Test</td>
<td>-117.5</td>
<td>-124.4</td>
<td>-124.1</td>
<td>-132.4</td>
</tr>
<tr>
<td>5k, Test</td>
<td>-105.6</td>
<td>-110.7</td>
<td>-109.8</td>
<td>-115.7</td>
</tr>
<tr>
<td>10k, Test</td>
<td>-103.5</td>
<td>-106.4</td>
<td>-105.2</td>
<td>-113.2</td>
</tr>
<tr>
<td>20k, Test</td>
<td>-102.3</td>
<td>-104.6</td>
<td>-103.0</td>
<td>-110.2</td>
</tr>
<tr>
<td>30k, Test</td>
<td>-101.7</td>
<td>-103.4</td>
<td>-102.7</td>
<td>-108.7</td>
</tr>
</tbody>
</table>

number of gradient steps, often by several nats. All algorithms that use geometry outperform SGD here.

2.6 Discussion

We have introduced a novel descent method for RBMs, Stochastic Spectral Descent, which utilizes the natural geometry of the RBM cost function by operating in the $\ell_\infty$ and $S_\infty$ norms. Empirical results suggest that this is a good general purpose algorithm for RBMs. To the best of our knowledge, the bounds presented in this paper are novel, and the algorithm gives state-of-the-art learning performance. Further improvements may be obtained by using better gradient estimators, either by using tempering schemes (Salakhutdinov, 2010; Cho et al., 2010) or the Enhanced Gradient (Cho et al., 2013). Future work will including extending this analysis directly to deep models, as well as related models, including the Sigmoid Belief Net.
3 Descent Algorithms for Deep Discrete Graphical Models

3.1 Introduction

Deep graphical models are becoming increasingly popular as a general feature extraction method, and have produced state-of-the-art performance in many classification problems (Bengio, 2012). These graphical models are based on having both visible observations and also hidden nodes. Undirected graphical models, or Markov Random Fields (MRFs), have been successfully applied to a wide number of data types. This includes binary images with the Restricted Boltzmann Machine (RBM) (Hinton, 2002) and count modeling with the Replicated Softmax (Salakhutdinov and Hinton, 2009b). They have been extended to deeper forms, including the Deep RBM (Salakhutdinov and Hinton, 2009a). Directed graphical models, also known as Bayesian Networks or Belief Nets (BN), have been increasing in popularity due to the tractability of variational methods (Frey and Hinton, 1999) and recognition models (Gregor et al., 2014).

These models provide impressive performance, but this performance is unfortu-
nately not free from large computational requirements. Learning schemes typically proceed by using stochastic gradient descent (SGD), or with methods that attempt to locally adapt to the geometry, including ADAgrad (Duchi et al., 2010) and RM-Sprop (Tieleman and LeCun, 2012). However, the computational bottleneck is the gradient estimation, where Markov Chain Monte Carlo (MCMC) methods, including Contrastive Divergence (CD) methods (Hinton, 2002), are used in both MRFs and BNs to estimate gradients.

Because the gradient is the computational bottleneck, we focus on improving the use of the gradients. To this end, we show that viewing BNs as a Boltzmann energy distribution allows a joint framework for analyzing both discrete MRFs and BNs. This analysis leads to novel majorization-minimization schemes. We introduce general forms of global bounds on the log likelihood of discrete MRFs and BNs models with hidden nodes. This theory is applied to to the Replicated Softmax-RBM (Salakhutdinov and Hinton, 2009b), the Sigmoid Belief Net (Neal, 1992), the Replicated Softmax Belief Nets (Mnih and Gregor, 2014), and the Deep Belief Net (Hinton et al., 2006).

Intriguingly, these models often have majorization functions that are not naturally bound on the $\ell_2$ or Frobenius-norm. Instead, the bound reveals information about the geometry of the learning problem. Using these bounds, we propose stochastic methods that take simple nonlinear operations on the gradient estimate to minimize the majorization bound. Empirically, these nonlinear stochastic methods provide quicker learning than stochastic gradient descent.

In additional to MCMC methods, Belief Nets are often fit by variational EM methods (Frey and Hinton, 1999), stochastic variational inference (Hoffman et al., 2013), and recognition models (Gregor et al., 2014). The global bounds and majorization-minimization framework we develop holds for both the log-likelihood and the Evidence Lower-Bound Objective (ELBO) used in variational methods. The nonlinear
stochastic methods that we develop show improved performance for both the (penalized) maximum likelihood and variational inference problem.

### 3.1.1 Related Work

In the belief net framework, there has been recent work on improving gradient estimation for recognition models. This has been achieved both with variance reduction techniques (Mnih and Gregor, 2014), and techniques useful in variational auto-encoders (Kingma and Welling, 2014). In the Sigmoid Belief Net, (Gan et al., 2015) developed an auxiliary variable sampler for Bayesian posterior inference.

There has been effort to adapt the step-size of SGD in the RBM (Bengio, 2012), and generally in deep models (Schaul et al., 2012). There is work on improving the quality of the gradient estimate via Persistent Contrastive Divergence (Tieleman and Hinton, 2009), as well as using transformation of variables for the Enhanced Gradient Method (Cho et al., 2013). For the specific case of the Binary Restricted Boltzmann Machine, (Carlson et al., 2015b) introduced a global bound on the log-likelihood function using the Shatten-$\infty$ norm, and showed that a descent method based on this bound improved performance over SGD, ADAgrad, and RMSprop.

There are general-purpose first-order methods that adapt to the geometry by using the gradient information. ADAgrad (Duchi et al., 2010) provides an element-wise step size scheme that dynamically adapts to the geometry of the data. This approach has theoretical guarantees on the regret bound in strongly convex problems. ADAdelta (Zeiler, 2012) and RMSprop (Tieleman and LeCun, 2012) are similar to ADAgrad, but the step-size decreases more slowly and can improve performance in real-world problems. Recent work in optimization has shown that coordinate descent methods can improve optimization over methods based on $\ell_2$-norms (Nesterov, 2012).

Instead of analyzing the form of the model to infer its geometry, there are first-order methods that attempt to adapt by using the gradient information. ADAgrad
(Duchi et al., 2010) provides an element-wise step size scheme that dynamically adapts to the geometry of the data. This approach theoretical guarantees on the regret bound in strongly convex problems. ADAdelta (Zeiler, 2012) and RMSprop (Tieleman and LeCun, 2012) are similar to ADAgrad, but the step-size decreases more slowly and can improve performance in real-world problems. Recent optimization work has shown that using majorization-minimization schemes besides the $\ell_2$ norm can improve both empirical and theoretical performance (Nesterov, 2012).

3.2 Global Majorization Bounds

Assume that a model has both visible observations (units) $v \in V$ and hidden units $h \in H$ where both $V$ and $H$ are finite sets, such as the binary vector $\{0,1\}^M$. The joint probability distribution for $\{v,h\}$ is parameterized by $\theta$, and the marginal likelihood on the observations is $p_\theta(v) = \sum_{h \in H} p_\theta(v,h)$. The maximum likelihood (ML) estimator for observations $\{v\}_{n=1, N}$ is $\theta^{ML} = \arg \max_\theta \prod_n p_\theta(v_n)$. This probability distribution can also be represented in terms of a Boltzmann or Gibbs distribution with an energy function $-E_\theta(v,h)$ that is uniquely defined (up to a constant) by the model, and then

$$p(v,h) = \frac{\exp(-E(v,h))}{Z(\theta)}$$

with $Z(\theta)$, called the partition function, forcing the sum of the probability for all possible states $\{v,h\}$ to equal 1. The objective function can be written as the sum of the data term $f(\theta)$ and the log-partition function $\log Z(\theta)$,

$$\theta^{ML} = \arg \min_\theta F(\theta) = f(\theta) + \log Z(\theta)$$

$$f(\theta) = -\frac{1}{N} \sum_n \log \sum_h \exp(-E_\theta(v_n,h))$$

$$\log Z(\theta) = \log \sum_v \sum_h \exp(-E_\theta(v,h))$$
Although the models we discuss in Section 4.2 are non-convex, it is possible to derive a global upper bound on $F(\theta)$ by combining upper bounds on the data term $f(\theta)$ and the log-partition function $\log Z(\theta)$. We first note that there is a general bound on the data term.

**Theorem 4.** The difference between $f(\theta)$ and $f(\phi)$ for parameters $\theta$ and $\phi$ is bound by

$$ f(\phi) \leq f(\theta) + \langle \nabla_{\theta} f(\theta), \phi - \theta \rangle - \min_{n,h}(-E_{\phi}(v_n, h) + E_{\theta}(v_n, h) - \langle \nabla_{\theta}(-E(v_n, h)), \phi - \theta \rangle) $$

(3.1)

The relationship depends on the error in the first order Taylor approximation on the energy function.

**Proof.** See Supplemental Section B.1. \qed

This bound holds generally and does not depend on the form of the energy function, except that it is smooth in the parameters $\theta$. Note that if the negative energy function is convex, like in the RBM, then a simpler version of this bound is

$$ f(\phi) \leq f(\theta) + \langle \nabla_{\theta} f(\theta), \phi - \theta \rangle $$

(3.2)

This occurs because Jensen’s inequality requires that the min term in (3.1) is non-negative.

**Theorem 5.** The difference in the log partition function evaluated at parameters $\theta$ and $\phi$ has a relationship that

$$ \log Z(\phi) \leq \log Z(\theta) + \langle \nabla_{\theta} \log Z(\theta), \phi - \theta \rangle $$

$$ + \max_{v,h}(-E_{\phi}(v, h) + E_{\theta}(v, h)) $$

$$ - \langle \nabla_{\theta}(-E(v, h)), \phi - \theta \rangle $$

$$ + \frac{1}{2} \max_{\phi}(-E_{\phi}(v, h) + E_{\theta}(v, h))^2 $$

(3.3)
Proof. See Supplemental Section B.2.

Unfortunately, standard bound techniques on general energy functions give results that are dependent on $O(||\phi - \theta||^4)$ in Theorem 5. However, we focus on two broad special cases. First, in many generative BN models, the partition function is known analytically and is a constant with only the data term changing. In this case, an upper bound can be found by utilizing only Theorem 4. The second special case is when the energy function is linear, such as the case in Ising Models, Binary RBMs, or in the RBM part of the Deep Belief Net. In this case, Theorem 2 reduces to a simpler form,

$$f(\phi) \leq f(\theta) + \langle \nabla_{\theta} f(\theta), \phi - \theta \rangle$$

$$+ \frac{1}{2} \max_{v,h} \langle \nabla_{\theta} (\mathcal{E}(v, h)), \phi - \theta \rangle^2 \quad (3.4)$$

A proof of (3.4) can be found in Supplemental Section B.3. Essentially, the first max statement in (3.3) drops out because the first order Taylor’s approximation in a linear function is exact. The second max statement is simplified to only depend on the gradient that determines the difference between the energy functions exactly.

Given Theorems 4 and 5, the global bound on $F(\theta)$ is the combination of the upper bound on (3.1) and (3.3) (or (3.2) and (3.4) if appropriate). Note that $\nabla F(\theta) = \nabla f(\theta) + \nabla (\log Z(\theta))$, so the bound is dependent on the gradient with respect to $F(\theta)$.

If instead of using the ML estimator, a penalized ML or maximum a posteriori scheme is used, the global lower bounds have an additional term due to the penalization on the parameters.

3.2.1 Variational Methods

Instead of directly using the model likelihood to estimate model parameters, variational methods may be utilized to provide a lower bound to the model likelihood,
replacing the true posterior $p_\theta(h|v)$ over the hidden units with a simple, tractable form $q(h)$. The Evidence Lower Bound Objective (ELBO) uses this variational posterior to give a lower bound on the model likelihood. The ELBO is then maximized instead of the model likelihood,

$$\log p_\theta(v) \geq \mathcal{L} = \mathbb{E}_{q(h)}[\log p_\theta(h, v) - \log q(h)]$$

$$\mathcal{L} = -g(\theta, q) - \log Z(\theta) - \mathbb{E}_{q(h)}[\log q(h)]$$

$$g(\theta, q) = -\log \sum_h \exp(-E_\theta(v, h) + \log q(h))$$

In this form, it is straightforward that the bound on the log partition function will hold. In the parallel to (3.1), the difference between the two parameter sets $g(\theta, q)$ and $g(\phi, q)$ has the bound

$$g(\phi, q) \leq g(\theta, q) + \langle \nabla_\theta g(\theta, q), \phi - \theta \rangle - \min_{n,h}(-E_\phi(v_n, h) + E_\theta(v_n, h))$$

$$+ \langle \nabla_\theta(-E(v_n, h)), \phi - \theta \rangle$$

(3.5)

The proof of (3.5) is identical in form to the proof of Theorem 4 in Supplemental Section B.1. Using (3.5), the bounding techniques hold for both the likelihood and the ELBO.

### 3.3 Model Definitions, Majorization Bounds, and Descent Schemes

Here we show the application of Theorems 4 and 5 to specific types of models, and we show that similar directed and undirected graphical models have similar bounds on their parameters.
3.3.1 Binary Models: Sigmoid Belief Nets and Binary Restricted Boltzmann Machines

Both the Sigmoid Belief Net (SBN) (Neal, 1992) and the Binary RBM (Hinton, 2002) consist of a two layer model with visible units \( v \in \{0, 1\}^M \) and hidden units \( h \in \{0, 1\}^J \) with parameters \( \theta = \{c, W, b\} \), where \( c \in \mathbb{R}^M \), \( b \in \mathbb{R}^J \), and \( W \in \mathbb{R}^{M \times J} \).

Both models have the relationship that

\[
p_{\theta}(v|h) = \prod_{m=1}^{M} p_{\theta}(v_m|h) = \prod_{m=1}^{M} \text{Bern}(v_m; \sigma([c + W h]_m))
\]

However, because the SBN is a directed graphical model and the RBM is an undirected graphical model, the relationship between the hidden units and the visible units is different. Specifically, the SBN has a form such that the hidden nodes are simple to draw \textit{a priori} and the RBM has posterior units that are simple to draw \textit{a posteriori}. This is summarized by

\[
\text{SBN: } p_{\theta}(h) = \prod_{j=1}^{J} \text{Bern}(h_j; \sigma(b_j)) \\
\text{RBM: } p_{\theta}(h|v) = \prod_{j=1}^{J} \text{Bern}(h_j; \sigma([b + W^T v]_j))
\]

This undirected versus directed relationship leads to similar but different energy functions, with

\[
\text{SBN: } -E_\theta(v, h) = v^T c + v^T W h + h^T b \\
- \sum_{m=1}^{M} \log(1 + \exp([c + W h]_m)) - \sum_{j=1}^{J} \log(1 + \exp(b_j)) \\
\text{RBM: } -E_\theta(v, h) = v^T c + v^T W h + h^T b
\]

Although the SBN has a more complicated energy function, the partition function is a constant at 1. In contrast, the RBM log partition function is intractable to
calculate for realistic problem sizes, and is estimated through Annealed Importance Sampling (AIS) (Salakhutdinov and Murray, 2008).

We first focus specifically on the global bounds for $W$ when perturbing it by an amount $U$. A bound for the RBM using the Shatten-$\infty$ norm was given by (Carlson et al., 2015b), which was derived by viewing the RBM objective as a difference-of-convex-functions problem. Their bound was

$$F(W + U) \leq F(W) + \text{tr}(\nabla W F(W) U^T) + \frac{M_J}{2} ||U||^{2}_{S^\infty}$$

(3.6)

Here we use our new scheme to generate a bound on the SBN problem. Given Theorem 4, in the SBN this reduces to bounding the first order Taylor approximation over $W$ on $v^T Wh - \sum_{m=1}^{M} \log(1 + \exp([c + Wh]_m))$. Although functions of this form are typically bound on the $\ell_2$ norm, we show in Supplemental Section B.4 that for this function the Shatten-$\infty$ norm gives a tighter bound. In the SBN, $W$ has the global bound

$$F(W + U) \leq F(W) + \text{tr}(\nabla W F(W) U^T) + \frac{J}{8} ||U||^{2}_{S^\infty}$$

(3.7)

Intriguingly, both the SBN (3.7) and RBM (3.6) have tighter bounds for $W$ on the Shatten-$\infty$ norm than the Frobenius norm. The minimizer of this majorization function is not in the direction of the gradient. Rather, the bound is minimized by taking the SVD of the gradient, $A \text{diag}(\lambda) B^T = \nabla W F(W)$, and setting $U = s||\lambda||_1 AB^T$, with $s$ set by the theory to $\frac{-4}{J}$ for the SBN and $\frac{-1}{M_J}$ for the RBM.

We bound the vector parameters $c$ and $b$ on the $\ell_2$ norm. This leads to standard gradient updates, and these majorization functions are both minimized by a stepsize of $-4$. Details can be found in Supplemental Section B.4.
Gradient Estimation for the SBN and Variational Approximations

Gradient estimation in the SBN is difficult. The gradient on $W$ is $-\nabla_W F(\theta) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{p(h|v)}[(v-D_n,\text{softmax}(c+Wh))h^T]$. This expectation is typically calculated by a Monte Carlo integration (MCI) scheme. A Gibbs sampling procedure was used in (Neal, 1992) to estimate the gradient. This is a computationally expensive operation, and the total time of this step is roughly $O(N_{\text{batch}}CMJ^2)$, where $N_{\text{batch}}$ is the number of data samples used in a mini-batch and $C$ is the number of Gibbs sweeps used in the estimation procedure. Typical settings for the gradient estimation in maximum likelihood will cause the computational time for the gradient to dominate the cost of the SVD necessary for the minimization step.

Previous work on the SBN estimated the model likelihood by the harmonic mean estimator (Gan et al., 2015). The harmonic mean estimator is quick and easy to implement, but unfortunately unreliable. We developed an AIS scheme to evaluate the model likelihood, and use this in the experiments.

Instead of using posterior sampling, variational methods (Frey and Hinton, 1999) are often used, putting a simple, tractable form on $q(h) = \prod_j \text{Bern}(h_j; \pi_j)$ and minimizing $\pi = \arg \min_{\pi} \text{KL}(q(h)||p(h|v))$ in order to get simple estimation procedures. In this case, a simpler MCI scheme is used to estimate the gradient by drawing samples from $q(h)$. This can be used with mini-batches to estimate parameters in the Stochastic Variational Inference scheme (Hoffman et al., 2013).

Deep Extensions

The Deep RBM (Salakhutdinov and Hinton, 2009a) and the Deep SBN (Gregor et al., 2014) are extensions to the RBM and SBN, respectively. In a three-layer model, they have visible nodes $v \in \{0,1\}^M$, $h^{(1)} \in \{0,1\}^{J^{(1)}}$, and $h \in \{0,1\}^{J^{(2)}}$, with parameters
\{c, b^{(1)}, b^{(2)}, W^{(1)}, W^{(2)}\}. The energy function for the DRBM is
\[
-E_{\theta}(v, h^{(1)}, h^{(2)}) = v^T c + v^T W^{(1)} h^{(1)} + (h^{(1)})^T b^{(1)} + h^{(1)} W^{(2)} h^{(2)} + (h^{(2)})^T b^{(2)}
\]
and the generative model for the DSBN is
\[
p_{\theta}(h^{(2)}) = \prod_{j=1}^{J^{(2)}} \text{Bern}(h^{(2)}_j; \sigma(b^{(2)}_j))
p_{\theta}(v|h^{(2)}) = \prod_{j=1}^{J^{(1)}} \text{Bern}(h^{(1)}_j; \sigma([b^{(1)} + W^{(2)} h^{(2)}]_j))
p_{\theta}(v|h^{(2)}) = \prod_{m=1}^{M} \text{Bern}(v_m; \sigma([c + W^{(1)} h^{(1)}]_m))
\]
Both the DSBN and the DRBM have Shatten-\infty bounds on the parameters \(W^{(1)}\) and \(W^{(2)}\). The minimizers of the majorization functions for these parameters are the same as (3.6) and (3.7) for \(W^{(1)}\), and for \(W^{(2)}\) the stepsize is \(-\frac{4}{J^{(2)}}\) for the DSBN and \(-\frac{1}{J^{(1)} J^{(2)}}\) for the DRBM. These patterns continue for deeper models.

### 3.3.2 Topic Modeling: Replicated Softmax Models

For topic modeling problems, each document \(n\) is represented by a vector of counts \(v_n \in Z^M\), where each unit \(v_{nm}\) represents the number of times the unique dictionary word \(m\) appears in document \(n\). The total number of words in the document is \(D_n = ||v_n||_1\). This bag-of-words assumption is common in probabilistic topic modeling problems (Blei et al., 2003).

An undirected topic model called the Replicated Softmax-RBM (RS-RBM) was proposed in (Salakhutdinov and Hinton, 2009b), and a directed topic model, which we will refer to as the Replicated Softmax Belief Network (RSBN) was proposed in (Mnih and Gregor, 2014). Both use this vector representation of the observations and binary hidden nodes \(b \in \{0, 1\}^J\). Both models are parameterized by \(\theta = \{c, W, b\}\), and both have a multinomial distribution given the hidden units, with
\[
v_n|h_n = \text{Multi}(D_n; \text{softmax}(c + Wh))
\]
Like in the binary models, the difference in the undirected and directed models is the relationship to the hidden units,

RSBN: \[ p_\theta(h) = \prod_{j=1}^J \text{Bern}(h_j; \sigma(b_j)) \]

RS-RBM: \[ p_\theta(h | v) = \prod_{j=1}^J \text{Bern}(h_j; \sigma([Db + W^T v]_j)) \]

This undirected versus directed relationship leads to related energy functions, with

RSBN: \[-E_\theta(v, h) = v^T c + v^T Wh + h^T b \]

\[ -D \log \sum_{m=1}^M \exp([c + Wh]_m)) - \sum_{j=1}^J \log(1 + \exp(b_j)) \]

RS-RBM: \[-E_\theta(v, h) = v^T c + v^T Wh + Dh^T b \]

We first examine the relationship for \( W \) versus a perturbed version \( W + U \). For the RS-RBM, the energy function is linear and convex, so we use (3.2) and (3.4). The bound depends on the final term in (3.4) over the gradient \(-\nabla_w E(v, h) = v^T h^T\), which is simplified

\[
\frac{1}{2} \max_{e \in \mathbb{Z}^H, |v| \leq D} \max_{h \in \{0,1\}^J} \text{tr}(vh^TU^T)^2 \leq \frac{D^2}{2} \max_{m=1,...,M} ||U_m||_1^2
\]

The relationship is bound by the maximum \( \ell_1 \)-norm of the perturbation in a row. Extending this to a collection of documents, a global bound is

\[
F(W + U) \leq F(W) + \text{tr}(\nabla_w F(W) U^T) + \frac{D^2}{2} \max_{m=1,...,M} ||U_m||_1^2
\]

(3.8)

where \( D^2 \) represents average squared number of words per document. An alternative bound on the Shatten-\( \infty \) norm is

\[
F(W + U) \leq F(W) + \text{tr}(\nabla_w F(W) U^T) + \frac{D^2}{2} ||U||_S^2
\]

(3.9)
Derivations of (3.8) and (3.9) are in Supplemental Section B.5.

The derivation for the RSBN depends on the error in the first order Taylor approximation of the concave negative energy function. Despite a different derivation, the RSBN has the same bounds as the RS-RBM expressed in (3.8) and (3.9). The derivation of (3.8) and (3.9) for the RSBN are in Supplemental Section B.6.

Unlike the assumption in gradient descent methods, the minimizer of (3.8) is not in the direction of the gradient. Letting $A = \nabla_w F(W)$, the minimizer of Equation 3.8 will be given by

$$\tilde{U}_{mj} = \left\{ \begin{array}{ll} s \sum m ||A_{mj}||_{\infty} \text{sign}(A_{mj}), & \text{if } |A_{mj}| = \max_j |A_{mj}| \\ 0, & \text{otherwise} \end{array} \right.$$ 

with $s = \frac{1}{D^2}$ for both the RSBN and the RS-RBM.

We note that similar to the binary models in Section 3.3.1, the bound for these problems is tighter on the Shatten-$\infty$ norm compared to the Frobenius norm. This motivates using methods that work in the Shatten-$\infty$ norm as well as the max $\ell_1$ row norm. This is a trade-off because the max $\ell_1$ row is parallelizable and the nonlinear operations on the gradient are calculable in linear time.

Both the RS-RBM and RSBN are bound on the $\ell_\infty$ norm for $c$ and the $\ell_2$ norm on $b$. The optimal step for the $\ell_\infty$ norm uses only the $\ell_1$ norm and the sign of the gradient.

**Gradient Estimation, Variational Methods, and Function Evaluations**

The gradients are known for the RSBN (Mnih and Gregor, 2014) and the RSRBM (Salakhutdinov and Hinton, 2009b). For the RSRBM, the gradients are estimated with CD sampling methods, and the function is estimated with AIS as in (Salakhutdinov and Hinton, 2009b).

Similarly, a Gibbs sampler and AIS are developed for the RSBN. (Mnih and
Gregor, 2014) also used a technique called a recognition model to do posterior inference and estimate a lower bound on the model likelihood. Standard variational techniques like the approximation in (Frey and Hinton, 1999) are also utilized in the experiments.

**Deep Extensions**

A deeper version of an autoregressive version of RSBN, called fDARN, was shown in (Gregor et al., 2014; Mnih and Gregor, 2014). Likewise, a deeper version of the RS-RBM, called the Over-Replicated Softmax-RBM, was introduced in (Srivastava et al., 2013). Instead of these models, a deep RSBN and a deep RS-RBM could also be defined by using the same relationship for hidden layers $h^{(1)}$ and $h^{(2)}$ shown in Section 3.3.1. In these models, $W^{(1)}$ has the bound given in (3.8) or (3.9), while the deeper layers would have the same global bounds and steps as the DSBN and the DRBM for $W^{(l)}$ for $l > 1$.

### 3.3.3 Deep Belief Nets

Deep Belief Nets (DBN) (Hinton et al., 2006) are deep graphical models that have both undirected and directed edges. For a 3-layer DBN, the first hidden layer $h^{(1)} \in \{0, 1\}^{J^{(1)}}$ and the second hidden layer $h^{(2)} \in \{0, 1\}^{J^{(2)}}$ are jointly drawn from an undirected RBM model with parameters $\{b^{(1)}, W^{(2)}, b^{(2)}\}$, and the visible units $v \in \{0, 1\}^M$ are drawn from the directed model with $v_m| h^{(1)} \sim \text{Bern}(v_m; \sigma([c + W^{(1)}h^{(1)}]_m))$. The energy function for this model is written as

$$-E_\theta(h, v^{(1)}, v^{(2)}) = v^T c + v^T W^{(1)} h^{(1)}$$

$$- \sum_{m=1}^{M} \log(1 + \exp([c + W^{(1)}h^{(1)}]_m))$$

$$+ (h^{(1)})^T b^{(1)} + (h^{(1)})^T W^{(2)} h^{(2)} + (h^{(2)})^T b^{(2)}$$

56
The exact log partition function is intractable for realistic size problems, but there are joint AIS and variational lower bound techniques to approximate the model likelihood (Salakhutdinov and Murray, 2008).

The theories that we have developed allow us to create global bounds on the DBN, and the DBN follows the bounds developed for the SBN and RBM cases. Specifically, the bound on $W^{(1)}$ is dependent on the Shatten-$\infty$ norm with $\frac{J^{(1)}}{8}||U^{(1)}||^2_{SW}$, and the bound on $W^{(2)}$ is dependent on the Shatten-$\infty$ norm with $\frac{J^{(1)}J^{(2)}}{2}||U^{(2)}||^2_{SW}$.

The DBN can also be used for topic modeling by replacing the binary $\mathbf{v}$ with the replicated softmax. In this case, the bound on $W^{(1)}$ is replaced by RSBN bound given in (3.8) or (3.9). The details on why these bounds hold are shown in Supplemental Section B.7.

3.4 Experiments

For the learning problems we consider here, there are many degrees of freedom. To standardize the results, all results use a minibatch of size 100. The learning rate decays with the number of iterations, $s^{-5}$. The only settings that vary are the geometry used in the descent algorithm and the stepsizes, as well as whether a (penalized)
maximum likelihood or variational lower bound was used. For the geometries, we denote the Stochastic Gradient Descent algorithm as SGD, the Stochastic Spectral Descent algorithm, which minimizes the Shatten-$\infty$ norm, as SSD, the stochastic method that minimizes the maximum $\ell_1$ row as L1R. Comparisons are given to the RMSprop algorithm (Tieleman and LeCun, 2012). ADAgrad and RMSprop gave similar results, so only RMSprop is shown for simplicity.

**Figure 3.2:** Results on RSBN (Left and Middle) training reporting the perplexity metric. (Left) Synthetic data with $J = 15$ using exact gradients and log-likelihood measurements (Middle) $J = 50$, using variational approximation to obtain gradients and log-likelihood. (Right) Results on RS-RBM for a synthetic dataset of size $J = 25$

**Figure 3.3:** (Left) Learning curves for 20 Newsgroups with the RS-RBM. (Middle) Learning curves for 20 newsgroups with the RSBN (Right) Learning curves for Reuters with the RSBN
3.4.1 *Sigmoid Belief Nets*

We show results on two datasets for the SBN. First, we use a small, synthetic dataset to show the properties of the learning. Second, we show the learning curves for the MNIST digits dataset. For the MNIST dataset, the images were binarized as described in (Salakhutdinov and Murray, 2008). We show the learning curves for SGD and SSD, as well as their variational counterparts, which are denoted SVI and Spectral SVI (SSVI), respectively. Gibbs sweeps were used to estimate the gradients for the penalized maximum likelihood setup. For AIS, 100 samples per image were used to evaluate the model likelihood.

For the synthetic dataset, we set the dimensionality of the observations, \( M \), to 250, and the number of hidden nodes, \( J \), to 25. The parameters were drawn from Gaussians for the matrix \( W \) with \( c = 0 \) and \( b = 0 \). 1000 data samples were generated from the true model. Learning proceeded by using either Gibbs sampling or a variational method to approximate the posterior, and the algorithms learned only the parameter \( W \). The number of iterations is a fair comparison for the SSD because the SVD was less than 2% of the per-iteration cost. Regularization parameters followed the standard \( \ell_2 \) penalty at \( 10^{-4} \). We set the stepsizes to start at the rate suggested by the theory, at \(-4/J\) for both SSD and SGD. Next, to see the sensitivity of the algorithms with respect to the step size, we tuned the algorithms by multiplying the step size by 2 until the learning rate decreased. In Figure 3.1(left) the mean log \( p(v_n) \) as a function of iterations is shown.

Interestingly, even the tuned SGD is outperformed by the SSD algorithm with the step size of \(-4/J\) prescribed in Section 3.3.1. A tuned SSD algorithm (in these cases, \(-16/J\)) improves the efficiency by over an order of magnitude compared to the SGD techniques. Similar trends are observed for both maximum likelihood and variational lower bound techniques. RMSprop offers a slight improvement over SGD.
In Figure 3.1 (middle and right), we show the learning curves on the MNIST dataset for networks of size $J = 25$ and $J = 100$. Again, the tuned SGD is outperformed by the untuned SSD, and tuned SSD dramatically improves performance. RMSprop offers a slight improvement over SGD. In the larger network size, the SGD gets stuck in worse local modes than SSD. While the focus of this paper is on the training of these models, the testing predictive log-likelihood we get for these models is state of the art for the model size. In the $J = 25$ case, the RBM has a testing log likelihood of -143.20 (Salakhutdinov and Murray, 2008) and the best known value for the SBN with a variational approximation was -138.34 (Gan et al., 2015). Our testing value is at -127.45, an improvement of over 10 nats. For a single-layer, the best known SBN is -113.1 (Mnih and Gregor, 2014) with a variational approximation and -94.70 with a the harmonic mean estimator. The harmonic mean estimator is known to overestimate performance and have infinite variance (Wallach et al., 2009). Our unbiased AIS method estimates a test performance of -96.73 on a network of size 100, which also improves the -97.11 for an RBM of this size (Carlson et al., 2015b).

3.4.2 Topic Modeling

To demonstrate performance on the RSBN and the RS-RBM models, we compared the algorithms using geometry (L1R, SSD) with SGD and RMSprop. Before tackling a real-world dataset, we test our algorithms on a dataset of synthetically generated data. These data were created by substantiating a RSBN or RS-RBM with weights sampled from a normal distribution. Unit biases were set to zero. Samples were generated by setting the hidden layer to a random binary vector and then sampling an assignment for the visible layer, either by a simple forward pass in the case of the RSBN, or with Gibbs sampling for the RS-RBM. Using initial small random weights, we then proceeded to test the convergence, in terms of perplexity, of the various algorithms on problems of different scale. For these synthetic datasets, we
show the performance for the RSBN when a true gradient is used in Figure 3.2(left), and well as variational posterior approximations in Figure 3.2(middle and right). In these cases, SSD and L1R show similar performance, and show greatly improved performance over SGD and RMSprop. For the RS-RBM, we set the CD order to 3 and used 100 samples in AIS per document as in (Salakhutdinov and Hinton, 2009b). We show a synthetic dataset with 25 hidden nodes with RS-RBM in Figure 3.2(right). In the RSRBM, SSD outperforms all competing algorithms, and L1R is initially very good, but is caught eventually.

We compare results on the well-known 20-Newsgroups dataset\(^1\). We used the preprocessing of (Salakhutdinov and Hinton, 2009b), which reduced the vocabulary to \(M=2,000\) most frequent words in the corpus, split the corpus into a training set (11,284 documents) and evaluation set (7,502 documents). For the RSBN, the resulting convergence plots are shown in Figure 3.3 (middle) for \(J=100\). SSD is the clear winner, and both L1R and RMSprop significantly improve SGD. The RS-RBM training result is shown in Figure 3.3 (left) for \(J=50\). For the RS-RBM, SSD shows dominant performance. As well, using the AIS scheme on the testing dataset, the estimated mean test perplexity is 841.53, which is considerably lower than the reported perplexity for SGD training of 953 (Salakhutdinov and Hinton, 2009b). Unlike the RBM problem (Yuille, 2004), different local modes in the RS-RBM have different performance levels. Our SGD code achieved a level of 945 after 50,000 iterations.

We applied the RSBN to the Reuters Corpus. This was split into a 794,414 training documents and 10,000 testing document. The size of the corpus allowed an online optimization scheme where mini-batches are observed in an online fashion. The data were preprocessed to contain \(M = 10,000\) unique words. To show that the algorithms scale in an online setting, an RSBN with \(J = 100\) hidden units was

\(^1\) http://qwone.com/jason/20Newsgroups/
used. Figure 3.3 (right) shows the results of training this experiment. SSD shows the best performance, and converges well before the other algorithms. After 10,000 iterations, the hold-out perplexity is 1712 from the SSD algorithm, versus 1841 for RMSprop and 2190 for L1R. SGD was uncompetitive for this network size.

3.5 Discussion

We have introduced a general bounding technique for developing majorization-minimization schemes for discrete graphical models with hidden units. Their specific application to directed and undirected graphical models revealed the parallel in the geometry of the learning problem for similar graph structures, and stochastic descent algorithms based on these geometries provide greatly improved efficiency over other first-order methods. Additionally, in topic modeling problems, we show the similar performance between an algorithm that operates in the Shatten-$\infty$ norm and the max $\ell_1$ row norm, and the max $\ell_1$ row norm is based on easily parallelizable coordinate updates.
The Focused Mixture Model for Spike Sorting

4.1 Introduction

Spike sorting of extracellular electrophysiological data is an important problem in contemporary neuroscience, with applications ranging from brain-machine interfaces (Nicolelis and Lebedev, 2009) to neural coding (Rieke et al., 1997) and beyond. Despite a rich history of work in this area (Wheeler, 1991; Einevoll et al., 2012), room for improvement remains for automatic methods. In particular, we are interested in sorting spikes from multichannel longitudinal data, where longitudinal data potentially consists of many experiments conducted in the same animal over weeks or months.

Here we propose a Bayesian generative model and associated inference procedure. Perhaps the most important advance in our present work over previous art is our joint feature learning and clustering strategy. More specifically, standard pipelines for processing extracellular electrophysiology data consist of the following steps: (i) filter the raw sensor readings, (ii) perform thresholding to “detect” the spikes, (iii) map each detected spike to a feature vector, and then (iv) cluster the feature vectors
(Lewicki, 1998b). Our primary conceptual contribution to spike sorting methodologies is a novel unification of steps (iii) and (iv) that utilizes all available data in such a way as to satisfy all of the above criteria. This joint dictionary learning and clustering approach improves results even for a single channel and a single recording experiment (i.e., not longitudinal data). Additional localized recording channels improve the performance of our methodology by incorporating more information. More recordings allow us to track dynamics of firing over time.

Although a comprehensive survey of previous spike sorting methods is beyond the scope of this manuscript, below we provide a summary of previous work as relevant to the above listed goals. Perhaps those methods that are most similar to ours include a number of recent Bayesian methods for spike sorting (Gasthaus et al., 2009; Chen et al., 2011b). One can think of our method as a direct extension of theirs with a number of enhancements. Most importantly, we learn features for clustering, rather than simply using principal components. We also incorporate multiple electrodes, assume a more appropriate prior over the number of clusters, and address longitudinal data.

Other popular methods utilize principal components analysis (PCA) (Lewicki, 1998b) or wavelets (Letelier and Weber, 2000) to find low-dimensional representations of waveforms for subsequent clustering. These methods typically require some manual tuning, for example, to choose the number of retained principal components. Moreover, these methods do not naturally handle missing data well. Finally, these methods choose low-dimensional embeddings for reconstruction and are not necessarily appropriate for downstream clustering.

Calabrese et al. (Calabrese and Paniski, 2010) recently proposed a Mixture of Kalman Filters (MoK) model to explicitly deal with slow changes in waveform shape. This approach also models spike rate (and even refractory period), but it does not address our other desiderata, perhaps most importantly, utilizing multiple electrodes.
or longitudinal data. It would be interesting to extend that work to utilize learned time-varying dictionaries rather than principal components.

Finally, several recently proposed methods address sparsely firing neurons (Pedreira et al., 2012; Adamos et al., 2012). By directly incorporating firing rate into our model and inference algorithm (see Section 4.2.3), our approach outperforms previous methods even in the absence of manual tuning (see Section 4.3.5).

The remainder of the manuscript is organized as follows. Section 4.2 begins with a conceptual description of our model followed by mathematical details and experimental methods for new data. Section 4.3 begins by comparing the performance of our approach to several other previous state-of-the-art methods, and then highlights the utility of a number of additional features that our method includes. Section 4.4 summarizes and provides some potential future directions. The Appendix provides details of the relationships between our method and other related Bayesian models or methodologies.

4.2 Models and Analysis

4.2.1 Model Concept

Our generative model derives from knowledge of the properties of electrophysiology signals. Specifically, we assume that each waveform can be represented as a sparse superposition of several dictionary elements, or features. Rather than presupposing a particular form of those features (e.g., wavelets), we learn features from the data. Importantly, we learn these features for the specific task at hand: spike sorting (i.e., clustering). This is in contrast to other popular feature learning approaches, such as principal component analysis (PCA) or independent component analysis (ICA), which learn features to optimize a different objective function (for example, minimizing reconstruction error). Dictionary learning has been demonstrated as a powerful idea, with demonstrably good performance in a number of applications (Zhou et al.,
Moreover, statistical guarantees associated with such approaches are beginning to be understood (Spielman et al., 2012). Section 4.2.2 provides mathematical details for our Bayesian dictionary learning assumptions.

We jointly perform dictionary learning and clustering for analysis of multiple spikes. The generative model requires a prior on the number of clusters. Regardless of the number of putative spikes detected, the number of different single units one could conceivably discriminate from a single electrode is upper bounded due to the conductive properties of the tissue. Thus, it is undesirable to employ Bayesian nonparametric methods (Antoniak, 1974) that enable the number of clusters (each cluster associated with a single-unit event) to increase in an unbounded manner as the number of threshold crossings increases. We develop a new prior to address this issue, which we refer to as a “focused mixture model” (FMM). The proposed prior is also appropriate for chronic recordings, in which single units may appear for a subset of the recording days, but also disappear and reappear intermittently. Sections 4.2.3 and 4.2.4 provide mathematical details for the general mixture modeling case, and our specific focused mixture model assumptions.

We are also interested in multichannel recordings. When we have multiple channels that are within close proximity to one another, we can “borrow statistical strength” across the channels to improve clustering accuracy. Moreover, we can ascertain that certain movement or other artifacts – which would appear to be spikes if only observing a single channel – are clearly not spikes from a single neuron, as evidenced by the fact that they are observed simultaneously across all the channels, which is implausible for a single neuron. While it is possible that different neurons may fire simultaneously and be observed coincidently across multiple sensor channels, we have found that this type of observed data are more likely associated with animal motion, and artifacts from the recording setup (based on recorded video of the animal). We employ the multiple-channel analysis to distinguish single-neuron events.
from artifacts due to animal movement (inferred based on the electrophysiological data alone, without having to view all of the data).

Finally, we explicitly model the spike rate of each cluster. This can help address refractory issues, and perhaps more importantly, enables us to detect sparsely firing neurons with high accuracy.

Because our model is fully Bayesian, we can readily impute missing data. Moreover, by placing relatively diffuse but informed hyperpriors on our model, our approach does not require any manual tuning. And by reformulating our priors, we can derive (local) conjugacy which admits efficient Gibbs sampling. Section 4.2.5 provides details on these computations. In some settings a neuroscientist may want to tune some parameters, to tests hypotheses and impose prior knowledge about the experiment; we also show how this may be done in Section 4.3.4.

4.2.2 Bayesian dictionary learning

Consider electrophysiological data measured over a prescribed time interval. Specifically, let $X_{ij} \in \mathbb{R}^{T \times N}$ represent the $j^{th}$ signal observed during interval $i$ (each $j$ indexes a threshold crossing within a time interval $i$). The data are assumed recorded on each of $N$ channels, from an $N$-element sensor array, and there are $T$ time points associated with each detected spike waveform (the signals are aligned with respect to the peak energy of all the channels). In tetrode arrays (Emondi et al., 2004), and related devices like those considered below, a single-unit event (action potential of a neuron) may be recorded on multiple adjacent channels, and therefore it is of interest to process the $N$ signals associated with $X_{ij}$ jointly; the joint analysis of all $N$ signals is also useful for longitudinal analysis, discussed in Section 4.3.

To constitute data $X_{ij}$, we assume that threshold-based detection (or a related method) is performed on data measured from each of the $N$ sensor channels. When a signal is detected on any of the channels, coincident data are also extracted from all
$N$ channels, within a window of (discretized) length $T$ centered at the spikes’ energy peak average over all channels. On some of the channels data may be associated with a single-unit event, and on other channels the data may represent background noise. Both types of data (signal and noise) are modeled jointly, as discussed below.

Following (Chen et al., 2011b), we employ dictionary learning to model each $X_{ij}$; however, unlike (Chen et al., 2011b) we jointly employ dictionary learning to all $N$ channels in $X_{ij}$ (rather than separately to each of the channels). The data are represented

$$X_{ij} = D\Lambda S_{ij} + E_{ij}, \quad (4.1)$$

where $D \in \mathbb{R}^{T \times K}$ represents a dictionary with $K$ dictionary elements (columns), $\Lambda \in \mathbb{R}^{K \times K}$ is a diagonal matrix with sparse diagonal elements, $S_{ij} \in \mathbb{R}^{K \times N}$ represents the dictionary weights (factor scores), and $E_{ij} \in \mathbb{R}^{T \times N}$ represents residual/noise. Let $D = (d_1, \ldots, d_K)$ and $E = (e_1, \ldots, e_N)$, with $d_k, e_n \in \mathbb{R}^T$. We impose priors

$$d_k \sim \mathcal{N}(0, \frac{1}{T}I_T), \quad e_n \sim \mathcal{N}(0, \text{diag}(\eta_1^{-1}, \ldots, \eta_T^{-1})), \quad (4.2)$$

where $I_T$ is the $T \times T$ dimensional identity matrix and $\eta_t \in \mathbb{R}$ for all $t$.

We wish to impose that each column of $X_{ij}$ lives in a linear subspace, with dimension and composition to be inferred. The composition of the subspace is defined by a selected subset of the columns of $D$, and that subset is defined by the non-zero elements in the diagonal of $\Lambda = \text{diag}(\lambda)$, with $\lambda = (\lambda_1, \ldots, \lambda_K)^T$ and $\lambda_k \in \mathbb{R}$ for all $k$. We impose $\lambda_k \sim \nu \delta_0 + (1 - \nu)\mathcal{N}_+(0, \alpha_0^{-1})$, with $\nu \sim \text{Beta}(a_0, b_0)$ and $\delta_0$ a unit measure concentrated at zero. The hyperparameters $a_0, b_0 \in \mathbb{R}$ are set to encourage sparse $\lambda$, and $\mathcal{N}_+(\cdot)$ represents a normal distribution truncated to be non-negative. Diffuse gamma priors are placed on $\{\eta_t\}$ and $\alpha_0$.

Concerning the model priors, the assumption $d_k \sim \mathcal{N}(0, \frac{1}{T}I_T)$ is consistent with a conventional $\ell_2$ regularization on the dictionary elements. Similarly, the assumption
\( e_n \sim \mathcal{N}(0, \text{diag}(\eta_1^{-1}, \ldots, \eta_T^{-1})) \) corresponds to an \( \ell_2 \) fit of the data to the model, with a weighting on the norm as a function of the sample point (in time) of the signal.

We also considered using a more general noise model, with \( e_n \sim \mathcal{N}(0, \Sigma) \). These priors are typically employed in dictionary learning; see (Zhou et al., 2012b) for a discussion of the connection between such priors and optimization-based dictionary learning.

### 4.2.3 Mixture modeling

A mixture model is imposed for the dictionary weights \( S_{ij} = (s_{ij1}, \ldots, s_{ijN}) \), with \( s_{ijn} \in \mathbb{R}^K \); \( s_{ijn} \) defines the weights on the dictionary elements for the data associated with the \( n \)th channel (\( n \)th column) in \( X_{ij} \). Specifically,

\[
\begin{align*}
\mathbf{s}_{ijn} &\sim \mathcal{N}(\mathbf{\mu}_{z_{ijn}}, \mathbf{\Omega}_{z_{ijn}}^{-1}), \quad z_{ij} \sim \sum_{m=1}^{M} \pi_m^{(i)} \delta_m, \\
(\mathbf{\mu}_{mn}, \mathbf{\Omega}_{mn}) &\sim G_0(\mathbf{\mu}_0, \beta_0, W_0, \nu_0)
\end{align*}
\]

(4.3)

(4.4)

where \( G_0 \) is a normal-Wishart distribution with \( \mu_0 \) a \( K \) dimension vector of zeros, \( \beta_0 = 1 \), \( W_0 \) is a \( K \) dimensional identity matrix, and \( \nu_0 = K \). The other parameters: \( \pi_m^{(i)} > 0 \), \( \sum_{m=1}^{M} \pi_m^{(i)} = 1 \), and \( \{s_{ijn}\}_{n=1,N} \) are all associated with cluster \( z_{ij} \); \( z_{ij} \in \{1, \ldots, M\} \) is an indicator variable defining with which cluster \( X_{ij} \) is associated, and \( M \) is a user-specified upper bound on the total number of clusters possible.

The use of the Gaussian model in (4.3) is convenient, as it simplifies computational inference, and the normal-Wishart distribution \( G_0 \) is selected because it is the conjugate prior for a normal distribution. The key novelty we wish to address in this paper concerns design of the mixture probability vector \( \mathbf{\pi}^{(i)} = (\pi_1^{(i)}, \ldots, \pi_M^{(i)})^T \).

### 4.2.4 Focused Mixture Model

The vector \( \mathbf{\pi}^{(i)} \) defines the probability with which each of the \( M \) mixture components are employed for data recording interval \( i \). We wish to place a prior probability...
distribution on $\pi^{(i)}$, and to infer an associated posterior distribution based upon the observed data. Let $b_m^{(i)}$ be a binary variable indicating whether interval $i$ uses mixture component $m$. Let $\hat{\phi}_m^{(i)}$ correspond to the relative probability of including mixture component $m$ in interval $i$, which is related to the firing rate of the single-unit corresponding to this cluster during that interval. Given this, the probability of cluster $m$ in interval $i$ is

$$\pi_m^{(i)} = \frac{1}{Z} b_m^{(i)} \hat{\phi}_m^{(i)}$$

where $Z = \sum_{m=1}^{M} b_m^{(i)} \hat{\phi}_m^{(i)}$ is the normalizing constant to ensure that $\sum_m \pi_m^{(i)} = 1$. To finalize this parameterization, we further assume the following priors on $b_m^{(i)}$ and $\hat{\phi}_m^{(i)}$:

$$\hat{\phi}_m^{(i)} \sim \text{Ga} (\phi_m, p_i/(1-p_i)),$$

$$\phi_m \sim \text{Ga} (\gamma_0, 1), \quad p_i \sim \text{Beta}(a_0, b_0) \quad (4.6)$$

$$b_m^{(i)} \sim \text{Bern}(\nu_m),$$

$$\nu_m \sim \text{Beta}(\alpha/M, 1), \quad \gamma_0 \sim \text{Ga}(c_0, 1/d_0) \quad (4.7)$$

where $\text{Ga}(\cdot)$ denotes the gamma distribution, and $\text{Bern}(\cdot)$ the Bernoulli distribution. Note that $\{\phi_m, \nu_m\}_{m=1,M}$ are shared across all intervals $i$, and it is in this manner we achieve joint clustering across all time intervals. The reasons for the choices of these various priors is discussed in Section C.2, when making connections to related models. For example, the choice $b_m^{(i)} \sim \text{Bern}(\nu_m)$ with $\nu_m \sim \text{Beta}(\alpha/M, 1)$ is motivated by the connection to the Indian buffet process (Griffiths and Ghahramani, 2005) as $M \to \infty$.

We refer to this as a focused mixture model (FMM) because the $\nu_m$ defines the probability with which cluster $m$ is observed, and via the prior in (4.7) the model only “focuses” on a small number of clusters, those with large $\nu_m$. Further, as discussed below, the parameter $\phi_m$ controls the firing rate of neuron/cluster $m$, and that is
also modeled. Concerning models to which we compare, when the $\pi^{(i)}_m$ are modeled
via a Dirichlet process (DP) (Antoniak, 1974), and the matrix of multi-channel data
are modeled jointly, we refer to the model as matrix DP (MDP). If a DP is employed
separately on each channel the results are simply termed DP. The hierarchical DP
model in (Chen et al., 2011b) for $\pi^{(i)}_m$ the model is referred to as HDP.

4.2.5 Computations

The posterior distribution of model parameters is approximated via Gibbs sampling.
Most of the update equations for the model are relatively standard due to conjugacy
of consecutive distributions in the hierarchical model; these “standard” updates are
not repeated here (see (Chen et al., 2011b)). Perhaps the most important update
equation is for $\phi_m$, as we found this to be a critical component of the success of our
inference. To perform such sampling we utilize the following lemma.

Lemma 6. Denote $s(n, j)$ as the Sterling numbers of the first kind (Johnson et al.,
2005) and $F(n, j) = (-1)^{n+j}s(n, j)/n!$ as their normalized and unsigned representa-
tions, with $F(0,0) = 1$, $F(n,0) = 0$ if $n > 0$, $F(n,j) = 0$ if $j > n$ and
$F(n+1,j) = \frac{n}{n+1}F(n,j) + \frac{1}{n+1}F(n,j-1)$ if $1 \leq j \leq n$. Assuming $n \sim \text{NegBin}(\phi, p)$ is
a negative binomial distributed random variable, and it is augmented into a compound
Poisson representation (Anscombe, 1949) as

$$n = \sum_{\ell=1}^{\ell} u_\ell, \; u_\ell \sim \text{Log}(p), \; \ell \sim \text{Pois}(-\phi \ln(1-p))$$

(4.8)

where Log($p$) is the logarithmic distribution (Anscombe, 1949) with probability gener-
ating function $G(z) = \ln(1-pz)/\ln(1-p)$, $|z| < p^{-1}$, then we have

$$\Pr(\ell = j | n, \phi) = R_{\phi}(n, j) = F(n, j)\phi^j \bigg/ \sum_{j'=1}^{n} F(n, j')\phi^{j'}$$

(4.9)

for $j = 0, 1, \cdots, n$. 71
The proof is provided in the Appendix.

Let the total set of data measured during interval $i$ be represented $\mathcal{D}_i = \{X_{ij}\}_{j=1}^{M_i}$, where $M_i$ is the total number of events during interval $i$. Let $n_{im}^*$ represent the number of data samples in $\mathcal{D}_i$ that are apportioned to cluster $m \in \{1, \ldots, M\} = \mathcal{S}$, with $M_i = \sum_{m=1}^{M} n_{im}^*$. To sample $\phi_m$, since

$$p(\phi_m | p, n_{im}^*) \propto \prod_{i; b_{m}^{(i)}} \text{NegBin}(n_{im}^*; \phi_m, p_i) \text{Ga}(\phi_m; \gamma_0, 1)$$

(see Appendix C.2 for details), using Lemma 6, we can first sample a latent count variable $\ell_{im}$ for each $n_{im}^*$ as

$$\Pr(\ell_{im} = l | n_{im}^*, \phi_m) = R_{\phi_m}(n_{im}^*, l), \ l = 0, \ldots, n_{im}^*. \quad (4.10)$$

Since $\ell_{im} \sim \text{Pois}(-\phi_m \ln(1 - p_i))$, using the conjugacy between the gamma and Poisson distributions, we have

$$\phi_m | \{\ell_{im}, b_{m}^{(i)}, p_i\} \sim \text{Ga} \left( \gamma_0 + \sum_{i; b_{m}^{(i)} = 1} \ell_{im}, \frac{1}{1 - \sum_{i; b_{m}^{(i)} = 1} \ln(1 - p_i)} \right). \quad (4.11)$$

Notice that marginalizing out $\phi_m$ in $\ell_{im} \sim \text{Pois}(-\phi_m \ln(1 - p_i))$ results in $\ell_{im} \sim \text{NegBin}(\gamma_0, \frac{-\ln(1 - p_i)}{1 - \ln(1 - p_i)})$, therefore, we can use the same data augmentation technique by sampling a latent count $\tilde{\ell}_{im}$ for each $\ell_{im}$ and then sampling $\gamma_0$ using the gamma Poisson conjugacy as

$$\Pr(\tilde{\ell}_{im} = l | \ell_{im}, \gamma_0) = R_{\gamma_0}(\ell_{im}, l), \ l = 0, \ldots, \ell_{im}$$

$$\gamma_0 | \{\tilde{\ell}_{im}, b_{m}^{(i)}, p_i\} \sim \text{Ga} \left( \gamma_0 + \sum_{i; b_{m}^{(i)} = 1} \tilde{\ell}_{im}, \frac{1}{d_0 - \sum_{i; b_{m}^{(i)} = 1} \ln \left( 1 - \frac{-\ln(1 - p_i)}{1 - \ln(1 - p_i)} \right)} \right).$$

72
Another important parameter is $b_{m}^{(i)}$. Since $b_{m}^{(i)}$ can only be zero if $n_{im}^{*} = 0$ and when $n_{im}^{*} = 0$, $\Pr(b_{m}^{(i)} = 1|\pi_{m}, \phi_{m}, p_{i}) \text{NegBin}(0; \phi_{m}, p_{i}) \pi_{m}$ and $\Pr(b_{m}^{(i)} = 0|\pi_{m}) \propto (1 - \pi_{m})$, we have

$$b_{m}^{(i)}|\pi_{m}, n_{im}^{*}, \phi_{m}, p_{i} \sim \text{Bernoulli}\left(\delta(n_{im}^{*} = 0) \frac{\pi_{m}(1-p_{i})^{\phi_{m}}}{\pi_{m}(1-p_{i})^{\phi_{m}} + (1-\pi_{m})} + \delta(n_{im}^{*} > 0)\right).$$

A large $p_{i}$ thus indicates a large variance-to-mean ratio on $n_{im}^{*}$ and $M_{i}$. Note that when $b_{m}^{(i)} = 0$, the observed zero count $n_{im}^{*} = 0$ is no longer explained by $n_{im}^{*} \sim \text{NegBin}(r_{m}, p_{i})$, this satisfies the intuition that the underlying beta-Bernoulli process is governing whether a cluster would be used or not, and once it is activated, it is $r_{m}$ and $p_{i}$ that control how much it would be used.

### 4.2.6 Data Acquisition and Pre-processing

In this work we use two datasets, the popular “hc-1” dataset\(^1\) and a new dataset based upon experiments we have performed with freely moving rats (institutional review board approvals were obtained). These data will be made available to the research community. Six animals were used in this study. Each animal was trained, under food restriction (15 g/animal/day, standard hard chow), on a simple lever-press-and-hold task until performance stabilized and then taken in for surgery. Each animal was implanted with four different silicon microelectrodes (NeuroNexus Technologies; Ann Arbor, MI; custom design) in the forelimb region of the primary or supplementary motor cortex. Each electrode contains up to 16 independent recording sites, with variations in device footprint and recording site position (e.g., Figure 4.3(a)). Electrophysiological data were measured during one-hour periods on eight consecutive days, starting on the day after implant (data were collected for additional days, but the signal quality degraded after 8 days, as discussed below). The record-

\(^1\) available from http://crcns.org/data-sets/hc/hc-1
ings were conducted in a high walled observation chamber under freely-behaving conditions. Note that nearby sensors are close enough to record the signal of a single or small group of neurons, termed a single-unit event. However, in the device in Figure 4.3(a), all eight sensors in a line are too far separated to simultaneously record a single-unit event on all eight.

The data were bandpass filtered (0.3-3 kHz), and then all signals 3.5 times the standard deviation of the background signal were deemed detections. The peak of the detection was placed in the center of a 1.3 msec window, which corresponds to \( T = 40 \) samples at the recording rate. The signal \( X_{ij} \in \mathbb{R}^{T \times N} \) corresponds to the data measured simultaneously across all \( N \) channels within this window. Here \( N = 8 \), with a concentration on the data measured from the 8 channels of the zoomed-in Figure 4.3(a).

4.2.7 Evaluation Criteria

We use several different criteria to evaluate the performance of the competing methodologies. Let \( F_p \) and \( F_n \) denote the total number of false positives and negatives for a given neuron, respectively, and let \( \#_w \) denote the total number of detected waveforms. We define:

\[
\text{Accuracy} = \left\{ 1 - \frac{F_p + F_n}{\#_w} \right\} \times 100\%.
\]

(4.13)

For synthetic missing data, as in Section 4.3.3, we compute the relative recovery error (RRE):

\[
\text{RRE} = \frac{||1||X - \hat{X}}{||1||X} \times 100\%,
\]

(4.14)

where \( X \) is the true waveform, \( \hat{X} \) is the estimated waveform, and \( ||1|| \) indicates the \( L_2 \) or Frobenius norm depending on context. When adding noise, we compute the
signal-to-noise ratio (SNR) as in (Suner et al., 2005):

$$\text{SNR} = \frac{A}{2SD_{\text{noise}}}, \quad (4.15)$$

where $A$ denotes the peak-to-peak voltage difference of the mean waveform and $SD_{\text{noise}}$ is the standard deviation of the noise. The noise level is estimated by mean absolute deviation.

To simulate a lower SNR in the sparse spiking experiments, we took background signals from the dataset where no spiking occurred and scale them by $\alpha$ and add them to our detected spikes; this gives a total noise variance of $\sigma^2(1 + \alpha^2)$, and we set the SNR to 2.5 and 1.5 for these experiments.

### 4.3 Results

For these experiments we used a truncation level of $K = 40$ dictionary elements, and the number of mixture components was truncated to $M = 20$ (these truncation levels are upper bounds, and within the analysis a subset of the possible dictionary elements and mixture components are utilized). In dictionary learning, the gamma priors for $\{\eta_k\}$ and $\alpha_0$ were set as $\text{Ga}(10^{-6}, 10^{-6})$. In the context of the focused mixture model, we set $a_0 = b_0 = 1$, $c_0 = 0.1$ and $d_0 = 0.1$. Prior $\text{Ga}(10^{-6}, 10^{-6})$ was placed on parameter $\alpha$ related to the Indian Buffet Process (see Appendix C.2 for details). None of these parameters have been tuned, and many related settings yield similar results. In all examples we ran 6,000 Gibbs samples, with the first 3,000 discarded as burn-in (however, typically high-quality results are inferred with far fewer samples, offering the potential for computational acceleration).

#### 4.3.1 Real data with partial ground truth

We first consider publicly available dataset hc-1. These data consist of both extracellular recordings and an intracellular recording from a nearby neuron in the
hippocampus of an anesthetized rat (Henze et al., 2000). Intracellular recordings give clean signals on a spike train from a specific neuron, providing accurate spike times for that neuron. Thus, if we detect a spike in a nearby extracellular recording within a close time period (< 0.5ms) to an intracellular spike, we assume that the spike detected in the extracellular recording corresponds to the known neuron’s spikes.

We considered the widely used data d533101 and the same preprocessing from (Calabrese and Paniski, 2010). These data consist of 4-channel extracellular recordings and 1-channel intracellular recording. We used 2491 detected spikes and 786 of those spikes came from the known neuron. Accuracy of cluster results based on multiple methods are shown in Figure 4.1. We consider several different clustering schemes and two different strategies for learning low-dimensional representations of the data. Specifically, we learn low-dimensional representations using either: dictionary learning (DL) or the first two principal components (PCs) of the matrix consisting of the concatenated waveforms. For the multichannel data, we stack each waveform matrix to yield a vector, and concatenate stacked waveforms to obtain the data matrix upon which PCA is run. Given this representation, we consider several different clustering strategies: (i) Matrix Dirichlet Process (MDP), which implements a DP on the $X_{ij}$ matrices, as opposed to previous DP approaches on vectors (Gasthaus et al., 2009; Chen et al., 2011b), (ii) focused mixture model (as described above), (iii) Hierarchical DP (HDP) (Chen et al., 2011b), (iv) independent DP (the HDP and independent DP are from (Chen et al., 2011b)), (v) Mixture of Kalman filters (MoK) (Calabrese and Paniski, 2010), (vi) Gaussian mixture models (GMM) (Bishop., 2006), and (vii) K-means (KMEANS) (Lewicki, 1998b). Although we do not consider all pairwise comparisons, we do consider many options. Note that all of the DL approaches are novel. It should be clear from Figure 4.1 that dictionary learning enhances performance over principal components for each clustering approach. Specifically, all DL
based methods outperform all PC based methods. Moreover, sharing information across channels, as in MDP and FMM (both novel methodologies), seems to further improve performance. The ordering of the algorithms is essentially unchanged upon using a different number of mixture components or a different number of principal components.

In Figure 4.2, we visualize the waveforms in the first 2 principle components for comparison. In Figure 4.2a, we show ground truth to compare to the results we get by clustering from the K-means algorithm shown in Figure 4.2b and the clustering from the GMM shown in Figure 4.2c. We observe that both K-means and GMM work well, but due to the constrained feature space they incorrectly classify some spikes (marked by arrows). However, the proposed model, shown in Figure 4.2(d), which incorporates dictionary learning with spike sorting, infers an appropriate feature space (not shown) and more effectively clusters the neurons.

Note that in Figure 4.1 and 4.2, in the context of PCA features, we considered the two principal components (similar results were obtained with the three principal components); when we considered the 20 principal components, for comparison, the results deteriorated, presumably because the higher-order components correspond to noise. An advantage of the proposed approach is that we model the noise explicitly, via the residual $E_{ij}$ in (4.1); with PCA the signal and noise are not explicitly distinguished.

### 4.3.2 Longitudinal analysis of electrophysiological data

Figure 4.3(b)(a) shows the recording probe used for the analysis of the rat motor cortex data. Figure 4.3(b) shows assignments of data to each of the possible clusters, for data measured across the 8 days, as computed by the proposed model (for example, for the first three days, two clusters were inferred). Results are shown for the maximum-likelihood collection sample. As a comparison to FMM-DL, we also
Figure 4.1: Accuracy of the various methods on d533101 data (Henze et al., 2000). All abbreviations are explained in the main text (Section 4.3.1). Note that dictionary learning dominates performance over principal components. Moreover, modeling multiple channels (as in MDP and FMM) dominates performance over operating on each channel separately.

considered the non-focused mixture model (NFMM-DL) methodology discussed in Section C.2, with the $b^{(i)}$ set to all ones (in both cases we employ the same form of dictionary learning, as in Section 4.2.2). From Figure 4.3(c), it is observed that on held-out data the FMM-DL yields improved results relative to the NFMM-DL.

In fact, the proposed model was developed specifically to address the problem of multi-day longitudinal analysis of electrophysiological data, as a consequence of observed limitations of HDP (which are only partially illuminated by Figure 4.3(c)). Specifically, while the focused nature of the FMM-DL allows learning of specialized clusters that occur over limited days, the “non-focused” HDP-DL tends to merge similar but distinct clusters. This yields HDP results that are characterized by fewer total clusters, and by cluster characteristics that are less revealing of detailed neural processes. Patterns of observed neural activity may shift over a period of days due to many reasons, including cell death, tissue encapsulation, or device movement; this shift necessitates the FMM-DL’s ability to focus on subtle but important differences
Figure 4.2: Clustering results shown in the 2 PC space of the various methods on d533101 data (Henze et al., 2000). All abbreviations are explained in the main text (Section 4.3.1). “Known neuron” denotes waveforms associated with the neuron from the cell with the intracellular recording, and “Unknown neuron” refers to all other detected waveforms. Note that all methods are shown in the first two PCs for visualization, but that the FMM-DL shown in (d) is jointly learning the feature space and clustering.

in the data properties over days. This ability to infer subtly different clusters is related to the focused topic model’s ability (Williamson et al., 2010) to discern distinct topics that differ in subtle ways. The study of large quantities of data (8 days) makes the ability to distinguish subtle differences in clusters more challenging (the DP-DL-based model works well when observing data from one recording session, like in Figure 4.1, but the analysis of multiple days of data is challenging for HDP).

Note from Figure 4.3(b) that the number of detected signals is different for different recording days, despite the fact that the recording period reflective of these
Figure 4.3: Longitudinal data analysis of the rat motor cortex data. (a) Schematic of the neural recording array that was placed in the rat motor cortex. The red numbers identify the sensors, and a zoom-in of the bottom-eight sensors is shown. The sensors are ordered by the order of the read-out pads, at left. The presented data are for sensors numbered 1 to 8, corresponding to the zoomed-in region. (b) From the maximum-likelihood collection sample, the apportionment of data among mixture components (clusters). Results are shown for 45 sec recording periods, on each of 8 days. For example, D-4 reflects data on day 4. Note that while the truncation level is such that there are 20 candidate clusters (vertical axis in (b)), only an inferred subset of clusters are actually used on any given day. (c) Predictive likelihood of held-out data. The horizontal axis represents the fraction of data held out during training. FMM-DL dominates NFMM-DL on these data.

Figure 4.4: Posteriors and dictionaries from rat motor cortex data (the same data as in Figure 4.3). (a) Approximate posterior distribution on the number of global clusters (mixture components). (b) Approximate posterior distribution of the number of dictionary elements. (c) Examples of inferred dictionary elements; amplitudes of dictionary elements are unit less.
data (45 secs) is the same for all days. This highlights the need to allow modeling of different firing rates, as in our model but not emphasized in these results.

Among the parameters inferred by the model are approximate posterior distributions on the number of clusters across all days, and on the required number of dictionary elements. These approximate posteriors are shown in Figures 4.4(a) and 4.4(b), and Figure 4.4(c) shows example dictionary elements. Although not shown for brevity, the \{p_i\} had posterior means in excess of 0.9.

To better represent insight that is garnered from the model, Figure 4.5 depicts the inferred properties of three of the clusters, from Day 4 (D-4 in Figure 4.3(b)). Shown are the mean signal for the 8 channels in the respective cluster (for the 8 channels at the bottom of Figure 4.3(a)), and the error bars represent one standard deviation, as defined by the estimated posterior. Note that the cluster in the top row of Figure 4.5 corresponds to a localized single-unit event, presumably from a neuron (or a coordinated small group of neurons) near the sensors associated with channels 7 and 8. The cluster in the middle row of Figure 4.5 similarly corresponds to a single-unit event situated near the sensors associated with channels 3 and 6. Note the proximity of sensors 7 and 8, and sensors 3 and 6, from Figure 4.3(a). The HDP model uncovered the cluster in the top row of Figure 4.5, but not that in the middle row of Figure 4.5 (not shown).

Note the bottom row of Figure 4.5, in which the mean signal across all 8 channels is approximately the same (HDP also found related clusters of this type). This cluster is deemed to not be associated with a single-unit event, as the sensors are too physically distant across the array for the signal to be observed simultaneously on all sensors from a single neuron. This class of signals is deemed associated with an artifact or some global phenomena, (possibly) due to movement of the device within the brain, and/or because of charges that build up in the device and manifest signals with animal motion (by examining separate video recordings, such electrophysiological-
ical data occurred when the animal constituted significant and abrupt movement, such as heading hitting the sides of the cage, or during grooming). Note that in the top two rows of Figure 4.5 the error bars are relatively tight with respect to the strong signals in the set of eight, while the error bars in Figure 4.5(c) are more pronounced (the mean curves look smooth, but this is based upon averaging thousands of signals).

![Graph of electrophysiological data](image)

**Figure 4.5**: Example clusters inferred for data on the bottom 8 channels of Fig. 4.3(a). (a)-(b) Example of single-unit events. (c) Example of a cluster not attributed to a single-unit-event. The 8 signals are ordered from left to right consistent with the numbering of the 8 channels at the bottom of Figure 4.3(a). The black curves represent the mean, and the error bars are one standard deviation.

In addition to recording the electrophysiological data, video was recorded of the rat throughout the experiment. Robust PCA (Wright et al., 2009) was used to quantify the change in the video from frame-to-frame, with high change associated
with large motion by the animal (this automation is useful because one hour of data are collected on each day; direct human viewing is tedious and unnecessary). On Day 4, the model infers that in periods of high animal activity, 20% to 40% of the detected signals are due to single-unit events (depending on which portion of data are considered); during periods of relative rest 40% to 70% of detected signals are due to single-unit events. This suggests that animal motion causes signal artifacts, as discussed in Section 4.1.

In these studies the total fraction of single-unit events, even when at rest, diminishes with increasing number of days from sensor implant; this may be reflective of changes in the system due to the glial immune response of the brain (Biran et al., 2005; Szarowski et al., 2003). The discerning ability of the proposed FMM-DL to distinguish subtly different signals, and analysis of data over multiple days, has played an important role in this analysis. Further, longitudinal analyses like that in Figure 4.5 were the principal reason for modeling the data on all \( N = 8 \) channels jointly (the ability to distinguish single-unit events from anomalies is predicated on this multi-channel analysis).

4.3.3 Handling missing data

The quantity of data acquired by a neural recording system is enormous, and therefore in many systems one first performs spike detection (for example, based on a threshold), and then a signal is extracted about each detection (a temporal window is placed around the peak of a given detection). This step is often imperfect, and significant portions of many of the spikes may be missing due to the windowed signal extraction (and the missing data are not retainable, as the original data are discarded). Conventional feature-extraction methods typically cannot be applied to such temporally clipped signals.

Returning to (4.1), this implies that some columns of the data \( X_{ij} \) may have
missing entries. Conditioned on $\mathbf{D}$, $\mathbf{A}$, $\mathbf{S}_{ij}$, and $(\eta_1, \ldots, \eta_T)$, we have

$$X_{ij} \sim \mathcal{N}(\mathbf{D}\mathbf{A}\mathbf{S}_{ij}, \text{diag}(\eta_1^{-1}, \ldots, \eta_T^{-1}))$$

The missing entries of $X_{ij}$ may be treated as random variables, and they are integrated out analytically within the Gaussian likelihood function. Therefore, for the case of missing data in $X_{ij}$, we simply evaluate (4.1) at the points of $X_{ij}$ for which data are observed. The columns of the dictionary $\mathbf{D}$ of course have support over the entire signal, and therefore given the inferred $\mathbf{S}_{ij}$ (in the presence of missing data), one may impute the missing components of $X_{ij}$ via $\mathbf{D}\mathbf{A}\mathbf{S}_{ij}$. As long as, across all $X_{ij}$, the same part of the signal is not clipped away (lost) for all observed spikes, by jointly processing all of the retained data (all spikes) we may infer $\mathbf{D}$, and hence infer missing data.

In practice we are less interested in observing the imputed missing parts of $X_{ij}$ than we are in simply clustering the data, in the presence of missing data. By evaluating $X_{ij} \sim \mathcal{N}(\mathbf{D}\mathbf{A}\mathbf{S}_{ij}, \text{diag}(\eta_1^{-1}, \ldots, \eta_T^{-1}))$ only at points for which data are observed, and via the mixture model in (4.4), we directly infer the desired clustering, in the presence of missing data (even if we are not explicitly interested in subsequently examining the imputed values of the missing data).

To examine the ability of the model to perform clustering in the presence of missing data, we reconsider the publicly available data from Section 4.3.1. For the first 10% of the spike signals (300 spike waveforms), we impose that a fraction of the beginning and end of the spike is absent. The original signals are of length $T = 40$ samples. As a demonstration, for the “clipped” signals, the first 10 and the last 16 samples of the signals are missing. A clipped waveform example is shown in Figure 4.6(a); we compare the mean estimation of the signal, and the error bars reflect one standard deviation from the full posterior on the signal. In the context of the analysis,
we processed all of the data as before, but now with these “damaged”/clipped signals.

We observed that 94.11% of the non-damaged signals were clustered properly (for
the one neuron for which we had truth), and 92.33% of the damaged signals were
sorted properly. The recovered signal in Figure 4.6(a) is typical, and is meant to give
a sense of the accuracy of the recovered missing signal. The ability of the model to
perform spike sorting in the presence of substantial missing data is a key attribute
of the dictionary-learning-based framework.

![Figure 4.6](image)

**Figure 4.6:** Our generative model elegantly addresses missing data. (a) Example
of a clipped waveform from the publicly available data (blue), original waveform
(gray) and recovery waveform (black); the error bars reflect one standard deviation
from the posterior distribution on the underlying signal. (b) Relative errors (with
respect to the mean estimated signal). Note that we only show part of the waveform
for visualization purposes.

### 4.3.4 Model tuning

As constituted in Section 4.2, the model is essentially parameter free. All of the
hyperparameters are set in a relatively diffuse manner (see the discussion at the
beginning of Section 4.3), and the model infers the number of clusters and their
composition with no parameter tuning required. Thus, our code runs “out-of-the-
box” to yield state-of-the-art accuracy on the dataset that we tested. And yet, an
expert experimentalist could desire different clustering results, further improving the performance. Because our inference methodology is based on a biophysical model, all of the hyperparameters have natural and intuitive interpretations. Therefore, adjusting the performance is relatively intuitive. Although all of the results presented above were manifested without any model tuning, we now discuss how one may constitute a single “knob” (parameter) that a neuroscientist may “turn” to examine different kinds of results.

In Section 4.2.2 the variance of additive noise \( e_1, \ldots, e_n \) are controlled by the covariance \( \text{diag}(\eta_1^{-1}, \ldots, \eta_T^{-1}) \). If we set \( \text{diag}(\eta_1^{-1}, \ldots, \eta_T^{-1}) = \omega_0^{-1}I_T \), then parameter \( \omega_0 \) may be tuned to control the variability (diversity) of spikes. The cluster diversity encouraged by setting different values of \( \omega_0 \) in turn manifests different numbers of clusters, which a neuroscientist may adjust as desired. As an example, we consider the publicly available data from Section 4.3.1, and clusterings (color coded) are shown for two settings of \( \omega_0 \) in Figure 4.7. In this figure, each spike is depicted in two-dimensional learned feature space, taking two arbitrary features (because features are not inherently ordered); this is simply for display purposes, as here feature learning is done via dictionary learning, and in general more than two dictionary components are utilized to represent a given waveform.

The value of \( \omega_0 \) defines how much of a given signal is associated with noise \( E_{ij} \), and how much is attributed to the term \( D\Lambda S_{ij} \) characterized by a summation of dictionary elements (see (1)). If \( \omega_0 \) is large, then the noise contribution to the signal is small (because the noise variance is imposed to be small), and therefore the variability in the observed data is associated with variability in the underlying signal (and that variability is captured via the dictionary elements). Since the clustering is performed on the dictionary usage, if \( \omega_0 \) is large we expect an increasing number of clusters, with these clusters capturing the greater diversity/variability in the underlying signal. By contrast, if \( \omega_0 \) is relatively small, more of the signal is attributed to noise \( E_{ij} \), and
the signal components modeled via the dictionary are less variable (variability is attributed to noise, not signal). Hence, as $\omega_0$ diminishes in size we would expect fewer clusters. This phenomenon is observed in the example in Figure 4.7, with this representative of behavior we have observed in a large set of experiments on the rat motor cortex data.

\begin{figure}[h]
\centering
\begin{subfigure}{0.45\textwidth}
\centering
\includegraphics[width=\textwidth]{figure_a}
\caption{(a) Waveforms projected down onto two learned features based on cluster result with $\omega_0 = 10^6$, the number of inferred clusters is two.}
\end{subfigure}
\begin{subfigure}{0.45\textwidth}
\centering
\includegraphics[width=\textwidth]{figure_b}
\caption{(b) Same as (a) with $\omega_0 = 10^8$; the number of inferred clusters is seven.}
\end{subfigure}
\caption{Effect of manually tuning $\omega_0$ to obtain a different number of features for the rat motor cortex data. (a) Waveforms projected down onto two learned features based on cluster result with $\omega_0 = 10^6$, the number of inferred clusters is two. (b) Same as (a) with $\omega_0 = 10^8$; the number of inferred clusters is seven.}
\end{figure}

4.3.5 Sparsely Firing Neurons

Recently, several manuscripts have directly addressed spike sorting in the presence of sparsely firing neurons (Pedreira et al., 2012; Adamos et al., 2012). We operationally define a sparsely firing neuron as a neuron whose spike count has significantly fewer spikes than the other isolated neurons. Based on reviewer recommendations, we assessed the performance of FMM-DL in such regimes utilizing the following synthetic data. First, we extracted spike waveforms from four clusters from the new dataset discussed in Section 4.2.6. We excluded all waveforms that did not clearly separate (Figure 4.8(a1)) to obtain clear clustering criteria (Figure 4.8(a2)). There were 2592,
148, 506, and 64 spikes in the first, second, third, and fourth cluster, respectively. Then, we added real noise—as described in section 4.2.7—to each waveform at two different levels to obtain increasingly noisy and less-well separated clusters (Figure 4.8(b1), (b2), (c1), and (c2)). We applied FMM-DL, Wave-clus (Pedreira et al., 2012) and Wave-clus “forced” (in which we hand tune the parameters to obtain optimal results) and ISOMAP dominant sets (Adamos et al., 2012) to all three signal-to-noise ratio (SNR) regimes to assess our relative performance with the following results.

The third column of Figure 4.8 shows the posterior estimate of the number of clusters for each of the three scenarios. As long as SNR is relatively good, for
example, higher than 2 in this simulation, the posterior number of clusters inferred by FMM-DL correctly has its maximum at four clusters. Similarly, for the good and moderate SNR regimes, the confusion matrix is essentially a diagonal matrix, indicating that FMM-DL assigns spikes to the correct cluster. Only in the poor SNR regime (SNR=1.5), does the posterior move away from the truth. This occurs because Unit 1 becomes over segmented, as depicted in (c2). (c4) shows that only this unit struggles with assignment issues, suggestive of the possibility of a post-hoc correction if desired.

Figure 4.9(a) compares the performance of FMM-DL to previously proposed methods. Even after fine-tuning the Wave-clus method to obtain its optimal performance on these data, FMM-DL yields a better accuracy. In addition to obtaining better point-estimates of spiking, via our Bayesian generative model, we also obtain posteriors over all random variables of our model, including number of spikes per unit. Figure 4.9(b) and (c) show such posteriors, which may be used by the experimentalist to assess data quality.

**Figure 4.9:** Performance analysis in the sparsely firing neuron case on synthetic data based on the Pittsburgh dataset. (a) Accuracy comparisons based on the cluster results under the various SNR. (b) Approximate posterior distributions of error rate for FMM-DL in the different SNR levels. (c) Approximate posterior distributions of spike waveform number for the unit 2, unit 3, and unit 4 under the various SNR regimes.

4.3.6 Computational requirements

The software used for the tests in this paper were written in (non-optimized) Matlab, and therefore computational efficiency has not been a focus. The principal motivat-
ing focus of this study concerned interpretation of longitudinal spike waveforms, as discussed in Section 4.3.2, for which computation speed is desirable, but there is not a need for real-time processing (for example, for a prosthetic). Nevertheless, to give a sense of the computational load for the model, it takes about 20 seconds for each Gibbs sample, when considering analysis of 170800 spikes across $N = 8$ channels; computations were performed on a PC, specifically a Lenevo T420 (CPU is Intel(R) Core (TM) i7 M620 with 4 GB RAM). Significant computational acceleration may be manifested by coding in C, and via development of online methods for Bayesian inference (for example, see (Wang et al., 2011)). In the context of such online Bayesian learning one typically employs approximate variational Bayes inference rather than Gibbs sampling, which typically manifests significant acceleration (Wang et al., 2011).

4.4 Discussion

4.4.1 Summary

A new focused mixture model (FMM) has been developed, motivated by real-world studies with longitudinal electrophysiological data, for which traditional methods like the hierarchical Dirichlet process have proven inadequate. In addition to performing “focused” clustering, the model jointly performs feature learning, via dictionary learning, which significantly improves performance over principal components. We explicitly model the count of signals within a recording period by $p_i$. The rate of neuron firing constitutes a primary information source (Donoghue et al., 2007), and therefore it is desirable that it be modeled. This rate is controlled here by a parameter $\phi_{m}^{(i)}$, and this was allowed to be unique for each recording period $i$. 
4.4.2 Future Directions

In future research one may constitute a mixture model on $\phi^{(i)}_m$, with each mixture component reflective of a latent neural (firing) state; one may also explicitly model the time dependence of $\phi^{(i)}_m$, as in the Mixture of Kalman’s work (Calabrese and Paniski, 2010). Inference of this state could be important for decoding neural signals and controlling external devices or muscles. In future work one may also wish to explicitly account for covariates associated with animal activity (Ventura, 2009a), which may be linked to the firing rate we model here (we may regress $p_i$ to observed covariates).

In the context of modeling and analyzing electrophysiological data, recent work on clustering models has accounted for refractory-time violations (Gasthaus et al., 2009; Calabrese and Paniski, 2010; Chen et al., 2011b), which occur when two or more spikes that are sufficiently proximate are improperly associated with the same cluster/neuron (which is impossible physiologically due to the refractory time delay required for the same neuron to re-emit a spike). The methods developed in (Gasthaus et al., 2009; Chen et al., 2011b) may be extended to the class of mixture models developed above. We have not done so for two reasons: (i) in the context of everything else that is modeled here (joint feature learning, clustering, and count modeling), the refractory-time-delay issue is a relatively minor issue in practice; and (ii) perhaps more importantly, an important issue is that not all components of electrophysiological data are spike related (which are associated with refractory-time issues). As demonstrated in Section 4.3, a key component of the proposed method is that it allows us to distinguish single-unit (spike) events from other phenomena.

Perhaps the most important feature of spike sorting methods that we have not explicitly included in this model is “overlapping spikes” (Bar-Gad et al., 2001; Zhang et al., 2004; Wang and McCallum, 2006; Vargas-Irwin and Donoghue, 2007; Herbst
et al., 2008; Adamos et al., 2010; Franke et al., 2010a). Preliminary analysis of our model in this regime (not shown), inspired by reviewer comments, demonstrated to us that while the FMM-DL as written is insufficient to address this issue, a minor modification to FMM-DL will enable “demixing” overlapping spikes. We are currently pursuing this avenue. Neuronal bursting—which can change the waveform shape of a neuron—is yet another possible avenue for future work.
5.1 Introduction

The recent heightened interest in understanding the brain calls for the development of technologies that will advance our understanding of neuroscience. Crucial for this endeavor is the advancement of our ability to understand the dynamics of the brain, via the measurement of large populations of neural activity at the single neuron level. Such reverse engineering efforts benefit from real-time decoding of neural activity, to facilitate effectively adapting the probing stimuli. Regardless of the experimental apparatus used (e.g., electrodes or calcium imaging), real-time decoding of individual neuron responses requires identifying and labeling individual spikes from recordings from large populations. In other words, real-time decoding requires real-time spike sorting.

Automatic spike sorting methods are continually evolving to deal with more sophisticated experiments. Most recently, several methods have been proposed to (i) learn the number of separable neurons on each electrode or “multi-trode” (Pillow
et al., 2013; Prentice et al., 2011), or (ii) operate online to resolve overlapping spikes from multiple neurons (Franke et al., 2010b). To our knowledge, no method to date is able to simultaneously address both of these challenges.

We develop a nonparametric Bayesian continuous-time generative model of population activity. Our model explains the continuous output of each neuron by a latent marked Poisson process, with the “marks” characterizing the shape of each spike. Previous efforts to address overlapping spiking often assume a fixed kernel for each waveform, but joint intracellular and extracellular recording clearly indicate that this assumption is false (see Figure 5.4.4). Thus, we assume that the statistics of the marks are time-varying. We use the framework of completely random measures to infer how many of a potentially infinite number of neurons (or single units) are responsible for the observed data, simultaneously characterizing spike times and waveforms of these neurons.

We describe an intuitive discrete-time approximation to the above infinite-dimensional continuous-time stochastic process, then develop an online variational Bayesian inference algorithm for this model. Via numerical simulations, we demonstrate that our inference procedure improves over the previous state-of-the-art, even though we allow the other methods to use the entire dataset for training, whereas we learn online. Moreover, we demonstrate that we can effectively track the time-varying changes in waveform, and detect overlapping spikes. Indeed, it seems that the false positive detections from our approach have indistinguishable first order statistics from the true positives, suggesting that second-order methods may be required to reduce the false positive rate (i.e., template methods may be inadequate). Our work therefore suggests that further improvements in real-time decoding of activity may be most effective if directed at simultaneous real-time spike sorting and decoding. To facilitate such developments and support reproducible research, all code and data associated with this work is provided in the Supplementary Materials.
5.2 Model

Our data is a time-series of multielectrode recordings \( X \equiv (x_1, \cdots, x_T) \), and consists of \( T \) recordings from \( M \) channels. As in usual measurement systems, the recording times lie on regular grid, with interval length \( \Delta \), and \( x_t \in \mathbb{R}^M \) for all \( t \). Underlying these observations is a continuous-time electrical signal driven by an unknown number of neurons. Each neuron generates a continuous-time voltage trace, and the outputs of all neurons are superimposed and discretely sampled to produce the recordings \( X \). At a high level, in §5.2.1 we model the continuous-time output of each neuron as a series of idealized Poisson events smoothed with appropriate kernels, while §5.2.2 uses the Gamma process to develop a nonparametric prior for an entire population. §5.2.3 then describes a discrete-time approximation based on the Bernoulli approximation to the Poisson process. For conceptual clarity, we restrict ourselves to single channel recordings until §5.2.4, where we describe the complete model for multichannel data.

5.2.1 Modeling the continuous-time output of a single neuron

There is a rich literature characterizing the spiking activity of a single neuron (Gerstner and Kistler, 2002) accounting in detail for factors like non-stationarity, refractoriness and spike waveform. We however make a number of simplifying assumptions (some of which we later relax). First, we model the spiking activity of each neuron are stationary and memoryless, so that its set of spike times are distributed as a homogeneous Poisson process (PP). We model the neurons themselves are heterogeneous, with the \( i^{th} \) neuron having an (unknown) firing rate \( \lambda_i \). Call the ordered set of spike times of the \( i^{th} \) neuron \( T_i = (\tau_{i1}, \tau_{i2}, \ldots) \); then the time between successive elements of \( T_i \) is exponentially distributed with mean 1/\( \lambda_i \). We write this as \( T_i \sim \text{PP}(\lambda_i) \).
The actual electrical output of a neuron is not binary; instead each spiking event is a smooth perturbation in voltage about a resting state. This perturbation forms the shape of the spike, with the spike shapes varying across neurons as well as across different spikes of the same neuron. However, each neuron has its own characteristic distribution over shapes, and we let $\theta_i \in \Theta$ parametrize this distribution for neuron $i$. Whenever this neuron emits a spike, a new shape is drawn independently from the corresponding distribution. This waveform is then offset to the time of the spike, and contributes to the voltage trace associated with that spike.

The complete recording from the neuron is the superposition of all these spike waveforms plus noise. Rather than treating the noise as white as is common in the literature (Lewicki, 1998b), we allow it to exhibit temporal correlation, recognizing that the ‘noise’ is in actual fact background neural activity. We model it as a realization of a Gaussian process (GP) (Rasmussen and Williams, 2006), with the covariance kernel $K$ of the GP determining the temporal structure. We use an exponential kernel, modeling the noise as Markov.

We model each spike shape as weighted superpositions of a dictionary of $K$ basis functions $d(t) = (d_1(t), \ldots, d_K(t))^T$. The dictionary elements are shared across all neurons, and each is a real-valued function of time, i.e., $d_k \in L_2$. Each spike time $\tau_{ij}$ is associated with a random $K$-dimensional weight vector $y_{ij} \equiv (y_{ij1}, \ldots, y_{ijk})^T$, and the shape of this spike at time $t$ is given by the weighted sum $\sum_{k=1}^K y_{ijk} d_k(t - \tau_{ij})$. We assume $y_{ij} \sim N_K(\mu^*, \Sigma^*)$, indicating a $K$-dimensional Gaussian distribution with mean and covariance given by $(\mu^*, \Sigma^*)$; we let $\theta_i \equiv (\mu^*_i, \Sigma^*_i)$. Then, at any time $t$, the output of neuron $i$ is $x_i(t) = \sum_{j=1}^{[T_i]} \sum_{k=1}^K y_{ijk} d_k(t - \tau_{ij})$.

The total signal received by any electrode is the superposition of the outputs of all neurons. Assume for the moment there are $N$ neurons, and define $T = \cup_{i \in [N]} T_i$ as the (ordered) union of the spike times of all neurons. Let $\tau_l \in T$ indicate the
time of the $l$th overall spike, whereas $\tau_{ij} \in \mathcal{T}_i$ is the time of the $j$th spike of neuron $i$. This defines a pair of mappings: $\nu : \{\mathcal{T}\} \rightarrow [N]$, and $p : \{\mathcal{T}\} \rightarrow \mathcal{T}_{\nu_l}$, with $\tau_l = \tau_{\nu_l p_l}$. In words, $\nu_l \in N$ is the neuron to which the $l$th element of $\mathcal{T}$ belongs, while $p_l$ indexes this spike in the spike train $\mathcal{T}_{\nu_l}$. Let $\theta_l \equiv (\mu_l, \Sigma_l)$ be the neuron parameter associated with spike $l$, so that $\theta_l = \theta_{\nu_l}^*$. Finally, define $y_l \equiv (y_{l1}, \ldots, y_{lK}) \mathcal{T} = y_{\nu_l p_l}^*$ as the weight vector of spike $\tau_l$. Then, we have that

$$x(t) = \sum_{i \in [N]} x_i(t) = \sum_{l \in \mathcal{T}} \sum_{k \in [K]} y_{lk} d_k(t - \tau_l), \quad \text{where } y_l \sim N_K(\mu_l, \Sigma_l). \quad (5.1)$$

From the superposition property of the Poisson process (Kingman, 1993), the overall spiking activity $\mathcal{T}$ is Poisson with rate $\Lambda = \sum_{i \in [N]} \lambda_i$. Each event $\tau_l \in \mathcal{T}$ has a pair of labels, its neuron parameter $\theta_l \equiv (\mu_l, \Sigma_l)$, and $y_l$, the weight-vector characterizing the spike shape. We view these weight-vectors as the “marks” of a marked Poisson process $\mathcal{T}$. From the properties of the Poisson process, we have that the marks $\theta_l$ are drawn i.i.d. from a probability measure $G(d\theta) = 1/\Lambda \sum_{i \in [N]} \lambda_i \delta_{\theta_i^*}$.

With probability one, the neurons have distinct parameters, so that the mark $\theta_l$ identifies the neuron which produced spike $l$: $G(\theta_l = \theta_i^*) = P(\nu_l = i) = \lambda_i/\Lambda$. Given $\theta_l$, $y_l$ is distributed as in Eq. (5.1). The output waveform $x(t)$ is then a linear functional of this marked Poisson process.

5.2.2 A nonparametric model of population activity

In practice, the number of neurons driving the recorded activity is unknown. We do not wish to bound this number a priori, moreover we expect this number to increase as we record over longer intervals. This suggests a nonparametric Bayesian approach: allow the total number of underlying neurons to be infinite. Over any finite interval, only a finite subset of these will be active, and typically, these dominate spiking activity over any interval. This elegant and flexible modeling approach allows the data to suggest how many neurons are active, and has already proved successful.
in neuroscience applications (Wood and Black, 2008a). We use the framework of completely random measures (CRMs) (Kingman, 1967) to model our data. CRMs have been well studied in the Bayesian nonparametrics community, and there is a wealth of literature on theoretical properties, as well as posterior computation; see e.g. (James et al., 2009; Hjort, 1990; Thibaux and Jordan, 2007). Recalling that each neuron is characterized by a pair of parameters \((\lambda_i, \theta_i^*)\), we map the infinite collection of pairs \(\{(\lambda_i, \theta_i^*)\}\) to an random measure \(\Lambda(\cdot)\) on \(\Theta\): \(\Lambda(d\theta) = \sum_{i=1}^{\infty} \lambda_i \delta_{\theta_i^*}\).

For a CRM, the distribution over measures is induced by distributions over the infinite sequence of weights, and the infinite sequence of their locations. The weights \(\lambda_i\) are the jumps of a Lévy process (Sato, 1990), and their distribution is characterized by a Lévy measure \(\rho(\lambda)\). The locations \(\theta_i^*\) are drawn i.i.d. from a base probability measure \(H(\theta^*)\). As is typical, we assume these to be independent.

We set the Lévy measure \(\rho(\lambda) = \alpha \lambda^{-1} \exp(-\lambda)\), resulting in a CRM called the Gamma process (\(\Gamma\)P) (Applebaum, 2004). The Gamma process has the convenient property that the total rate \(\Lambda(\cdot) = \sum_{i=1}^{\infty} \lambda_i\) is Gamma distributed (and thus conjugate to the Poisson process prior on \(\mathcal{T}\)). The Gamma process is also closely connected with the Dirichlet process (Ferguson, 1973b), which will prove useful later on. To complete the specification on the Gamma process, we set \(H(\theta^*)\) to the conjugate normal-Wishart distribution with hyperparameters \(\phi\).

It is easy to directly specify the resulting continuous-time model, we provide the equations in the Supplementary Material. However it is more convenient to represent the model using the marked Poisson process of Eq. (5.1). There, the overall process \(\mathcal{T}\) is a rate \(\Lambda\) Poisson process, and under a Gamma process prior, \(\Lambda\) is Gamma(\(\alpha, 1\)) distributed (Ferguson, 1973b). The labels \(\theta_i\) assigning events to neurons are drawn i.i.d. from a normalized Gamma process: \(G(d\theta) = (1/\Lambda) \sum_{i=1}^{\infty} \lambda_i\).

\(G(d\theta)\) is a random probability measure (RPM) called a normalized random measure (James et al., 2009). Crucially, a normalized Gamma process is the Dirichlet
process (DP) (Ferguson, 1973b), so that the spike parameters $\theta$ are i.i.d. draws with a DP-distributed RPM. For spike $l$, the shape vector is drawn from a normal with parameters $(\mu_l, \Sigma_l)$; these are thus draws from a DP mixture (DPM) of Gaussians (Lo, 1984).

We can exploit the connection with the DP to integrate out the infinite-dimensional measure $G(\cdot)$ (and thus $\Lambda(\cdot)$), and assign spikes to neurons via the so-called Chinese restaurant process (CRP) (Pitman, 2002). Under this scheme, the $l^{th}$ spike is assigned the same parameter as an earlier spike with probability proportional to the number of earlier spikes having that parameter. It is assigned a new parameter (and thus, a new neuron is observed) with probability proportional to $\alpha$. Letting $C_t$ be the number of neurons observed until time $t$, and $T_i^t = T_i \cap [0,t)$ be the times of spikes produced by neuron $i$ before time $t$, we then have for spike $l$ at time $t = \tau_l$:

$$\theta_l = \theta_{\nu_l}^*, \text{ where } P(\nu_l = i) \propto \begin{cases} \left| T_i^t \right| & i \in [C_t], \\ \alpha & i = C_t + 1, \end{cases}$$

This marginalization property of the DP allows us to integrate out the infinite-dimensional rate vector $\Lambda(\cdot)$, and sequentially assign spikes to neurons based on the assignments of earlier spikes. This requires one last property: for the Gamma process, the RPM $G(\cdot)$ is independent of the total mass $\Lambda$. Consequently, the clustering of spikes (determined by $G(\cdot)$) is independent of the rate $\Lambda$ at which they are produced.

We then have the following model:

$$\mathcal{T} \sim \text{PP}(\Lambda), \quad \text{where } \Lambda \sim \Gamma P(\alpha, 1),$$

$$y_l \sim N_K(\mu_l, \Sigma_l), \quad \text{where } (\mu_l, \Sigma_l) \sim \text{CRP}(\alpha, H_\phi(\cdot)), \quad l \in [||\mathcal{T}||],$$

$$x(t) = \sum_{l \in \mathcal{T}} \sum_{k \in [K]} y_{lk} d_k(t - \tau_l) + \epsilon_t \quad \text{where } \epsilon \sim \text{GP}(0,K).$$

99
5.2.3 A discrete-time approximation

The previous subsections modeled the continuous-time voltage output of a neural population. Our data on the other hand consists of recordings at a discrete set of times. While it is possible to make inferences about the continuous-time process underlying these discrete recordings, in this paper, we restrict ourselves to the discrete case. The marked Poisson process characterization of Eq. 5.3 leads to a simple discrete-time approximation of our model.

Recall first the Bernoulli approximation to the Poisson process: a sample from a Poisson process with rate $\Lambda$ can be approximated by discretizing time at a granularity $\Delta$, and assigning each bin an event independently with probability $\Lambda\Delta$ (the accuracy of the approximation increasing as $\Delta$ tends to 0). To approximate the marked Poisson process $T$, all that is additionally required is to assign marks $\theta_i$ and $y_i$ to each event in the Bernoulli approximation. Following Eqs. (5.3b) and (5.3c), the $\theta_i$’s are distributed according to a Chinese restaurant process, while each $y_i$ is drawn from a normal distribution parametrized by the corresponding $\theta_i$. We discretize the elements of dictionary as well, yielding discrete dictionary elements $\tilde{d}_{k,:} = (\tilde{d}_{k,1}, \ldots, \tilde{d}_{k,L}) T$. These form the rows of a $K \times L$ matrix $\tilde{D}$ (we call its columns $\tilde{d}_{i,:}$). The shape of the $j^{th}$ spike is now a vector of length $L$, and for a weight vector $y$, is given by $\tilde{D} y$.

We can simplify notation a little for the discrete-time model. Let $t$ index time-bins (so that for an observation interval of length $T$, $t \in [T/\Delta]$). We use tildes for variables indexed by bin-position. Thus, $\tilde{\nu}_t$ and $\tilde{\theta}_t$ are the neuron and neuron parameter associated with time bin $t$, and $\tilde{y}_t$ is its weight-vector. Let the binary variable $\tilde{z}_t$ indicate whether or not a spike is present in time bin $t$ (recall that $\tilde{z}_t \sim$ Bernoulli($\Lambda\Delta$)). If there is no spike associated with bin $t$, then we ignore the marks $\tilde{\mu}$ and $\tilde{y}$. Thus the output at time $t$, $x_t$ is given by $x_t = \sum_{h=1}^T \tilde{z}_{t-h} \tilde{d}_{h,:} \tilde{y}_{t-h-1} + \epsilon_t$. Note that the noise $\epsilon_t$ is now a discrete-time Markov Gaussian process. Let $a$ and
$r_t$ be the decay and innovation of the resulting autoregressive (AR) process, so that $\epsilon_{t+1} = a\epsilon_t + r_t$.

5.2.4 Correlations in time and across electrodes

So far, for simplicity, we restricted our model to recordings from a single channel. We now describe the full model we use in experiments with multichannel recordings. We let every spike affect the recordings at all channels, with the spike shape varying across channels. For spike $l$ in channel $m$, call the weight-vector $y_l^m$. All these vectors must be correlated as they correspond to the same spike; we do this simply by concatenating the set of vectors into a single $MK$-element vector $y_l = (y_l^1; \cdots; y_l^M)$, and modeling this as a multivariate normal. In principle, one might expect the associated covariance matrix to possess a block structure (corresponding to the subvector associated with each channel); however, rather than building this into the model, we allow the data to inform us about any such structure.

We also relax the requirement that the parameters $\theta^*$ of each neuron remain constant, and instead allow $\mu^*$, the mean of the weight-vector distribution, to evolve with time (we keep the covariance parameter $\Sigma^*_i$ fixed, however). Such flexibility can capture effects like changing cell characteristics or moving electrodes. Like the noise term, we model the time-evolution of this quantity as a realization of a Markov Gaussian process; again, in discrete-time, this corresponds to a simple first-order AR process. With $B \in \mathbb{R}^{K \times K}$ the transition matrix, and $r_t \in \mathbb{R}^K$, independent Gaussian innovations, we have $\mu_{t+1}^* = B\mu_t^* + r_t$. Where we previously had a DP mixture of Gaussians, we now have a DP mixture of GPs. Each neuron is now associated with a vector-valued function $\theta^*(\cdot)$, rather than a constant. When a spike at time $\tau_l$ is assigned to neuron $i$, it is assigned a weight-vector $y_l$ drawn from a Gaussian with mean $\mu_i^*(\tau_l)$. Algorithm 2 in the Supplementary Material summarizes the full generative mechanism for the full discrete-time model.
5.3 Inference

There exists a vast literature on computational approaches to posterior inference for Bayesian nonparametric models, especially so for models based on the DP. Traditional approaches are sampling-based, typically involving Markov chain Monte Carlo techniques (see eg. (Neal, 2000; Ishwaran and James, 2001b)), and recently there has also been work on constructing deterministic approximations to the intractable posterior (eg. (Blei and Jordan, 2006b; Minka and Ghahramani, 2003)). Our problem is complicated by two additional factors. The first is the convolutional nature of our observation process, where at each time, we observe a function of the previous observations drawn from the DPMM. This is in contrast to the usual situation where one directly observes the DPMM outputs themselves. The second complication is a computational requirement: typical inference schemes are batch methods that are slow and computationally expensive. Our ultimate goal, on the other hand, is to perform inference in real time, making these approaches unsuitable. Instead, we develop an online algorithm for posterior inference. Our algorithm is inspired by the sequential update and greedy search (SUGS) algorithm of (Wang and Dunson, 2009), though that work was concerned with the usual case of i.i.d. observations from a DPMM. We generalize SUGS to our observation process, also accounting for the time-evolution of the cluster parameters and correlated noise.

Below, we describe a single iteration of our algorithm for the case a single electrode; generalizing to the multielectrode case is straightforward. At each time $t$, our algorithm maintains the set of times of the spikes it has inferred from the observations so far. It also maintains the identities of the neurons that it assigned each of these spikes to, as well as the weight vectors determining the shapes of the associated spike waveforms. We indicate these point estimates with the hat operator, so, for example $\hat{T}_i^t$ is the set of estimated spike times before time $t$ assigned to neuron
In addition to these point estimates, the algorithm also keeps a set of posterior distributions $q_{it}(\theta^*_i)$ where $i$ spans over the set of neurons seen so far (i.e. $i \in [\hat{C}_t]$). For each $i$, $q_{it}(\theta^*_i)$ approximates the distribution over the parameters $\theta^*_i \equiv (\mu^*_i, \Sigma^*_i)$ of neuron $i$ given the observations until time $t$.

Having identified the time and shape of spikes from earlier times, we can calculate their contribution to the recordings $x^L_t \equiv (x_t, \cdots, x_{t+L-1})^T$. Recalling that the basis functions $D$, and thus all spike waveforms, span $L$ time bins, the residual at time $t + t_1$ is then given by $\delta x_{t+t_1} = x_t - \sum_{h \in [L-t_1]} \hat{z}_{t-h} D \hat{y}_{t-h}$ (at time $t$, for $t_1 > 0$, we define $\hat{z}_{t+t_1} = 0$). We treat the residual $\delta x_t = (\delta x_t, \cdots, \delta x_{t+L})^T$ as an observation from a DP mixture model, and use this to make hard decisions about whether or not this was produced by an underlying spike, what neuron that spike belongs to (one of the earlier neurons or a new neuron), and what the shape of the associated spike waveform is. The latter is used to calculate $q_{i,t+1}(\theta^*_i)$, the new distribution over neuron parameters at time $t + 1$. Our algorithm proceeds recursively in this manner.

For the first step we use Bayes’ rule to decide whether there is a spike underlying the residual:

$$P(\tilde{z}_t = 1|\delta x_t) \propto \sum_{i \in \hat{C}_{t+1}} P(\delta x_t, \nu_t = i|\tilde{z}_t = 1)P(\tilde{z}_t = 1) \tag{5.4}$$

Here, $P(\delta x_t|\nu_t = i, \tilde{z}_t = 1) = \int_{\Theta} P(\delta x_t|\theta_t) q_{it}(\theta_t) d\theta_t$, while $P(\nu_t = i|\tilde{z}_t = 1)$ follows from the CRP update rule (equation (5.2)). $P(\delta x_t|\theta_t)$ is just the normal distribution, while we restrict $q_{it}(\cdot)$ be the family of normal-Wishart distribution. We can then evaluate the integral, and then summation (5.4) to approximate $P(\tilde{z}_t = 1|\delta x_t)$. If this exceeds a threshold of 0.5 we decide that there is a spike present at time $t$, otherwise, we set $\tilde{z}_t = 0$. Observe that making this decision involves marginalizing over all possible cluster assignments $\nu_t$, and all values of the weight vector $y_t$. On the other hand, having made this decision, we collapse these posterior distributions to point estimates $\hat{\nu}_t$ and $\hat{y}_t$ equal to their MAP values.
In the event of a spike ($\hat{z}_t = 1$), we use these point estimates to update the posterior distribution over parameters of cluster $\hat{\nu}_t$, to obtain $q_{t,t+1}(\cdot)$ from $q_{t,t}(\cdot)$; this is straightforward because of conjugacy. We follow this up with an additional update step for the distributions of the means of all clusters: this is to account for the AR evolution of the cluster means. We use a variational update to keep $q_{t,t+1}(\cdot)$ in the normal-Wishart distribution. Finally we take a stochastic gradient step to update any hyperparameters we wish to learn. We provide all details in the Supplementary material.

5.4 Results

5.4.1 Data

In the following, we refer to our algorithm as **Opass**\(^1\). We used two different datasets to demonstrate the efficacy of **Opass**. First, the ever popular, publicly available HC1 dataset as described in (Henze et al., 2000). We used the dataset d533101 that consisted of an extracellular tetrode and a single intracellular electrode. The recording was made simultaneously on all electrodes and was set up such that the cell with the intracellular electrode was also recorded on the extracellular array implanted in the hippocampus of an anesthetized rat. The intracellular recording is relatively noiseless and gives nearly certain firing times of the intracellular neuron. The extracellular recording contains the spike waveforms from the intracellular neuron as well as an unknown number of additional neurons. The data is a 4-minute recording at a 10 kHz sampling rate.

The second dataset comes from novel NeuroNexus devices implanted in the rat motor cortex. The data was recorded at 32.5 kHz in freely-moving rats. The first device we consider is a set of 3 channels of data (Fig. D.1). The neighboring electrode sites in these devices have 30 $\mu$m between electrode edges and 60 $\mu$m between

\(^1\) Online gamma Process Autoregressive Spike Sorting
electrode centers. These devices are close enough that a locally-firing neuron could appear on multiple electrode sites (Prentice et al., 2011), so neighboring channels warrant joint processing. The second device has 8-channels (see Fig. D.1), but is otherwise similar to the first. We used a 15-minute segment of this data for our experiments.

For both datasets, we preprocessed with a high-pass filter at 800 Hz using a fourth order Butterworth filter before we analyzed the time series. To define $D$, we used the first five principle components of all spikes detected with a threshold (three times the standard deviation of the noise above the mean) in the first five seconds. The noise standard deviation was estimated both over the first five seconds of the recording as well as the entire recording, and the estimate was nearly identical. Our results were also robust to minor variations in the choice of the number of principal components. The autoregressive parameters were estimated by using lag-1 autocorrelation on the same set of data. For the multichannel algorithms we estimate the covariance between channels and normalize by our noise variance estimate.

Each algorithm gives a clustering of the detected spikes. In this dataset, we only have a partial ground truth, so we can only verify accuracy for the neuron with the intracellular (IC) recording. We define a detected spike to be an IC spike if the IC recording has a spike within 0.5 milliseconds (ms) of the detected spike in the extracellular recording. We define the cluster with the greatest number of intracellular spikes as a the “IC cluster”. We refer to these data as “partial ground truth data”, because we know the ground truth spike times for one of the neurons, but not all the others.

5.4.2 Algorithm Comparisons

We compare a number of variants of $\text{OPASS}$, as well as several previously proposed methods, as described below. The vanilla version of $\text{OPASS}$ operates on a single
channel with colored noise. When using multiple channels, we append an “M” to obtain \textbf{MOpass}. When we model the mean of the waveforms as an auto-regressive process, we “post-pend” to obtain \textbf{OpassR}. We compare these variants of \textbf{Opass} to Gaussian mixture models and k-means (Lewicki, 1998b) with \(N\) components (\textbf{Gmm-N} and \textbf{K-N}, respectively), where \(N\) indicates the number of components. We compare with a Dirichlet Process Mixture Model (\textbf{DPMM}) (Wood and Black, 2008a) as well as the Focused Mixture Model (\textbf{Fmm}) (Carlson et al., 2013b), a recently proposed Bayesian generative model with state-of-the-art performance. Finally, we compare with \textbf{OSORT} (Rutishauser et al., 2006), an online sorting algorithm. Only \textbf{Opass} and \textbf{OSORT} methods were online as we desired to compare to the state-of-the-art \textit{batch} algorithms which use all the data. Note that \textbf{Opass} algorithms learned \(D\) from the first five seconds of data, whereas all other algorithms used a dictionary learned from the entire data set.

The single-channel experiments were all run on channel 2 (the results were nearly identical for all channels). The spike detections for the offline methods used a threshold of three times the noise standard deviation (Lewicki, 1998b) (unless stated otherwise), and windowed at a size \(L = 30\). For multichannel data, we concatenated the \(M\) channels for each waveform to obtain a \(M \times L\)-dimensional vector.

The online algorithms were all run with weakly informative parameters. For the normal-Wishart, we used \(\mu_0 = 0\) , \(\lambda_0 = 0.1\), \(W = 10I\), and \(\nu = 1\) (\(I\) is the identity matrix). The AR process corresponded to a GP with length-scale 30 seconds, and variance 0.1. \(\alpha\) was set to 0.1. The parameters were insensitive to minor changes. Running time in unoptimized MATLAB code for 4 minutes of data was 31 seconds for a single channel and 3 minutes for all 4 channels on a 3.2 GHz Intel Core i5 machine with 6 GB of memory (see Supplementary Fig. D.7 for details).
5.4.3 Performance on partial ground truth data

The main empirical result of our contribution is that all variants of \textsc{Opass} detect more true positives with fewer false positives than any of the other algorithms on the partial ground truth data (see Fig. 5.1). The only comparable result is the \textsc{Osort}; however, the \textsc{Osort} algorithm split the IC cluster into 2 different clusters and we combined the two clusters into one by hand. Our improved sensitivity and specificity is \textit{despite} the fact that \textsc{Opass} is fully online, whereas all the algorithms (besides \textsc{Osort}) that we compare to are batch algorithms using all data for all spikes. Note that all the comparison algorithms pre-process the data via thresholding at some constant (which we set to three standard deviations above the mean). To assess the extent to which performance of \textsc{Opass} is due to \textit{not} thresholding, we implement \textsc{Fake-Opass}, which thresholds the data. Indeed, \textsc{Fake-Opass}'s performance is much like that of the batch algorithms. To get uncertainty estimates, we split the data into ten random two minute segments and repeat this analysis and the results are qualitatively similar.

One possible explanation for the relatively poor performance of the batch algorithms as compared to \textsc{Opass} is a poor choice of the important—but often overlooked—threshold parameter. The right panel of Fig. 5.1 shows the receiver operating characteristic (ROC) curve for the k-means algorithms as well as \textsc{Opass} and \textsc{MoPass} (where \textsc{M} indicates multichannel, see below for detail). Although we typically run \textsc{Opass} without tuning parameters, the prior on $\Lambda$ sets the expected number of spikes, which we can vary in a kind of “empirical Bayes” strategy. Indeed, the \textsc{Opass} curves are fully above the batch curves for all thresholds and priors, suggesting that regardless of which threshold one chooses for pre-processing, \textsc{Opass} always does better on these data than all the competitor algorithms. Moreover, in \textsc{Opass} we are able to infer the parameter $\Lambda$ at a reasonable point, and the inferred $\Lambda$ is shown in the left
panel of Fig. 5.1. and the points along the curve in the right panel. These figures also reveal that using the correlated noise model greatly improves performance.

The above analysis suggests \texttt{OPASS}'s ability to detect signals more reliably than thresholding contributes to its success. In the following, we provide evidence suggesting how several of \texttt{OPASS}'s key features are fundamental to this improvement.

### 5.4.4 Overlapping Spike Detection

A putative reason for the improved sensitivity and specificity of \texttt{OPASS} over other algorithms is its ability to detect overlapping spikes. When spikes overlap, although the result can accurately be modeled as a linear sum in voltage space, the resulting waveform often does not appear in any cluster in PC space (see (Pillow et al., 2013)). However, our online approach can readily find such overlapping spikes. Fig. 5.2 (top left panel) shows one example of 135 examples where \texttt{OPASS} believed that multiple waveforms were overlapping. Note that even though the waveform peaks are approximately 1 ms from one another, thresholding algorithms do not pick up these spikes, because they look different in PC space.

Indeed, by virtue of estimating the presence of multiple spikes, the residual squared error between the expected voltage and observed voltage shrinks for this snippet (bottom left). The right panel of Fig. 5.2 shows the density of the residual errors for all putative overlapping spikes. The mass of this density is significantly smaller than the mass of the other scenarios. Of the 135 pairs of overlapping spikes, 37 of those spikes came from the intracellular neuron. Thus, while it seems detecting overlapping spikes helps, it does not fully explain the improvements over the competitor algorithms.
Time-Varying Waveform Adaptation As has been demonstrated previously (Calabrese and Paninski, 2011), the waveform shape of a neuron may change over time. The mean waveform over time for the intracellular neuron is shown in Fig. 5.4.4. Clearly, the mean waveform is changing over time. Moreover, these changes are reflected in the principal component space (Fig. 5.4.4). We therefore compared means and variances $\text{Opass}$ with $\text{OpassR}$, which models the mean of the dictionary weights via an auto-regressive process. Fig. 5.4.4 shows that the auto-regressive model for the mean dictionary weights yields a time-varying posterior (top), whereas the static prior yields a constant posterior mean with increasing posterior marginal variances (bottom). More precisely, the mean of the posterior standard deviations for the time-varying prior is about half of that for the static prior’s posteriors. Indeed, the $\text{OpassR}$ yields 11 more true detections than $\text{Opass}$.

Multielectrode Array $\text{Opass}$ achieved a heightened sensitivity by incorporating multiple channels (see $\text{MOpass}$ point in Fig. 5.1). We further evaluate the impact of multiple channels using a three channel NeuroNexus shank (Supp. Fig. D.1). In Fig. 5.4.4 we show the top two most prevalent waveforms from these data across the three electrodes. Had only the third electrode been used, these two waveforms would not be distinct (as evidenced by their substantial overlap in PC space upon using only the third channel in Fig. D.1). This suggests that borrowing strength across electrodes improves detection accuracy. Supplementary Fig. D.6 shows a similar plot for the eight channel data.
5.5 Discussion

Our improved sensitivity and specificity seem to arise from multiple sources including (i) improved detection, (ii) accounting for correlated noise, (iii) capturing overlapping spikes, (iv) tracking waveform dynamics, and (v) utilizing multiple channels. While others have developed closely related Bayesian models for clustering (Wood and Black, 2008a; Gasthaus et al., 2009), deconvolution based techniques (Pillow et al., 2013), time-varying waveforms (Calabrese and Paninski, 2011), or online methods (Rutishauser et al., 2006; Franke et al., 2010b), we are the first to our knowledge to incorporate all of these.

An interesting implication of our work is that it seems that our errors may be irreconcilable using merely first order methods (that only consider the mean waveform to detect and cluster). Supp. Fig. D.4a shows the mean waveform of the true and false positives are essentially identical, suggesting that even in the full 30-dimensional space excluding those waveforms from intracellular cluster would be difficult. Projecting each waveform into the first two PCs is similarly suggestive, as the missed positives do not seem to be in the cluster of the true positives (Supp. Fig. D.4b). Thus, in future work, we will explore dynamic and multiscale dictionaries (Chen et al., 2012), as well as incorporate a more rich history and stimulus dependence.
**Figure 5.1**: **Opass** achieves improved sensitivity and specificity over all competing methods on partial ground truth data. (a) True positive and false positive rates for all variants of **Opass** and several competing algorithms. (b) ROC curves demonstrating that **Opass** outperforms all competitor algorithms, regardless of threshold (● indicates learning Λ from the data).

**Figure 5.2**: **Opass** detects multiple overlapping waveforms (Top Left) The observed voltage (solid black), MAP waveform 1 (red), MAP waveform 2 (blue), and waveform from the sum (dashed-black). (Bottom Left) Residuals from same example snippet, showing a clear improvement in residuals.
**Figure 5.3:** The IC waveform changes over time, which our posterior parameters track. (a) Mean IC waveforms over time. Each colored line represents the mean of the waveform averaged over 24 seconds with color denoting the time interval. This neuron decreases in amplitude over the period of the recording. (b) The same waveforms plotted in PC space still captures the temporal variance. (c) The mean and standard deviation of the waveforms at three time points for the auto-regressive prior on the mean waveform (top) and static prior (bottom). While the auto-regressive prior admits adaptation to the time-varying mean, the posterior of the static prior simply increases its variance.

**Figure 5.4:** Improving $\text{0Pass}$ by incorporating multiple channels. The top 2 most prevalent waveforms from the NeuroNexus dataset with three channels. Note that the left panel has a waveform that appears on both channel 2 and channel 3, whereas the waveform in the right panel only appears in channel 3. If only channel 3 was used, it would be difficult to separate these waveform.
6

Joint model for Spike Trains and Local Field Potentials

6.1 Introduction

One of the most fundamental challenges in neuroscience is the “large-scale integration problem”: how does distributed neural activity lead to precise, unified cognitive moments Varela et al. (2001). This paper seeks to examine this challenge from the perspective of extracellular electrodes inserted into the brain. An extracellular electrode inserted into the brain picks up two types of signals: (1) the local field potential (LFP), which represents local oscillations in frequencies below 200 Hz; and (2) single neuron action potentials (also known as “spikes”), which typically occur in frequencies of 0.5 kHz. LFPs represent network activity summed over long distances, whereas action potentials represent the precise activity of cells near the tip of an electrode. Although action potentials are often treated as the “currency” of information transfer in the brain, relationships between behaviors and LFP activity can be equally precise, and sometimes even more precise, than those with the activity of individual neurons Pesaran et al. (2002); Mehring et al. (2003). Further, LFP network disruptions are highly implicated in many forms of psychiatric disease.
Uhlhaas and Singer (2010). This has led to much interest in understanding the mechanisms of how LFPs and action potentials interact to create specific types of behaviors. New multisite recording techniques that allow simultaneous recordings from a large number of brain regions provide unprecedented opportunities to study these interactions. However, this type of multi-dimensional data poses significant challenges that require new analysis techniques.

Three of the most challenging characteristics of multisite recordings are that: 1) the networks they represent are dynamic in space and time, 2) subpopulations of neurons within a local area can have different functions and may therefore relate to LFP oscillations in specific ways, and 3) different frequencies of LFP oscillations often relate to single neurons in specific ways Le Van Quyen and Bragin (2007). Here new models are proposed to examine the relationship between neurons and neural networks that accommodate these characteristics. First, each LFP in a brain region is modeled as convolutions between a bounded-time dictionary element and the observed spike trains. Critically, the convolutional factors are allowed to be dynamic, by binning the LFP and spike time series, and modeling the dictionary element for each bin of the time series. Next, a clustering model is proposed making each neuron’s dictionary element a scaled version of an autoregressive template shared among all neurons in a cluster. This allows one to identify sub-populations of neurons that have similar dynamics over their functional connectivity to a brain region. Finally, we provide a strategy for exploring which frequency bands characterize spike-to-LFP functional connectivity. We show, using two novel multi-region electrophysiology datasets from mice, how these models can be used to identify coordinated interactions within and between different neuronal subsystems, defined jointly by the activity of single cells and LFPs. These methods may lead to better understanding of the relationship between brain activity and behavior, as well as the pathology underlying brain diseases.
6.2 Model

6.2.1 Data and notation

The data used here consists of multiple LFP and spike-train time series, measured simultaneously from \( M \) regions of a mouse brain. Spike sorting is performed on the spiking data by a VB implementation of Carlson et al. (2013b), from which \( J \) single units are assumed detected from across the multiple regions (henceforth we refer to single units as “neurons”); the number of observed neurons \( J \) depends on the data considered, and is inferred as discussed in Carlson et al. (2013b). Since multiple microwires are inserted into single brain regions in our experiments (described in Dzirasa et al. (2010)), we typically detect between 4-50 neurons for each of the \( M \) regions in which the microwires are inserted (discussed further when presenting results). The analysis objective is to examine the degree to which one may relate (predict) the LFP data from one brain region using the \( J \)-neuron spiking data from all brain regions. This analysis allows the identification of multi-site neural networks through the examination of the degree to which neurons in one region are predictive of LFPs in another.

Let \( \mathbf{x} \in \mathbb{R}^T \) represent a time series of LFP data measured from a particular brain region. The \( T \) samples are recorded on a regular grid, with temporal interval \( \Delta \). The spike trains from \( J \) different neurons (after sorting) are represented by the set of vectors \( \{ \mathbf{y}_1, \ldots, \mathbf{y}_J \} \), binned in the same manner temporally as the LFP data. Each \( \mathbf{y}_j \in \mathbb{Z}^T_+ \) is reflective of the number of times neuron \( j \in \{1, \ldots, J\} \) fired within each of the \( T \) time bins, where \( \mathbb{Z}_+ \) represents nonnegative integers.

In the proposed model LFP data \( \mathbf{x} \) are represented as a superposition of signals associated with each neuron \( \mathbf{y}_j \), plus a residual that captures LFP signal unrelated to the spiking data. The contribution to \( \mathbf{x} \) from information in \( \mathbf{y}_j \) is assumed generated by the convolution of \( \mathbf{y}_j \) with a bounded-time dictionary element \( \mathbf{d}_j \) (residing within the interval \(-L \) to \( L \), with \( L \ll T \)). This model is related to convolutional dictio-
nary learning Chen et al. (2011a), where the observed (after spike sorting) signal $y_j$ represents the signal we convolve the learned dictionary $d_j$ against.

We model $d_j$ as time evolving, motivated by the expectation that neuron $j$ may contribute differently to specified LFP data, based upon the latent state of the brain (which will be related to observed animal activity). The time series $x$ is binned into a set of $B$ equal-size contiguous windows, where $x = \text{vec}([x_1, \ldots, x_B])$, and likewise $y = \text{vec}([y_{j1}, \ldots, y_{jB}])$. The dictionary element for neuron $j$ is similarly binned as $\{d_{j1}, \ldots, d_{jB}\}$, and the contribution of neuron $j$ to $x_b$ is represented as a convolution of $d_{jb}$ and $y_{jb}$. This bin size is a trade-off between how finely time is discretized and the computational costs.

In the experiments, in one example the bins are chosen to be 30 seconds wide (novel-environment data) and in the other 1 minute (sleep-cycle data), and these are principally chosen for computational convenience (the second data set is nine times longer). Similar results were found with windows as narrow as 10 second, or as wide as 2 minutes.

6.2.2 Modeling the LFP contribution of multiple neurons jointly

Given $\{y_1, \ldots, y_J\}$, the LFP voltage at time window $b$ is represented as

$$x_b = \sum_{j=1}^{J} y_{jb} * d_{jb} + \epsilon_b$$

(6.1)

where $*$ represents the convolution operator. Let $D_j = [d_{j1}, \ldots, d_{jB}] \in \mathbb{R}^{(2L+1) \times B}$ represent the sequence of dictionary elements used to represent the LFP data over the $B$ windows, from the perspective of neuron $j$. We impose the clustering prior

$$D_j = \zeta_j A_j, \quad A_j \sim G, \quad G \sim \text{DP}(\beta, G_0)$$

(6.2)

where $G$ is a draw from a Dirichlet process (DP) Ishwaran and James (2001a); Ferguson (1973a), with scale parameter $\beta > 0$ and base probability measure $G_0$. 

116
Note that we cluster the *shape* of the dictionary elements, and each neuron has its own scaling \( \zeta \in \mathbb{R} \). Concerning the base measure, we impose an autoregressive prior on the temporal dynamics, and therefore \( G_0 \) is defined by an AR(\( \alpha, \gamma \)) process

\[
\mathbf{a}_b = \alpha \mathbf{a}_{b-1} + \mathbf{\nu}_t, \quad \mathbf{\nu}_t \sim \mathcal{N}(0, \gamma^{-1} \mathbf{I})
\]

where \( \mathbf{I} \) is the identity matrix. This AR prior is used to constitute the \( B \) columns of the DP "atoms" \( \mathbf{A}_h = (\mathbf{a}_{h1}, \ldots, \mathbf{a}_{hB}) \), with \( G = \sum_{k=1}^{\infty} \pi_k \delta_{\mathbf{A}_k} \). The elements of the vector \( \pi = (\pi_1, \pi_2, \ldots) \) are drawn from the "stick-breaking" Ishwaran and James (2001a) process \( \pi_h = V_h \prod_{i=h}^{1} (1 - V_i) \) with \( V_h \sim \text{Beta}(1, \beta) \). We place the prior \( \text{Gamma}(a_{\beta}, b_{\beta}) \) on \( \beta \), and priors Uniform(0,1) and \( \text{Gamma}(a_{\gamma}, b_{\gamma}) \) respectively on \( \alpha \) and \( \gamma \). To complete the model, we place the prior \( \mathcal{N}(0, \tau^{-1} \mathbf{I}) \) on \( \mathbf{e}_b \), and \( \zeta_j \sim \mathcal{N}(0, 1) \).

In the implementation, a truncated stick-breaking representation is employed for \( G \), using \( K \) “sticks” \( (V_K = 1) \), which simplifies the implementation and has been shown to be effective in practice Ishwaran and James (2001a) if \( K \) is made large enough, and the size of \( K \) is inferred during the inference algorithm.

Special cases of this model are clear. For example, if the \( \mathbf{A}_j \) are simply drawn i.i.d. from \( G_0 \), rather than from the DP, each neuron is allowed to contribute its own unique dictionary shape to represent \( \mathbf{x}_b \), called a *non-clustering* model in the results. In Rasch et al. (2009) the authors considered a similar model, but the time evolution of \( \mathbf{d}_j \) was not considered (each neuron was assumed to contribute in the same way to represent the LFP, independent of time). Further, in Rasch et al. (2009) only a single neuron was considered, and therefore no clustering was considered. A multi-neuron version of this model is inferred by setting \( B = 1 \).

### 6.3 Inference

#### 6.3.1 Mean-field Variational Inference

Letting \( \Theta = \{ \mathbf{z}, \zeta, \mathbf{A}_{1..K}, V_{1..K}, \beta, \alpha, \gamma \} \), the full likelihood of the clustering model...
The non-clustering model can be recovered by setting $z_j = \delta_j$ and the truncation level in the stick-breaking process $K$ to $J$. The time-invariant model is recovered by setting the number of bins $B$ to 1, with or without clustering. The model of Rasch et al. (2009) is recovered by using a single bin and a single neuron.

Many recent methods Hughes and Sudderth (2013); Heller and Ghahramani (2005) have been proposed to provide quick approximations to the Dirichlet process mixture model. Critically, in these models the latent assignment variables are conditionally independent when the DP parameters are given. However, in the proposed model this assumption does not hold because the observation $x$ is the superposition of the convolved draws from the Dirichlet process.

A factorized variational distribution $q$ is proposed to approximate the posterior distribution, and the non-clustering model arises as a special case of the clustering model. The inference to fit the distribution $q$ is based on Bayesian Hierarchical Clustering Heller and Ghahramani (2005) and the VB Dirichlet Process Split-Merge method Hughes and Sudderth (2013). The proposed model does not fit in either of these frameworks, so a method to learn $K$ by merging clusters by adapting Heller and Ghahramani (2005); Hughes and Sudderth (2013) is presented in Section 6.3.1. The factorized distribution $q$ takes the form:

$$q(\Theta) = \prod_j \left[ q(z_j) \prod_k q(\zeta_{jk}) \right] q(\beta)q(\alpha)q(\gamma) \prod_k \left[ q(A^*_k)q(V_k) \right]$$

(6.5)
Standard forms on these distributions are assumed, with

\[ q(z_j) = \text{Categorical}(r_j) \]

\[ q(\gamma) = \Gamma(a_{\gamma}', b_{\gamma}') \]

\[ q(\alpha) = \mathcal{N}(0, 1)(\hat{\alpha}, \eta^{-1}_\alpha) \]

\[ q(A_k) = \mathcal{N}(%vec(A_k); \text{vec}(\hat{a}_{k1}, \ldots, \hat{a}_{kB}), \Lambda_k^{-1}) \]

\[ \Sigma_k = \Lambda_k^{-1} \]

\[ q(\beta) = \Gamma(a_{\beta}', b_{\beta}') \]

. To facilitate inference, the distribution on \( \zeta_j \) is split into \( q(\zeta_{jk}) = \mathcal{N}(\mu_{jk}, \eta_{jk}^{-1}) \), the variational distribution for \( \zeta \) on the \( j^{th} \) spike train given that it is in cluster \( k \). The non-clustering model can be represented as a special case of the clustering model where \( q(\zeta_{jk}) = \delta_1 \), and \( q(z_j) = \delta_j \). As noted in Hughes and Sudderth (2013), this factorized posterior has the property that a \( q \) with \( K' \) clusters is nested in a representation of \( q \) for \( K \) clusters for \( K \geq K' \), so any number of clusters up to \( K' \) is represented.

Variational algorithms find a \( q \) that minimizes the KL divergence from the true, intractable posterior Blei and Jordan (2006a), finding a \( q \) that locally maximizes the evidence lower bound (ELBO) objective:

\[
\log p(x|\Theta) \geq \mathcal{L}(q) = \mathbb{E}_q[\log p(x, z, \zeta, A^*_{1,\ldots,K}, \beta, \alpha, \gamma|\Theta) - \log q(z, \zeta, A^*_{1,\ldots,K}, \beta, \alpha, \gamma)]
\]

(6.6)

To facilitate inference, approximations to \( p(y|\Theta) \) are developed. Let \( T_b \) be the number of time points in bin \( b \), and define \( R_{jib} \in \mathbb{R}^{(2L+1)\times(2L+1)} \) with entries \( R_{jib,ik} = \frac{1}{T_b} \sum_{t=1}^{T_b} y_{jib,t} y_{ib,t+k-i} \). Let \( y_{jib,t} \) is \( y_j \) at time point \( t \) in window/bin \( b \). Let \( x_{b}^{-j} = x_b - \sum_{j' \neq j} y_{b} \ast (\sum_k r_{jk} \mu_{jk} \hat{a}_{kb}) \), or the residual after all but the contribution from the \( j^{th} \) neuron have been removed, and define let \( \nu_{j_{b}}^{-j} \in \mathbb{R}^{2L+1} \) with entries \( \nu_{j_{b}}^{-j} \).
\[
\frac{1}{T_b} \sum_{t=1}^{T_b} y_{jb,t} x_{b,t+i} \text{ for } i \in \{-L, \ldots, L\}. \]
Both \( R_{jb} \) and \( \nu_{jb} \) can be efficiently estimated with the FFT. For each time bin \( b \), we can write:
\[
\log p(x_{b}^{-j} | y_{jb}, d_{jb}) = \text{const} - \frac{\tau}{2} (x_{b,j}^{-j} - \sum_{t=-L}^{L} y_{jb,t+t} d_{jb,t+t})^2 \approx \text{const} - \frac{\tau T_b}{2} (d_{jb}^T R_{jb} d - 2 (\nu_{jb}^{-j})^T d_{jb})
\]

To define the key updates, let \( y'_b = \sum_j r_{jk} \mu_{jk} y_{jb} \), and \( x_{b}^{-k} = x_b - \sum_{j' \neq j} y'_{k'} \hat{a}_{kb} \). \( \Sigma_{kbb'} \) denotes the block in \( \Sigma_k \) indexing the \( b \) and the \( b' \) bins, which is efficiently calculated because \( \Sigma_k^{-1} \) is a block tri-diagonal matrix from the first-order autoregressive process, and explicit equations exist. Letting \( \hat{N}_k = \sum_{j} r_{jk} \), then \( q(V_k) \) is updated by are \( a_k = 1 + \hat{N}_k, \ b_k = \hat{\beta} + \sum_{k'=k+1}^{K} \hat{N}_{k'} \). For \( q(\zeta_{jk}) \), the parameters are updated
\[
\eta_{jk} = 1 + \sum_{b} \text{trace}(R_{jb}(\hat{a}_{kb} \hat{a}_{kb}^T + \Sigma_{kbb})) \quad \text{and} \quad \mu_{jk} = \eta_{jk}^{-1} \sum_{b} \hat{a}_{kb}^T R_{jb} \nu_{jb}^{-j}.
\]
The clustering latent variables are updated sequentially by:
\[
\log(r_{jk}) \propto -\frac{\tau}{2} \sum_{b} (\mu_{jk} + \eta_{jk}^{-1}) \text{tr}(R_{jb}(T_b \Sigma_{kbb} + \hat{a}_{kb} \hat{a}_{kb}^T)) - 2 \mu_{jk} (x_{b,j}^{-j})^T (y_b R_{jb} \hat{a}_{kb})) + \mathbb{E}_q[\pi(\pi)]
\]
\( x_{b}^{-k} \) and \( y_{b}^{-k} \) can be used to calculate \( q(A_k^+) \). The mean of the distribution \( q(A_k) \) is evaluated using the forward filtering-backward smoothing algorithm, and \( \Sigma_k^{-1} \) is a block tridiagonal matrix, enabling efficient computations. Further details on updating \( q(A_k^+) \) are found in Section E.1 of the Supplemental Material. Approximating distributions \( q(\beta) \), \( q(\alpha) \) and \( q(\gamma) \) are standard Blei and Jordan (2006a); Roberts and Penny (2002).

**Merge steps**

The model is initialized to \( K = J \) clusters and the algorithm first finds \( q \) for the non-clustering model. This initialization is important because of the superposition measurement model. The algorithm proceeds to merge down to \( K' \), where \( K' \) is a local mode of the VB algorithm. The procedure is as follows: (i) Randomly choose two clusters \( k \) and \( k' \) to merge. (ii) Propose a new variational distribution \( \tilde{q} \) with \( K - 1 \) clusters. (iii) Calculate the change in the variational lower bound, \( \mathcal{L}(\tilde{q}) - \mathcal{L}(q) \),
and accept the merge if the variational lower bound increases. As in Hughes and Sudderth (2013), intelligent sampling of $k$ and $k'$ significantly improves performance. Here, we sample $k$ and $k'$ with weight proportional to $\exp(-\mathcal{K}(A_k, A_{k'}; c_0))$, where $\mathcal{K}(\cdot, \cdot; c_0)$ is the radial basis function. In Heller and Ghahramani (2005) all pairwise clusterings were considered, but that is computationally infeasible in this problem. This approach for merging clusters is similar to that developed in Hughes and Sudderth (2013).

This algorithm requires efficient estimation of the difference in the lower bound. For a proposed $k$ and $k'$, a new variational distribution $\tilde{q}$ is proposed, with $\tilde{q}(z_j = k) = q(z_j = k) + q(z_j = k')$ and $\tilde{q}(z_j = k') = 0$, $\tilde{q}(\beta_k) = \text{Beta}(a_0 + \hat{N}_k + \hat{N}_{k'}, b_0 + \sum_{k'^* = k+1}^{K, k'^* \neq k'} \hat{N}_{k'^*})$, $q(\beta_{k'}) = \delta_0$, and $q(A_k)$ is calculated. Letting $H(q) = -\sum_j \sum_k r_{jk} \log r_{jk}$, the difference in the lower bound can be calculated:

$$
\mathcal{L}(\tilde{q}) - \mathcal{L}(q) = \mathbb{E}_q \left[ \log p(y|A_1, \ldots, K, \zeta, \tau) \frac{p(A_k|\alpha, \gamma) p(\beta_k)}{\tilde{q}(A_k) q(\beta_k)} \right] - H(\tilde{q}) + H(\tilde{p}) - \mathbb{E}_q \left[ \log p(y|A_1, \ldots, K, \zeta, \tau) \frac{p(A_k|\alpha, \gamma) p(A_{k'}|\alpha, \gamma) p(\beta_k) p(\beta_{k'})}{q(A_k) q(A_{k'}) q(\beta_k) q(\beta_{k'})} \right] + H(q) - H(p)
$$

Explicit details on the calculations of these variables are found in Section E.1 of the Supplementary Material, and the block tridiagonal nature of $\Lambda_k$ allows the complete calculation of this value in $O(BT_b((\hat{N}_k + \hat{N}_{k'}) + L^3))$. This is linear in the amount of data used in the model. The algorithm is stopped after 10 merges in a row are rejected.

6.3.2 Integrated Nested Laplacian Approximation for the Non-Clustering Model

The VB inference method assumes a separable posterior. In the non-clustering model, Integrated Nested Laplacian Approximation (INLA) Rue et al. (2009) was used to estimate of the joint posterior, without assuming separability. Comparisons to INLA
Table 6.1: Mean held-out RFE of the multi-cell models predicting the Hippocampus LFP. “Invariant” denotes the time-invariant model, “Non-cluster” and “clustering” denote the dynamic model without and with clustering.

<table>
<thead>
<tr>
<th>Animal</th>
<th>Invariant</th>
<th>Non-Cluster</th>
<th>Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1394</td>
<td>0.1968</td>
<td>0.2094</td>
</tr>
<tr>
<td>2</td>
<td>0.1465</td>
<td>0.2382</td>
<td>0.2340</td>
</tr>
<tr>
<td>3</td>
<td>0.2251</td>
<td>0.3050</td>
<td>0.3414</td>
</tr>
<tr>
<td>4</td>
<td>0.0867</td>
<td>0.1433</td>
<td>0.1434</td>
</tr>
<tr>
<td>5</td>
<td>0.1238</td>
<td>0.1867</td>
<td>0.1882</td>
</tr>
<tr>
<td>6</td>
<td>0.0675</td>
<td>0.1407</td>
<td>0.1351</td>
</tr>
<tr>
<td>7</td>
<td>0.1385</td>
<td>0.2567</td>
<td>0.2442</td>
</tr>
<tr>
<td>8</td>
<td>0.0902</td>
<td>0.3440</td>
<td>0.3182</td>
</tr>
<tr>
<td>9</td>
<td>0.1597</td>
<td>0.1881</td>
<td>0.2362</td>
</tr>
<tr>
<td>10</td>
<td>0.0311</td>
<td>0.0803</td>
<td>0.0865</td>
</tr>
<tr>
<td>11</td>
<td>0.675</td>
<td>0.1064</td>
<td>0.1161</td>
</tr>
</tbody>
</table>

constitute an independent validation of VB, for inference in the non-clustering version of the model. The INLA inference procedure is detailed in Supplemental Section E.2. INLA inference was found to be significantly slower than the VB approximation, so experimental results below are shown for VB. The INLA and VB predictive performance were quantitatively similar for the non-clustering model, providing confidence in the VB results.

6.4 Experiments

6.4.1 Results on Mice Introduced to a Novel Environment

This data set is from a group of 12 mice consisting of male Clock-Δ19 (mouse numbers 7-12) and male wild-type littermate controls (mouse numbers 1-6) (further described in Dzirasa et al. (2010)). For each animal, 32-48 total microwires were implanted, with 6-16 wires in each of the Nucleus Accumbens, Hippocampus (HP), Prelimbic Cortex (PrL), Thalamus, and the Ventral Tegmental Area (VTA). LFPs were averaged over all electrodes in an area and filtered from 3-50Hz and sampled at 125 Hz. Neuronal activity was recorded using a Multi-Neuron Acquisition Processor
Figure 6.1: (Left) Mean single-cell holdout RFE predicting mouse 3’s Nucleus Accumbens LFP comparing the dynamic and time-invariant model. Each point is a single neuron. (Middle) Convolutional dictionary for a VTA cell predicting mouse 3’s Nucleus Accumbens LFP at 5 minutes, 15 minutes, and 38 minutes after the experiment start. (Right) Hold-out RFE over experiment time with the time-invariant, non-clustering, and the clustering model to predict mouse 3’s Hippocampus LFP.

Figure 6.2: Example clusters predicting mouse 3’s Hippocampus LFP. The top part shows the convolutional factor throughout the duration of the experiment, and the bottom part shows the location of the cells in the cluster. Some of the clusters are dynamic whereas others were consistent through the duration of the experiment.

(Plexon). 99-192 individual spike trains (single units) were detected per animal. In this dataset animals begin in their home cage, and after 10 minutes are placed in a novel environment for 30 minutes. For analysis, this 40 minute data sequence was binned into 30 second chunks, giving 80 bins. For all experiments we choose $L$ such that the dictionary element covered 0.5 seconds before and after each spike event.

Cross-validation was performed using leave-one-out analysis over time bins, using the error metric of reduction in fractional error (RFE), $1 - ||x_b - \hat{x}_b||^2_2/||x_b||^2_2$. Figure 6.1(left) shows the average hold-out RFE for the time-invariant model and the dynamic model for single spike train predicting mouse 3’s Nucleus Accumbens, showing
that the dynamic model can give strong improvements on the scale of a single cell
(these results are typical). The dynamic model has a higher hold-out RFE on 98.4% of
detected cells across all animals and all regions, indicating that the dynamic model
generally outperforms the time-invariant model. A dynamic dictionary element from
a VTA cell predicting mouse 3’s Nucleus Accumbens is shown in Figure 6.1(middle).
At the beginning of the experiment, this cell is linked with a slow, high-amplitude
oscillation. After the animal is initially placed into a new environment (illustrated
by the 15-minute data point), the amplitude of the dictionary element drops close
to zero. Once the animal becomes accustomed to its new environment (illustrated
by the 38-minute data point), the cell’s original periodic dictionary element begins
to appear again. This example shows how cells and LFPs clearly have time-evolving
relationships.

The leave-one-out performance of the time-invariant, non-clustering, and cluster-
ing models predicting animal 3’s Hippocampus LFP with 182 neurons is shown in
Figure 6.1(right). These results show that predictability changes over time, and indi-
cate that there is a strong increase in LFP predictability when the mouse is placed in
the novel environment. Using dynamics improves the results dramatically, and the
clustering hold-out results showed further improvements in hold-out performance.
The mean hold-out RFE results for the Hippocampus for 11 animals are shown in Table 6.1 (1 animal was missing this region recording). Results for other regions are shown in Supplemental Tables E.1, E.2, E.3, and E.4, and show similar results.

In this dataset, there is little quantitative difference between the clustering and non-clustering models; however, the clustering result is much better for interpretation. One reason for this is that spike-sorting procedures are notoriously imprecise, and often under- or over-cluster. A clustering model with equivalent performance is evidence that many neurons have the same shapes and dynamics, and repeated dynamic patterns reduces concerns that dynamics are the result of failure to distinguish distinct neurons. Similarly, clustering of neuron shapes in a single electrode could be the result of over-clustering from the spike-sorting algorithm, but clustering across electrodes gives strong evidence that truly different neurons are clustering together. Additionally, neural action potential shapes drift over time Calabrese and Paniski (2010); Carlson et al. (2013b), but since cells in a cluster come from different electrodes and regions, this is strong evidence that the dynamics are not due to over-sorting drifting neurons.

Each cluster has both a dynamic shape result as well as well as a neural distribution over regions. Example clustering shapes and histogram cell locations for clusters predicting mouse 3’s Thalamus LFP are shown in Figure 6.2. The top part of this figure shows the base dictionary element evolution over the duration of the experiment. Note that both the (left) and (middle) plots show a dynamic effect around 10 minutes, and the cells primarily come from the Ventral Tegmental Area. The (right) plot shows a fairly stable factor, and its cells are mostly in the Hippocampus region.

The ability to predict the LFP constitutes functional connectivity between a neuron and the neuronal circuit around the electrode for the LFP Nauhaus et al. (2009). Neural circuits have been shown to transfer information through specific frequencies of oscillations, so it is of scientific interest to know the functional connectivity of
Table 6.2: Mean held-out RFE of the animal going through sleep cycles in each region.

<table>
<thead>
<tr>
<th>Region</th>
<th>PrLCx</th>
<th>MOFCCx</th>
<th>NAcShell</th>
<th>NAcCore</th>
<th>Amyg</th>
<th>Hipp</th>
<th>V1</th>
<th>VTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-Invariant</td>
<td>0.1055</td>
<td>0.1304</td>
<td>0.0904</td>
<td>0.1076</td>
<td>0.0883</td>
<td>0.2091</td>
<td>0.1366</td>
<td>0.1317</td>
</tr>
<tr>
<td>Non-Clustering</td>
<td>0.1686</td>
<td>0.1994</td>
<td>0.1599</td>
<td>0.1796</td>
<td>0.1422</td>
<td>0.2662</td>
<td>0.1972</td>
<td>0.1907</td>
</tr>
<tr>
<td>Clustering</td>
<td>0.1749</td>
<td>0.2029</td>
<td>0.1609</td>
<td>0.1814</td>
<td>0.1390</td>
<td>0.2798</td>
<td>0.2020</td>
<td>0.1923</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
<th>Subnigra</th>
<th>Thal</th>
<th>LHb</th>
<th>DLS</th>
<th>DMS</th>
<th>M1</th>
<th>OFC</th>
<th>FrA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-Invariant</td>
<td>0.1309</td>
<td>0.1550</td>
<td>0.1240</td>
<td>0.1237</td>
<td>0.1518</td>
<td>0.1350</td>
<td>0.1878</td>
<td>0.1164</td>
</tr>
<tr>
<td>Non-Clustering</td>
<td>0.1939</td>
<td>0.2188</td>
<td>0.1801</td>
<td>0.1973</td>
<td>0.2363</td>
<td>0.2034</td>
<td>0.2695</td>
<td>0.1894</td>
</tr>
<tr>
<td>Clustering</td>
<td>0.1950</td>
<td>0.2204</td>
<td>0.1813</td>
<td>0.2012</td>
<td>0.2378</td>
<td>0.2080</td>
<td>0.2723</td>
<td>0.1912</td>
</tr>
</tbody>
</table>

a group of neurons as a function of frequency Le Van Quyen and Bragin (2007).

Frequency relationships were explored by filtering the LFP signal after the predicted signal has been removed, using a notch filter at 1 Hz intervals with a 1 Hz bandwidth, and the RFE was calculated for each held-out time bin and frequency bin.

All cells in the Thalamus were used to predict each frequency band in mouse 3’s Hippocampus LFP, and this result is shown in Figure 6.3(left). This figure shows an increase in RFE of the 25-35 Hz band after the animal has been moved to a new location. The RFE on the band from 25-35 Hz is shown in Figure 6.3(middle), and shows that while the raw energy in this frequency band is much higher after the move to the novel environment, the cells from the Hippocampus can explain much of the additional energy in this band. In Figure 6.3(right), we show the same result using only the cluster in Figure 6.2. Note that there is a change around 10 minutes that is due to both a slight change in the convolutional dictionary and a change in the neural firing patterns.

6.4.2 Results on Sleep Data Set

The second data set was recorded from one mouse going through different sleep cycles over 6 hours. 64 microwires were implanted in 16 different regions of the brain, using the Prelimbix Cortex (PrL), Medial Orbital Frontal Cortex (MOFCCx), the core and shell of the Nucleus Accumbens (NAc), Basal Amygdala (Amy), Hippocampus...
Figure 6.4: The predictive patterns of individual neurons predicting multiple regions. (Left) A Hippocampus cell is the best single cell predictor of the V1 LFP (Middle) A V1 cell with a relationship only to the V1 LFP. (Right) A Nucleus Accumbens Shell cell that is equivalent in predictive ability to the best V1 cell.

Figure 6.5: (Left) The cluster predicting the V1 region of the brain, matching known pattern for individual V1 cells Rasch et al. (2009); Nauhaus et al. (2009). (Middle,Right) Clusters predicting the motor cortex that show positive (pro) and negative (anti) relationships between amplitude and sleep.

(Hipp), V1, Ventral Tegmental Area (VTA), Substantia nigra (Subnigra), Medial Dorsal Thalamus (MDThal), Lateral Habenula (LHb), Dorsolateral Striatum (DLS), Dorsomedial Striatum (DMS), Motor Cortex (M1), Orbital Frontal Cortex (OFC), and Frontal Association Cortex (FrA). LFPs were averaged over all electrodes in an area and filtered from 3-50Hz and sampled at 125Hz, and $L$ was set to 0.5 seconds. 163 total neurons (single units) were detected using spike sorting, and the data were split into 360 1-minute time bins. The leave-one-out predictive performance was higher for the dynamic single cell model on 159 out of 163 neurons predicting the Hippocampus LFP. The mean hold-out RFEs for all recorded regions of the brain are
Figure 6.6: Mean RFE when the animal is awake and when it is asleep. (Left) Cluster’s convolution factor is stable, and shows only minor differences between sleep and awake prediction. (Middle and Right) Clusters shown in Figure 6.5 (left and right), depicting varying patterns with the mouse’s sleep state shown in Table 6.2 for all models, and the clustering model is the best performing model in 15 of the 16 regions.

Previously published work looked at the predictability of the V1 LFP signal from individual V1 neurons Kelly et al. (2010); Rasch et al. (2009); Nauhaus et al. (2009). Our experiments find that the dictionary elements for all V1 cells (4 electrodes, 4 cells in this dataset) are time-invariant and match the single-cell time-invariant dictionary shape of Rasch et al. (2009). The dictionary elements for a single V1 cell predicting multiple regions are shown in Figure 6.4(middle; for simplicity, only a subset of brain regions recorded from are shown). This suggests that the V1 cell has a connection to the V1 region, but no other brain region that was recorded from in this model. However, cells in other brain regions showed functional connectivity to V1. The best individual predictor is a cell in the Hippocampus shown in Figure 6.4(left). An additional example cell is a cell in the Nucleus Accumbens shell that has the same RFE as the best V1 cell, and its shape is shown in Figure 6.4(right).

Sleep states are typically defined by dynamic changes in functional connectivity across brain regions as measured by EEG (LFPs recorded from the scalp) Larson-Prior et al. (2009), but little is known about how single neurons contribute to, or interact with, these network changes. To get sleep covariates, each second of data
was scored into “awake” or “sleep” states using the methods in Dzirasa et al. (2006), and the sleep state was averaged over the time bin. We defined a time bin to be a sleep state if $\geq 95\%$ of the individual seconds are scored as a sleep state, and the animal is awake if $\leq 5\%$ of the individual seconds are scored as a sleep state. In Figure 6.5(middle) we show a cluster that is most strongly positively correlated with sleep (pro-sleep), and in Figure 6.5(right) we show a cluster that is most negatively correlated with sleep (pro-awake). Both figures show the neuron locations as well as the mean waveform shape during sleep and wake. In this case, the pro-sleep cluster is dominantly Hippocampus cells and the anti-sleep cluster comes from many different regions. There may be concern that because these are the maximally correlated clusters, that these results may be atypical. To address this concern, the p-value for finding a cluster this strongly correlated has a p-value $4 \times 10^{-6}$ for Pearson correlation with the Bonferroni correction for multiple tests. Furthermore, 4 of the 25 clusters detected showed correlation above .4 between amplitude and sleep state, so this is not an isolated phenomena.

The RFE changes as both a function of frequency and sleep state for some clusters of neurons. Using 1Hz bandwidth frequency bins, in Figure 6.6 (middle and right) we show the mean RFE using only the clusters in Figure 6.5 (middle and right). The cluster associated positively with sleeping shifts its frequency peak and increases its ability to predict when the animal is sleeping. Likewise, the sleep-decreased cluster performs worst at predicting when the animal is asleep. For comparison, in Figure 6.6 (left) we include the frequency results for cluster with a stable dictionary element. The total RFE is comparable and there is not a dramatic shift in the peak frequency between the sleep and awake states.
6.5 Conclusions

Novel models and methods are developed here to account for time-varying relationships between neurons and LFPs. Within the context of our experiments, significantly improved predictive performance is realized when one accounts for temporal dynamics in the neuron-LFP interrelationship. Further, the clustering model reveals which neurons have similar relationships to a specific brain region, and the frequencies that are predictable in the LFP change with known dynamics of the animal state. In future work, these ideas can be incorporated with attempts to learn network structure, and LFPs can be considered a common input when exploring networks of neurons Pillow and Latham (2007); Kelly et al. (2010); Rasch et al. (2008). Moreover, future experiments are being designed to place additional electrodes in a single brain region, with the goal of detecting 100 neurons in a single brain region while recording LFPs in up to 20 regions. The methods proposed here will facilitate exploration of both the diversity of neurons and the differences in functional connectivity on an individual neuron scale.
Appendix A

Proofs and Supplemental Material of Stochastic Spectral Descent for Restricted Boltzmann Machines

A.1 RBM Theorem proofs

A.1.1 Proof of Theorem 1

Proof. Proof of Theorem 1.

The Hessian of the lse function is given by

\[ \nabla^2 \text{lse}_\omega(u) = \frac{\text{diag}(\omega \odot \exp(u))}{\omega^T \exp(u)} \]

\[ \quad - \frac{(\omega \odot \exp(u))(\omega \odot \exp(u))^T}{(\omega^T \exp(u))^2} \] \hspace{1cm} (A.1)

There are two terms in the Hessian matrix. The first term is

\[ \frac{\text{diag}(\omega \odot \exp(u))}{\omega^T \exp(u)} \]
This is a diagonal matrix where the diagonal entries are nonnegative and sum to one.

The second term is

\[- \frac{(\mathbf{\omega} \odot \exp(\mathbf{u}))(\mathbf{\omega} \odot \exp(\mathbf{u}))^T}{(\mathbf{\omega}^T \exp(\mathbf{u}))^2}\]

This term is a rank-one matrix with a negative eigenvalue.

Writing Taylor’s theorem:

\[lse_{\mathbf{\omega}}(\mathbf{v}) = lse_{\mathbf{\omega}}(\mathbf{u}) + \langle \nabla lse_{\mathbf{\omega}}(\mathbf{u}), \mathbf{v} - \mathbf{u} \rangle + \int_0^1 (1 - t)(\mathbf{v} - \mathbf{u})^T \nabla^2 lse_{\mathbf{\omega}}(\mathbf{u} + t(\mathbf{v} - \mathbf{u}))(\mathbf{v} - \mathbf{u}) dt\]

The terms in the integral can be bound

\[(\mathbf{v} - \mathbf{u})^T \nabla^2 lse_{\mathbf{\omega}}(\mathbf{u} + t(\mathbf{v} - \mathbf{u}))(\mathbf{v} - \mathbf{u})\]

\[\leq (\mathbf{v} - \mathbf{u}) \frac{\text{diag}(\mathbf{\omega} \odot \exp(\mathbf{u} + t(\mathbf{v} - \mathbf{u})))}{\omega^T \exp(\mathbf{u} + t(\mathbf{v} - \mathbf{u}))}(\mathbf{v} - \mathbf{u})\]

\[= \sum_{j=1}^J \frac{\omega_j \exp(\mathbf{u}_j + t(v_j - u_j))}{\omega^T \exp(\mathbf{u} + t(\mathbf{v} - \mathbf{u}))}(v_j - u_j)^2\]

\[\leq \max_{c \geq 0, \sum_{j=1}^J c_j(v_j - u_j)^2} \sum_{j=1}^J c_j(v_j - u_j)^2\]

\[= ||\mathbf{v} - \mathbf{u}||_z^2\]

Eq. A.2 follows because the second term in the Hessian will give a nonpositive value and Eq. A.3 follows because the diagonal entries are nonnegative and sum to 1. The integral has an upper bound of $\frac{1}{2}||\mathbf{v} - \mathbf{u}||_z^2$. ■

A.1.2 Proof of Theorem 2

Proof. Proof of Theorem 2.

The log partition function can be written as a sum over only the hidden units to give a similar form to Theorem 1. Define the set $\{h_i\}_{i=1}^{2^J}$ as the set of unique binary
vectors \( \{0, 1\}^J \), and let \( \mathbf{H} \in \{0, 1\}^{J \times 2^J} \) be the matrix form of this set.

\[
f(\theta) = \log \sum_{i=1}^{2^J} \omega_i \exp(h_i^T \mathbf{b})
\]
(A.5)

\[
\omega_i = \sum_{m=1}^{M} \log(1 + \exp(W_m, h_i + c_m))
\]
(A.6)

Equation A.5 can be equivalently written as

\[
f(\theta) = \log \omega^T \exp(H^T \mathbf{b})
\]
(A.7)

with \( \omega \) not dependent on \( \mathbf{b} \). Plugging into Equation 2.17,

\[
f(\{\mathbf{b}, \mathbf{c}^k, \mathbf{W}^k\}) \leq f(\theta^k) + \langle \nabla_{H^T \mathbf{b}^k \omega} \exp(H^T \mathbf{b}^k), H^T(\mathbf{b} - \mathbf{b}^k) \rangle + \frac{1}{2}\|H^T(\mathbf{b} - \mathbf{b}^k)\|_2^2
\]
(A.8)

To rewrite the inner product term, note that

\[
\nabla_{H^T \mathbf{b}^k \omega} \exp(H^T \mathbf{b}^k) = H^T \nabla_b f(\theta^k)
\]
(A.9)

\[
(\nabla_{H^T \mathbf{b}^k \omega} \exp(H^T \mathbf{b}^k))^T H (\mathbf{b} - \mathbf{b}^k) = (\nabla_b f(\theta^k))^T (\mathbf{b} - \mathbf{b}^k)
\]

The bound is simplified as

\[
\|H^T(\mathbf{b} - \mathbf{b}^k)\|_\infty = \max_i |h_i^T(\mathbf{b} - \mathbf{b}^k)| \leq J\|\mathbf{b} - \mathbf{b}^k\|_\infty
\]

Alternatively, this could be bound as

\[
\|H^T(\mathbf{b} - \mathbf{b}^k)\|_\infty \leq \sqrt{J}\|\mathbf{b} - \mathbf{b}^k\|_2
\]
(A.10)

\[
\|H^T(\mathbf{b} - \mathbf{b}^k)\|_\infty \leq \|\mathbf{b} - \mathbf{b}^k\|_1
\]
(A.11)

The proof on \( \mathbf{c} \) follows with the same techniques.
A.1.3 Proof of Theorem 3

Proof. Proof of Theorem 3.

As in the proof for Theorem 2, let $H \in \{0, 1\}^{J \times 2^J}$ and $V \in \{0, 1\}^{M \times 2^M}$, where each column is an unique binary vector. Define $U = V^TWH$ and $\Omega_{ij} = \mathbf{v}_i^T c + h_j^T b$. Let $u = \text{vec}(U)$ and $\omega = \text{vec}(\Omega)$. The log partition function is equivalently written

\[
\begin{align*}
    f(\theta) &= \log \sum_{i=1}^{2^J} \sum_{j=1}^{2^M} \Omega_{ij} \exp U_{ij} & (A.12) \\
    f(\theta) &= \log (\omega^T \exp u) & (A.13)
\end{align*}
\]

Plugging this form into Equation 2.17:

\[
\begin{align*}
    lse(\omega(u)) & \geq lse(\omega(u^k)) + \langle \nabla_u lse(\omega(u^k)), u - u^k \rangle \\
    & + \frac{1}{2} \| \text{vec}(U - U^k) \|_F^2 & (A.14)
\end{align*}
\]

Note that

\[
\begin{align*}
    \langle \nabla_u lse(\omega(u)), u - u^k \rangle &= \text{tr}((\nabla_u lse(\Omega(U))^T(U - U^k)) \\
    V \nabla_u lse(\Omega(U))H^T &= \nabla_w f(\theta) & (A.15)
\end{align*}
\]

Writing the inner product in terms of $W$ gives

\[
\begin{align*}
    \text{tr}((\nabla_u lse(\Omega(U))^T(U - U^k)) = \text{tr}((\nabla_w)^T(W - W^k)) & (A.16)
\end{align*}
\]

The bound is simplified:

\[
\| \text{vec}(U - U^k) \|_\infty = \max_{i,j} | \mathbf{v}_i^T(W - W^k) h_j | 
\leq \sqrt{MJ} \| W - W^k \|_\infty & (A.17)
\]

Combining these two elements proves Theorem 3. □
A.2 Derivation of optimal steps

Proof. Proof of $b^*$ in Equation 2.25.

We want to find the minimizer of

$$\min_b \langle \nabla_b F(\theta^k), b - b^k \rangle + \frac{J}{2} \|b - b^k\|_{\infty}^2$$

First, add an additional variable $a$ such that the minimizer of the expanded problem is the same as the original problem

$$\min_{b,a|b_j \leq a, a \geq 0} \langle \nabla_b F(\theta^k), b - b^k \rangle + \frac{J}{2} a^2$$

(A.18)

This is straightforward to solve:

$$\begin{align*}
  &= \min_{a, a \geq 0} \langle \nabla_b F(\theta^k), -a \times \text{sign}(\nabla_b F(\theta^k)) \rangle + \frac{J}{2} a^2 \\
  a^* &= \frac{1}{J} \|\nabla_b F(\theta^k)\|_1 \\
  b^* &= b - \frac{1}{J} \|\nabla_b F(\theta^k)\|_1 \times \text{sign}(\nabla_b F(\theta^k))
\end{align*}$$

(A.19) (A.20)

Proof. Proof of $W^*$ in Equation 2.28.

Let $D = W - W^k$, and decompose $D = A RB^T$, with $A$ and $B$ denoting the left and right singular vectors of $\nabla_w F(\theta^k)$. Then we want to minimize the quantity

$$\min_D \text{tr}(\nabla_w F(\theta^k)D) + \frac{MJ}{2} \|D\|_{S^\infty}^2$$

As in the proof on the biases, add an additional variable that will give the same
minimizer and solve for the solution.

\[ \begin{align*}
= & \min_{D, a, \|D\|_{s\infty} < a} \text{tr}(\nabla_w F(\theta^k)D) + \frac{MJ}{2}a^2 \\
= & \min_{D, a, \|D\|_{s\infty} < a} \text{tr}(\nabla_w F(\theta^k)D) + \frac{MJ}{2}a^2 \\
= & \min_{a, F, \|F\|_{s\infty} < a} \lambda^T \text{diag}(R) + \frac{MJ}{2}a^2 
\end{align*} \]

Letting \( I_M \) denote the \( M \)-dimensional identity matrix, this gives:

\[ \begin{align*}
R^* &= \frac{-a}{MJ}I_M \\
a &= \|\lambda\|_1 \\
R^* &= \left( \frac{-1}{MJ} \|\lambda\|_1 \times I_M \right)
\end{align*} \]  

\[ \square \]

A.3 Discussion of using \( \ell_2 \) bound instead of \( \ell_{\infty} \) bound on \textit{lse} function

(Böhning, 1992) introduces a bound on the \textit{lse} function

\[ lse_1(v) \leq lse_1(u) + \langle \nabla_u lse_1(u), v - u \rangle + \frac{1}{2}(v - u)^T B(v - u) \]  

\[ B = \frac{1}{2} \left[ I_J - \frac{1}{J} 1_J 1_J^T \right] \]  

Where \( I \) is the \( J \)-dimensional identity matrix and \( 1_J \) is a \( J \)-dimensional ones vector. This is trivially extended to use a nonnegative vector \( \omega \) in place of \( 1_J \). The quadratic term is equivalently written

\[ \frac{1}{2}(v - u)^T B(v - u) = \frac{1}{4}\|v - u\|^2_2 - \frac{1}{4} \text{mean}(v - u)^2 \]  

136
Because of the differences of logsumexp functions, the mean term drops out and so this bound gives

\[
ls_e \omega(v) \leq lse_\omega(u) + \langle \nabla_u lse_\omega(u), v - u \rangle + \frac{1}{2} \|v - u\|_2^2 \tag{A.27}
\]

Using Equation A.27 instead of Equation 2.17 in the proofs in Supplemental Section A.1 leads to looser bounds due to the high-dimensional nature of the observation space. However, it should be noted that it may be possible to bound this more tightly.

First, examining the bound on the matrix \( W \),

\[
\frac{1}{4} \| \text{vec}(U - U^k) \|_2^2 \tag{A.28}
\]

\[
= \frac{1}{4} \sum_{i=1}^{2^M} \sum_{j=1}^{2^J} (v_i^T (W - W^k) u_j)^2 \tag{A.29}
\]

\[
\leq \frac{1}{4} \sum_{i=1}^{2^M} \sum_{j=1}^{2^J} v_i^T ((W - W^k) \odot (W - W^k)) u_j \tag{A.30}
\]

\[
= \frac{1}{4} \text{tr}(((W - W^k) \odot (W - W^k)) \sum_{i=1}^{2^M} \sum_{j=1}^{2^J} h_j v_i^T) \]

\[
= \frac{1}{4} \text{tr}(((W - W^k) \odot (W - W^k))(\frac{2^{M+J}}{4} 1_{J \times M})) \]

\[
= \frac{2^{M+J}}{16} \| W - W \|_F \tag{A.31}
\]

For realistic problems sizes of RBMs, the bound that comes out of the logsumexp \( \infty \)-norm bound is exponentially tighter than the bound using logsumexp \( \ell_2 \) norm bound.

137
Similar analysis on the bias terms reveals a bounding term equations

\[
f(b, c^k, W^k) \leq f(\theta^k) + \langle \nabla_b f(\theta^k), b - b^k \rangle + \frac{2^J}{8} ||b - b^k||_\infty^2 \tag{A.32}
\]

\[
f(b^k, c, W^k) \leq f(\theta^k) + \langle \nabla_c f(\theta^k), c - c^k \rangle + \frac{2^M}{8} ||c - c^k||_\infty^2 \tag{A.33}
\]
Appendix B

Proofs and Supplemental Material of Descent Algorithms of Deep Discrete Graphical Models

B.1 Proof of Theorem 4

The function on the data term used in Theorem 4 has the form

\[ f(\theta) = -\frac{1}{N} \sum_n \log \sum_h \exp(-E_\theta(v_n, h)) \]

We first consider bounding this function where \( N = 1 \), and then the bound expands trivially to multiple data points. For a single data point, the form is

\[ f(\theta) = -\log \sum_h \exp(-E_\theta(v, h)) \tag{B.1} \]

We want to find a upper bound on the difference between \( f(\phi) - f(\theta) \). To do this, first define the log-sum-exp function as

\[ g_\omega(x) = \log \sum_i \omega_i \exp(x_i), \quad \omega > 0 \tag{B.2} \]

If \( \omega \) is omitted it is considered to be a vector of ones. The log-sum-exp function is a convex function.
Define a vector $\mathbf{x}^{(\theta)} \in \mathbb{R}^{|\mathcal{H}|}$, with each entry $x_i^{(\theta)} = -E_{\theta}(\mathbf{v}, \mathbf{h}_i)$, $\mathbf{h}_i \in \mathcal{H}$. Note that $g(x_i^{(\theta)}) = -f(\theta)$, and so a lower bound on $g(\mathbf{x})$ will correspond to an upper bound on $f(\theta)$. Define a second vector $\mathbf{x}^{(\phi)}$ with entries $x_i^{(\phi)} = -E_{\phi}(\mathbf{v}, \mathbf{h}_i)$. Using the convexity of the log-sum-exp function, we can write a bound on the difference between $f(\phi) - f(\theta)$, as

$$g(x^{(\phi)}) \geq g(x^{(\theta)}) + \langle \nabla g(x^{(\theta)}), x^{(\phi)} - x^{(\theta)} \rangle$$

$$-f(\phi) \geq -f(\theta) + \langle \nabla g(x^{(\theta)}), x^{(\phi)} - x^{(\theta)} \rangle$$

$$f(\phi) \leq f(\theta) - \langle \nabla g(x^{(\theta)}), x^{(\phi)} - x^{(\theta)} \rangle$$

(B.3)

The inner product $-\langle \nabla g(x^{(\theta)}), x^{(\phi)} - x^{(\theta)} \rangle$ needs to upper bounded in terms of $\phi$ and $\theta$. First, note that

$$\langle \nabla g(x^{(\theta)}), x^{(\phi)} - x^{(\theta)} \rangle = \sum_i \left[ \frac{d}{dx_i} g(x^{(\theta)}) \right] (x_i^{(\phi)} - x_i^{(\theta)})$$

We want to write this in relation to the inner product

$$\langle \nabla_\theta f(\theta), \phi - \theta \rangle = -\sum_i \left[ \frac{d}{dx_i} g(x^{(\theta)}) \right] \langle \nabla_\theta x_i^{(\theta)} \rangle, \phi - \theta \rangle$$

$$= -\sum_i \left[ \frac{d}{dx_i} g(x^{(\theta)}) \right] \langle \nabla_\theta (-E_{\theta}(\mathbf{v}, \mathbf{h}_i)), \phi - \theta \rangle$$

This form elucidates that we have

$$-\langle \nabla g(x^{(\theta)}), x^{(\phi)} - x^{(\theta)} \rangle = \langle \nabla_\theta f(\theta), \phi - \theta \rangle$$

$$-\sum_i \left[ \frac{d}{dx_i} g(x^{(\theta)}) \right] ( -E_{\phi}(\mathbf{v}, \mathbf{h}_i) + E_{\theta}(\mathbf{v}, \mathbf{h}_i) - \langle \nabla_\theta (-E_{\theta}(\mathbf{v}, \mathbf{h}_i)), \phi - \theta \rangle)$$

(B.4)

By the properties of the log-sum-exp function, the entries of $\nabla g(x^{(\theta)})$ will be non-
negative and sum to 1. Because of this, we can bound this relationship by
\[- \langle \nabla g(x^{(\theta)}), x^{(\phi)} - x^{(\theta)} \rangle \leq \langle \nabla f(\theta), \phi - \theta \rangle \]
\[= \min_{i=1, \ldots, |H|} (-E_\phi(v, h_i) + E_\theta(v, h_i) - \langle \nabla \theta(-E_\theta(v, h_i)), \phi - \theta \rangle) \tag{B.5} \]
Combining the relationships in Equations B.3 and B.5 gives the result in Theorem 4 for a single data point. The extension to multiple data points is a simple application of the unity bound.

### B.2 Proof of Theorem 5

We focus on bounding the function
\[\log Z(\theta) = \log \sum_v \sum_h \exp(-E_\theta(v, h))\]

We use the definition of the log-sum-exp in Equation B.2, and also note the \(\ell_\infty\) norm bound on the log-sum-exp function (Carlson et al., 2015b),
\[g_\omega(y) \leq g_\omega(x) + \langle \nabla g_\omega(x), y - x \rangle + \frac{1}{2} ||y - x||_2 \tag{B.6} \]
In high dimensions, the \(\ell_\infty\) norm will lead to tighter results than the \(\ell_2\) norm of (Böhning, 1992). In problems like the RBM, the bound from the Frobenius norm will be exponentially worse in the number of nodes for large dimensions.

Define a vector \(x^{(\theta)} \in \mathbb{R}^{|V \times H|}\), where each entry corresponds to the energy of a unique pair of visible and hidden nodes, \(x_i^{(\theta)} = -E_\theta(v_i, h_i), (v_i, h_i) \in V \times H\). Then \(g(x^{(\theta)}) = \log Z(\theta)\).

We use the same relationship as Section B.1 to give
\[g(x^{(\phi)}) \geq g(x^{(\theta)}) + \langle \nabla g(x^{(\theta)}), x^{(\phi)} - x^{(\theta)} \rangle + \frac{1}{2} ||x^{(\phi)} - x^{(\theta)}||_\infty \]
\[\log Z(\phi) \leq \log Z(\theta) - \langle \nabla g(x^{(\theta)}), x^{(\phi)} - x^{(\theta)} \rangle + \frac{1}{2} ||x^{(\phi)} - x^{(\theta)}||_\infty \tag{B.7} \]
We have the upper bound on the inner product, as in Equation B.5, with

\[
\langle \nabla g(x^{(\theta)}), x^{(\phi)} - x^{(\theta)} \rangle \\
\leq \langle \nabla \theta \log Z(\theta), \phi - \theta \rangle \\
+ \max_{v \in V, h \in H} (-E_{\phi}(v, h) + E_{\theta}(v, h) + \langle \nabla \theta (-E_{\theta}(v, h)), \phi - \theta \rangle) \tag{B.8}
\]

Equation B.8 is derived in the same way as Equation B.5. Combining Equation B.7 with Equation B.8 completes the proof of Theorem 5.

B.3 Proof of Equation 3.4

Equation 3.4 is the special case of Theorem 5 when the energy function is linear. Theorem 5 states that

\[
\log Z(\phi) \leq \log Z(\theta) + \langle \nabla \theta \log Z(\theta), \phi - \theta \rangle \\
+ \max_{v, h} (-E_{\phi}(v, h) + E_{\theta}(v, h) - \langle \nabla \theta (-E(v, h)), \phi - \theta \rangle) \tag{B.9}
\]

\[
+ \frac{1}{2} \max_{\phi} (-E_{\phi}(v, h) + E_{\theta}(v, h))^2 \tag{B.10}
\]

In the case where the energy function is linear, then we have the property that

\[-E_{\phi}(v, h) = -E_{\theta}(v, h) + \langle \nabla \theta (-E(v, h)), \phi - \theta \rangle\]

This property removes line B.10 from the Equation. Next, by the equivalency,

\[-E_{\phi}(v, h) + E_{\theta}(v, h) = \langle \nabla \theta (-E(v, h)), \phi - \theta \rangle\]

we can replace line B.11 with \(\frac{1}{2} \max_{\phi} (\langle \nabla \theta (-E(v, h)), \phi - \theta \rangle)^2\). This will give the form in Equation 3.4.
B.4 Proofs of Sigmoid Belief Net Results

The Sigmoid Belief Net energy function is written as

\[-E_\theta(v, h) = v^T c + v^T Wh + h^T b - \sum_{m=1}^{M} \log(1 + \exp([c + Wh]_m)) - \sum_{j=1}^{J} \log(1 + \exp(b_j))\]  

(B.12)

Because the log partition function is a constant at 0, the bounds depend on the form

\[-\min_{n,h}(E_\theta(v_n, h) + E_\theta(v_n, h) - \langle \nabla_\theta(-E(v_n, h)), \phi - \theta \rangle)\]

This requires the bound on the first order Taylor approximation. Note that

\[\nabla^2_{W_m}(-E(v_n, h)) = -\sigma([c + Wh]_m)\sigma(-[c + Wh]_m)hh^T\]

(B.13)

For the Taylor approximation, we want a lower bound on

\[\min_h \sum_{W_m} \nabla^2_{W_m}(-E(v_n, h))W^T_{m}.\]

If we replace \(\nabla^2_{W_m}(-E(v_n, h)) = -\sigma([c + Wh]_m)\sigma(-[c + Wh]_m)hh^T\) with \(\frac{1}{4}hh^T\), which can only decrease the minimum value, then we have

\[4\min_h \sum_{W_m} \nabla^2_{W_m}(-E(v_n, h))W^T_{m}. \geq \min_h \sum_{W_m} hh^T W^T_{m.} = -\max_h \|Wh\|_2^2\]

(B.14)

This is bound the Shatten-\(\infty\) norm, with \(-\frac{1}{4}\max_h \|Wh\|_2^2 \geq -\frac{1}{4}\|W\|_S^2\). Combining with the minus sign in Equation 3.1, this gives the final term of \(\frac{t}{8}\|W\|_S^2\).

Bias terms on \(c\) and \(h\) follow standard arguments for sigmoid functions, where \(\frac{d}{dc_m}(-\log(1 + \exp([c + Wh]_m))) \geq -\frac{1}{4}\).

B.5 Proofs of Replicated Softmax-Restricted Boltzmann Machine Results

The energy function for the RS-RBM is

\[-E_\theta(v, h) = v^T c + v^T Wh + Dh^T b\]
Because this is convex and linear, we focus only on using Equation 3.4. For \( W \), this term depends on the inner product of the gradient and the parameter change \( U \), which is

\[
\max_{v \in \mathbb{Z}^M, \|v\|_1 = D, h \in \{0,1\}} (\text{tr}(Uhv^T))^2 = \\
\max_{v \in \mathbb{Z}^M, \|v\|_1 = D, h \in \{0,1\}} (v^TUh)^2 = D^2 \max_{v \in \mathbb{Z}^M, \|v\|_1 = 1, h \in \{0,1\}} (v^TUh)^2 
\]

(B.15)

To bound B.15 on the maximum \( \ell_1 \) norm of a row, we note that:

\[
D^2 \max_{v \in \mathbb{Z}^M, \|v\|_1 = 1, h \in \{0,1\}} (v^TUh)^2 \leq D^2 \max_{\|v\|_1 = 1, h \in \{0,1\}} (v^TUh)^2 = D^2 \max_{m,h \in \{0,1\}} ([ Uh ]_m)^2 \\
\leq D^2 \| U_m \|_1^2
\]

Alternatively, we note that:

\[
D^2 \max_{v \in \mathbb{Z}^M, \|v\|_1 = 1, h \in \{0,1\}} (v^TUh)^2 \leq D^2 \max_{\|v\|_2 = 1, h \in \{0,1\}} (v^TUh)^2 \\
D^2 J \max_{\|v\|_2 \leq 1, \|h\| \leq 1} (v^TUh)^2 = D^2 J \| U \|_{\infty}^2
\]

The visible bias terms depend on

\[
\max_{v \in \mathbb{Z}^M, \|v\|_1 = D} (cv)^2 \leq D^2 \max_{v \in \mathbb{Z}^M, \|v\|_1 = 1} (cv)^2 \leq D^2 \max_{\|v\|_1 = 1} (cv)^2 \leq D^2 \| v \|_{\infty}^2 
\]

(B.16)

B.5.1 Derivation of optimal steps with a maximum \( \ell_1 \) row penalty.

Suppose that we have a function on a matrix \( X \in \mathbb{R}^{M \times J} \) that is of the form

\[
g(X) = \text{tr}(XA^T) + \frac{L}{2} \max_{m=1,...,M} || X_{m,:} ||_1^2
\]

(B.17)
The minimizer of this equation \( \mathbf{X}^* = \arg\min_{\mathbf{X}} g(\mathbf{X}) \) is analytic. To see this, assume that \( \|\mathbf{X}_m\| \leq b \) for \( m = 1, \ldots, M \), then note

\[
\min_{\mathbf{X}} g(\mathbf{X}) = \min_{\mathbf{X}, b, \|\mathbf{X}_m\|_1 \leq b} \text{tr}(\mathbf{X} \mathbf{A}^T) + \frac{L}{2} b^2 = \min_{\mathbf{X}, b, \|\mathbf{X}_m\|_1 \leq b} \sum_{m} \mathbf{X}_m \mathbf{A}_{m,r} + \frac{L}{2} b^2
\]

The minimizer on \( b^* \) occurs at

\[
b^* = \frac{1}{L} \sum_{m} \|\mathbf{A}_{m,r}\|_\infty. \]

If we assume that rows in \( \mathbf{A} \) have unique entries, then the minimizer on \( \mathbf{X} \) will only have one entry per row, with the minimizer \( \mathbf{X}^* \),

\[
\mathbf{X}_{m,j}^* = \begin{cases} 
-b^* \text{sign}(\mathbf{A}_{m,j}), & |\mathbf{A}_{m,j}| = \max_j |\mathbf{A}_{m,j}| \\
0, & \text{otherwise}
\end{cases}
\]

**B.6 Proofs of Replicated Softmax Belief Net Results**

The energy function for the RSBN is defined

\[
-E_\theta(\mathbf{v}, \mathbf{h}) = \mathbf{v}^T \mathbf{c} + \mathbf{v}^T \mathbf{W} \mathbf{h} + \mathbf{h}^T \mathbf{b} - D \log \sum_{m=1}^{M} \exp([\mathbf{c} + \mathbf{W} \mathbf{h}]_m) - \sum_{j=1}^{J} \log(1 + \exp(b_j))
\]

Focusing first on the matrix \( \mathbf{W} \). Using the infinity norm bound on the log-sum-exp function in Equation B.6 on the term \( -D \log \sum_{m=1}^{M} \exp([\mathbf{c} + \mathbf{W} \mathbf{h}]_m) \). If we perturb the matrix \( \mathbf{W} \) by an matrix \( \mathbf{U} \), then this bound will depend on the final infinity norm in the from Equation B.6, which is

\[
-D^2 \max_{\mathbf{h} \in \mathbb{0}_j} \|\mathbf{U} \mathbf{h}\|_\infty^2 = -D^2 \max_{m, \mathbf{h} \in \mathbb{0}_j} (\mathbf{U}_m \mathbf{h})^2
\]

Now, there is a parallel between Equation B.18 and the bound form on the RS-RBM in Equation B.17, and the bounds derived in Supplemental Section B.5 will hold for \( \mathbf{W} \). The bound on \( \mathbf{c} \) is once again based on the infinity norm in the log-sum-exp
function from Equation B.6, and gives the same bound for $c$ as Equation B.6. The parameter $b$ is bound in the same way as in Supplemental Section B.4.

B.7 Proofs of Deep Belief Net

The energy for a 2-layer DBN is

$$-E_\theta(h, v^{(1)}, v^{(2)}) = v^T c + v^T W^{(1)} h^{(1)} - \sum_{m=1}^M \log(1 + \exp([c + W^{(1)} h^{(1)}]_m))$$

$$+ \langle h^{(1)} \rangle^T b^{(1)} + \langle h^{(1)} \rangle^T W^{(2)} h^{(2)} + \langle h^{(2)} \rangle^T b^{(2)}$$

We first note that the gradients on $W^{(2)}$ and $b^{(2)}$ and $b^{(1)}$ are linear terms in the energy function. This allows us to use Equation 3.4 to bound these terms. For $W^{(2)}$ with a perturbation $U$, this would give

$$\max_{h^{(1)} \in \{0,1\}^{J^{(1)}}, h^{(2)} \in \{0,1\}^{J^{(2)}}} \langle h^{(1)} \rangle^T U h^{(2)} \leq J^{(1)} J^{(2)} \max_{\|h^{(1)}\|_2 \leq 1, \|h^{(2)}\|_2 \leq 1} \langle h^{(1)} \rangle^T U h^{(2)} = J^{(1)} J^{(2)} \|U\|_F^2$$

We now show that $W^{(1)}$ doesn’t change the partition function. To do this, note that the marginal prior on $h^{(1)}$ is independent of $W^{(1)}$. We denote RBM energy as $-E_\theta(h^{(1)}, h^{(2)})$ and the marginal energy on $v$ as $-E_\theta(v|h^{(1)})$. Then the partition function is

$$\log Z(\theta) = \log \sum_v \sum_{h^{(1)}} \sum_{h^{(2)}} \exp(-E_\theta(v|h^{(1)}) - E_\theta(h^{(1)}, h^{(2)}))$$

$$\log Z(\theta) = \log \sum_v \sum_{h^{(1)}} \sum_{h^{(2)}} \exp(-E_\theta(v|h^{(1)})) \exp(-E_\theta(h^{(1)}, h^{(2)}))$$

$$\log Z(\theta) = \log \sum_v \sum_{h^{(1)}} \sum_{h^{(2)}} \exp(-E_\theta(v|h^{(1)})) + \log \sum_v \sum_{h^{(1)}} \sum_{h^{(2)}} \exp(-E_\theta(h^{(1)}, h^{(2)}))$$

The first part of the log partition function, which is the only part dependent on $W$, is a constant. Thus we can simply apply the methods in Supplemental Section B.4.
C.1 Relationship between the FMM and other Bayesian Nonparametric Models

The use of nonparametric Bayesian methods like the Dirichlet process (DP) (Gasthaus et al., 2009; Chen et al., 2011b) removes some of the ad hoc character of classical clustering methods, but there are other limitations within the context of electrophysiological data analysis. The DP and related models are characterized by a scale parameter $\alpha > 0$, and the number of clusters grows as $O(\alpha \log S)$ (Teh, 2010), with $S$ the number of data samples. This growth without limit in the number of clusters with increasing data is undesirable in the context of electrophysiological data, for which there are a finite set of processes responsible for the observed data. Further, when jointly performing mixture modeling across multiple tasks, the hierarchical Dirichlet process (HDP) (Teh et al., 2006) shares all mixture components, which may undermine inference of subtly different clusters.

In this paper we integrate dictionary learning and clustering for analysis of electrophysiological data.
trophysiological data, as in (Gorur et al., 2004; Chen et al., 2011b). However, as an alternative to utilizing a method like DP or HDP (Gasthaus et al., 2009; Chen et al., 2011b) for clustering, we develop a new hierarchical clustering model in which the number of clusters is modeled explicitly; this implies that we model the number of underlying neurons—or clusters—separately from the firing rate, with the latter controlling the total number of observations. This is done by integrating the Indian buffet process (IBP) (Griffiths and Ghahramani, 2005) with the Dirichlet distribution, similar to (Williamson et al., 2010), but with unique characteristics. The IBP is a model that may be used to learn features representative of data, and each potential feature is a “dish” at a “buffet”; each data sample (here a neuronal spike) selects which features from the “buffet” are most appropriate for its representation. The Dirichlet distribution is used for clustering data, and therefore here we jointly perform feature learning and clustering, by integrating the IBP with the Dirichlet distribution. The proposed framework explicitly models the quantity of data (for example, spikes) measured within a given recording interval. To our knowledge, this is the first time the firing rate of electrophysiological data is modeled jointly with clustering and jointly with feature/dictionary learning. The model demonstrates state-of-the-art clustering performance on publicly available data. Further, concerning distinguishing single-unit-events, we demonstrate how this may be achieved using the FMM-DL method, considering new measured (experimental) electrophysiological data.

C.2 Relationship between the FMM and Dirichlet priors

A typical prior for $\pi^{(i)}$ is a symmetric Dirichlet distribution (Gorur et al., 2004),

$$\pi^{(i)} \sim \text{Dir}(\alpha_0/M, \ldots, \alpha_0/M). \quad (C.1)$$
In the limit, $M \to \infty$, this reduces to a draw from a Dirichlet process (Gasthaus et al., 2009; Chen et al., 2011b), represented $\pi^{(i)} \sim \text{DP}(\tilde{\alpha}_0 G_0)$, with $G_0$ the “base” distribution defined in (4.4). Rather than drawing each $\pi^{(i)}$ independently from $\text{DP}(\tilde{\alpha}_0 G_0)$, we may consider the hierarchical Dirichlet process (HDP) (Teh et al., 2006) as

$$\pi^{(i)} \sim \text{DP}(\tilde{\alpha}_1 G), \quad G \sim \text{DP}(\tilde{\alpha}_0 G_0) \quad (C.2)$$

The HDP methodology imposes that the $\{\pi^{(i)}\}$ share the same set of “atoms” $\{\mu_{mn}, \Omega_{mn}\}$, implying a sharing of the different types of clusters across the time intervals $i$ at which data are collected. A detailed discussion of the HDP formulation is provided in (Chen et al., 2011b).

These models have limitations in that the inferred number of clusters grows with observed data (here the clusters are ideally connected to neurons, the number of which will not necessarily grow with longer samples). Further, the above clustering model assumes the number of samples is given, and hence is not modeled (the information-rich firing rate is not modeled). Below we develop a framework that yields hierarchical clustering like HDP, but the number of clusters and the data count (for example, spike rate) are modeled explicitly.

### C.3 Other Formulations of the FMM

Let the total set of data measured during interval $i$ be represented $\mathcal{D}_i = \{X_{ij}\}_{j=1}^{M_i}$, where $M_i$ is the total number of events during interval $i$. In the experiments below, a “recording interval” corresponds to a day on which data were recorded for an hour (data are collected separately on a sequence of days), and the set $\{X_{ij}\}_{j=1}^{M_i}$ defines all signals that exceeded a threshold during that recording period. In addition to modeling $M_i$, we wish to infer the number of distinct clusters $C_i$ characteristic of $\mathcal{D}_i$, and the relative fraction (probability) with which the $M_i$ observations are
apportioned to the $C_i$ clusters.

Let $n_{im}^*$ represent the number of data samples in $D_i$ that are apportioned to cluster $m \in \{1, \ldots, M\} = S$, with $M = \sum_{m=1}^{M} n_{im}^*$. The set $S_i \subset S$, with $C_i = |S_i|$, defines the active set of clusters for representation of $D_i$, and therefore $M$ serves as an upper bound ($n_{im}^* = 0$ for $m \in S \setminus S_i$).

We impose $n_{im}^* \sim \text{Poisson}(b_{m}^{(i)} \phi_m^{(i)})$ with the priors for $b_{m}^{(i)}$ and $\phi_m^{(i)}$ given in Eqs. (4.6) and (4.7). Note that $n_{im}^* = 0$ when $b_{m}^{(i)} = 0$, and therefore $\mathbf{b}^{(i)} = (b_{1}^{(i)}, \ldots, b_{M}^{(i)})^T$ defines indicator variables identifying the active subset of clusters $S_i$ for representation of $D_i$. Marginalizing out $\phi_m^{(i)}$, $n_{im}^* \sim \text{NegBin}(b_{m}^{(i)} \phi_m, p_i)$. This emphasize another motivation for the form of the prior: the negative binomial modeling of the counts (firing rate) is more flexible than a Poisson model, as it allows the mean and variance on the number of counts to be different (they are the same for a Poisson model).

While the above methodology yields a generative process for the number, $n_{im}^*$, of elements of $D_i$ apportioned to cluster $m$, it is desirable to explicitly associate each member of $D_i$ with one of the clusters (to know not just how many members of $D_i$ are apportioned to a given cluster, but also which data are associated with a given cluster). Toward this end, consider the alternative equivalent generative process for $\{n_{im}^*\}_{m=1}^{M}$ (see Lemma 4.1 in (Zhou et al., 2012a) for a proof of equivalence): first draw $M_i \sim \text{Poisson}(\sum_{m=1}^{M} b_{m}^{(i)} \phi_m^{(i)})$, and then

$$(n_{i1}^*, \ldots, n_{iM}^*) \sim \text{Mult}(M_i; \pi_1^{(i)}, \ldots, \pi_M^{(i)}) \quad (C.3)$$

$$\pi_m^{(i)} = b_{m}^{(i)} \phi_m^{(i)}/\sum_{m'=1}^{M} b_{m'}^{(i)} \phi_m^{(i)} \quad (C.4)$$

with $\phi_m^{(i)}$, $\{\phi_m\}$, $\{b_{m}^{(i)}\}$, and $\{p_i\}$ constituted as in (4.6)-(4.7). Note that we have $M_i \sim \text{NegBin}(\sum_{m=1}^{M} b_{m}^{(i)} \phi_m, p_i)$ by marginalizing out $\phi_m^{(i)}$.

Rather than drawing $(n_{i1}^*, \ldots, n_{iM}^*) \sim \text{Mult}(M_i; \pi_1^{(i)}, \ldots, \pi_M^{(i)})$, for each of the $M_i$ data we may draw indicator variables $z_{ij} \sim \sum_{m=1}^{M} \pi_m^{(i)} \delta_m$, where $\delta_m$ is a unit measure

150
concentrated at the point \( m \). Variable \( z_{ij} \) assigns data sample \( j \in \{1, \ldots, M\} \) to one of the \( M \) possible clusters, and \( n^*_{im} = \sum_{j=1}^{M} 1(z_{ij} = m) \), with \( 1(\cdot) \) equal to one if the argument is true, and zero otherwise. The probability vector \( \pi^{(i)} \) defined in (C.4) is now used within the mixture model in (4.4).

As a consequence of the manner in which \( \hat{\phi}^{(i)}_m \) is drawn in (4.6), and the definition of \( \pi^{(i)} \) in (C.4), for any \( p_i \in (0, 1) \), the proposed model imposes

\[
\pi^{(i)} \sim \text{Dir}(b^{(i)}_1 \phi_1, \ldots, b^{(i)}_M \phi_M) \tag{C.5}
\]

### C.4 Additional Connections to Other Bayesian Models

Eq. (C.5) demonstrates that the proposed model is a generalization of (C.1). Considering the limit \( M \to \infty \), and upon marginalizing out the \( \{\nu_m\} \), the binary vectors \( \{b^{(i)}\} \) are drawn from the Indian buffet process (IBP), denoted \( b^{(i)} \sim \text{IBP}(\alpha) \). The number of non-zero components in each \( b^{(i)} \) is drawn from Poisson(\( \alpha \)), and therefore for finite \( \alpha \) the number of non-zero components in \( b^{(i)} \) is finite, even when \( M \to \infty \). Consequently \( \text{Dir}(b^{(i)}_1 \phi_1, \ldots, b^{(i)}_M \phi_M) \) is well defined even when \( M \to \infty \) since, with probability one, there are only a finite number of non-zero parameters in \( (b^{(i)}_1 \phi_1, \ldots, b^{(i)}_M \phi_M) \). This model is closely related to the compound IBP Dirichlet (CID) process developed in (Williamson et al., 2010), with the following differences.

Above we have explicitly derived the relationship between the negative binomial distribution and the CID, and with this understanding we recognize the importance of \( p_i \); the CID assumes \( p_i = 1/2 \), but there is no theoretical justification for this. Note that \( M_i \sim \text{NegBin}(\sum_{m=1}^{M} b^{(i)}_m \phi_m, p_i) \). The mean of \( M_i \) is \( (\sum_{m=1}^{M} b^{(i)}_m \phi_m) p_i / (1 - p_i) \), and the variance is \( (\sum_{m=1}^{M} b^{(i)}_m \phi_m) p_i / (1 - p_i)^2 \). If \( p_i \) is fixed to be 1/2 as in (Williamson et al., 2010), this implies that we believe that the variance is two times the mean, and the mean and variance of \( M_i \) are the same for all intervals \( i \) and \( i' \) for which \( b^{(i)} = b^{(i')} \). However, in the context of electrophysiological data, the rate at which neurons fire...
plays an important role in information content (Donoghue et al., 2007). Therefore, there are many cases for which intervals \( i \) and \( i' \) may be characterized by firing of the same neurons (i.e., \( b^{(i)} = b^{(i')} \)) but with very different rates \( (M_i \neq M_{i'}) \). The modeling flexibility imposed by inferring \( p_i \) therefore plays an important practical role for modeling electrophysiological data, and likely for other clustering problems of this type.

To make a connection between the proposed model and the HDP, motivated by (4.6)-(4.7), consider \( \tilde{\phi} = (\tilde{\phi}_1, \ldots, \tilde{\phi}_M) \sim \text{Dir}(\gamma_0, \ldots, \gamma_0) \), which corresponds to \( (\phi_1, \ldots, \phi_M)/\sum_{m=1}^{M} \phi_m' \). From \( \tilde{\phi} \) we yield a normalized form of the vector \( \phi = (\phi_1, \ldots, \phi_M) \). The normalization constant \( \sum_{m=1}^{M} \phi_m \) is lost after drawing \( \tilde{\phi} \); however, because \( \phi_m \sim \text{Ga}(\gamma_0, 1) \), we may consider drawing \( \tilde{\alpha}_1 \sim \text{Ga}(M\gamma_0, 1) \), and approximating \( \phi \approx \tilde{\alpha}_1 \tilde{\phi} \). With this approximation for \( \phi \), \( \pi^{(i)} \) may be drawn approximately as \( \pi^{(i)} \sim \text{Dir}(\tilde{\alpha}_1 b_1^{(i)} \tilde{\phi}_1, \ldots, \tilde{\alpha}_1 b_M^{(i)} \tilde{\phi}_M) \). This yields a simplified and approximate hierarchy

\[
\pi^{(i)} \sim \text{Dir}(\tilde{\alpha}_1 (b^{(i)} \odot \tilde{\phi})) \quad (C.6)
\]

\[
\tilde{\phi} = (\tilde{\phi}_1, \ldots, \tilde{\phi}_M) \sim \text{Dir}(\gamma_0, \ldots, \gamma_0), \quad \tilde{\alpha}_1 \sim \text{Ga}(M\gamma_0, 1)
\]

with \( b^{(i)} \sim \text{IBP}(\alpha) \) and \( \odot \) representing a pointwise/Hadamard product. If we consider \( \gamma_0 = \hat{\alpha}_0/M \), and the limit \( M \rightarrow \infty \), with \( b^{(i)} \) all ones, this corresponds to the HDP, with \( \hat{\alpha}_1 \sim \text{Ga}(\hat{\alpha}_0, 1) \). We call such a model the non-focused mixture model (NFMM). Therefore, the proposed model is intimately related to the HDP, with three differences: (i) \( p_i \) is not restricted to be 1/2, which adds flexibility when modeling counts; (ii) rather than drawing \( \tilde{\phi} \) and the normalization constant \( \tilde{\alpha}_1 \) separately, as in the HDP, in the proposed model \( \phi \) is drawn directly via \( \phi_m \sim \text{Ga}(\gamma_0, 1) \), with an explicit link to the count of observations \( M_i \sim \text{NegBin}(\sum_{m=1}^{M} b_m^{(i)} \phi_m, p_i) \); and (iii) the binary vectors \( b^{(i)} \) “focus” the model on a sparse subset of the mixture components, while in general, within the HDP, all mixture components have non-zero probability
of occurrence for all tasks $i$. As demonstrated in Section 4.3, this focusing nature of the proposed model is important in the context of electrophysiological data.

C.5 Proof of Lemma 6

Proof. Denote $w_j = \sum_{i=1}^j u_i$, $j = 1, \cdots, m$. Since $w_j$ is the summation of $j$ iid $\log(p)$ distributed random variables, the probability generating function of $w_j$ can be expressed as $G_{W_j}(z) = [\ln(1 - pz)/\ln(1 - p)]^j$, $|z| < p^{-1}$, thus we have

$$
\Pr(w_j = m) = G_{W_j}^{(m)}(0)/m! = \frac{d^m}{dz^m}[\ln(1 - pz)/\ln(1 - p)]^j
$$

$$
= (-1)^mp^j j! s(m, j)/[\ln(1 - p)]^j
$$

(C.7)

where we use the property that $[\ln(1 + x)]^j = j! \sum_{n=j}^{\infty} \frac{(n, j)x^n}{n!}$ (Johnson et al., 2005). Therefore, we have

$$
\Pr(\ell = j|\cdot) \Pr(w_j = n) \Pr(j; -r \ln(1 - p))
$$

$$
\propto (-1)^{n+j} s(n, j)/n! r^j = F(n, j) r^j.
$$

(C.8)

The values $F(n, j)$ can be iteratively calculated and each row sums to one, e.g., the 3rd to 5th rows are

$$
\begin{pmatrix}
2/3! & 3/3! & 1/3! & 0 & 0 & 0 & \cdots \\
6/4! & 11/4! & 6/4! & 1/4! & 0 & 0 & \cdots \\
24/5! & 50/5! & 35/5! & 10/5! & 1/5! & 0 & \cdots 
\end{pmatrix}.
$$

To ensure numerical stability when $\phi > 1$, we may also iteratively calculate the values of $R_\phi(n, j)$.  

153
Appendix D

Supplemental Material for the OPASS Algorithm for Spike-Sorting

D.1 Supplemental Materials

**Notation:** Unless otherwise specified, lower-case English alphabet characters indicate scalars $x \in \mathbb{R}$. Bold indicates column vectors $\mathbf{x} \in \mathbb{R}^p$, and upper-case bold indicates matrices, $\mathbf{X} \in \mathbb{R}^{p \times q}$. Parameters and constants are Greek characters. Time is $t \in [0, T]$, $i \in [N]$ indexes the $N$ neurons, where $[N] = \{1, 2, \ldots, N\}$. Script denotes sets and pipes denote the cardinality of the set, e.g. $|T|$.

Our overall model is then:

\begin{align}
\mathcal{T}_i & \sim \text{PP}(\lambda_i), \quad i \in \mathbb{N}, & \text{where } \Lambda(\cdot) = \sum_{i=1}^{\infty} \lambda_i \delta_{\theta^*_i} \sim \text{GP}(\alpha, \mathcal{H}(\cdot | \phi)), \\
x_i(t) &= \sum_{j=1}^{\mathcal{T}_i} \sum_{k=1}^{K} y_{ijk}^* d_k(t - \tau_{ij}), \quad \text{where } y_{ij}^* \sim \mathcal{N}_K(\mu_i^*, \Sigma_i^*) \quad i, j \in \mathbb{N}, \\
x(t) &= \sum_{i=1}^{\infty} x_i(t) + \epsilon_t, \quad \text{where at any time } t, \epsilon_t \sim \mathcal{N}(0, \Sigma_x) \text{ independently}
\end{align}

(D.1)
Algorithm 2 Generative mechanism for the multi-electrode, non-stationary, discrete-time process

Input: a) the number of bins $T$, and the bin-width $\Delta$
       b) the $K$-by-$L$ dictionary $D$ of $K$ basis functions
       c) the DP hyperparameters $\alpha$ and $\phi$.
       d) the transition matrix $B$ of the neuron AR process

Output: An $M$-by-$T$ matrix $X$ of multielectrode recordings.

1: Initialize the number of clusters $C_1$ to 0.
2: Draw the overall spiking rate $\Lambda \sim \text{Gamma}(\alpha, 1)$.
3: for $t$ in $[T]$ do
4:     Sample $\tilde{z}_t \sim \text{Bernoulli}(\Lambda \Delta)$, with $\tilde{z}_t = 1$ indicating a spike in bin $t$.
5:     if $\tilde{z}_t = 1$ then
6:         Sample $\tilde{\nu}_t$, assigning the spike to a neuron, with $P(\tilde{\nu}_t = i) \propto$
7:            \begin{cases} $|T_i|$ & $i \in [C]$ \\ $\alpha$ & $i = C + 1$ \end{cases}
8:         if $\nu_t = C_t + 1$ then
9:             $C_{t+1} \leftarrow C_t + 1$.
10:            Set $\theta^*_{C_{t+1}} \sim H_{\phi}(\cdot)$, and $\mathcal{T}_{C_{t+1}} = \{t\}$.
11:        else
12:            $\mathcal{T}_{\nu_t} \leftarrow \mathcal{T}_{\nu_t} \cup \{t\}$.
13:        end if
14:     end if
15:     Sample $y_t = (y^1_t; \cdots; y^M_t) \sim N(\mu_t, \Sigma_t)$, determining the spike shape at all electrodes.
16:     $x^m_t = \sum_{h=1}^L D^T_{m,h} y^m_{t-h} + \epsilon^m_t$ where $\epsilon^m_t \sim N(0, \sigma^2)$, $m \in [M]$.
17:     Update the cluster parameters: $\mu_i^* = B \mu_i^* + r_i$ $i \in [C_{t+1}]$
18: end for
Figure D.1: (a) Dictionary learned from the first 5 seconds of data from the HC1 dataset. (b) Percentage of variance explained by each PCA component.

Figure D.2: (a) This shows the average number of true positives versus the average number of false positives in the intracellular cluster for 2 minute segments of the 4 minutes of the experiment. OPASS does better than all the competitors.
**Figure D.3:** Improving $\text{OPass}$ by incorporating multiple channels. (a) Three electrode device showing local proximity of electrodes with channel indexes in large, red numbers. (b) The representation of detected spikes on the 3rd channel in PCA space. This cluster does not seem separable here.

**Figure D.4:** False and true positive detections have the same first-order statistics, making detection using only these statistics quite difficult. (a) Errorbar plots of the true positives, false positives, and missed positives in the IC cluster. While the false positives have slightly more variability, the mean shape for the false positives and the true positives is nearly identical. The true misses have a significantly lower amplitude as well as high variability. (b) All waveforms from the IC neuron as well as those we estimated from the IC neuron projected onto the first two PC space.
**Figure D.5:** Pairs plot of true positives (black), false positives (blue ×’s), and missed positives (red +’s). It does not seem like clustering in this space could yield much improvement.
Figure D.6: `0pass` multielectrode performance. (a) 8 electrode device showing local proximity of electrodes with channel indexes in large, red numbers. (b,c,d) Top three most prevalent waveforms. Each waveform shape is 2 ms long.

Figure D.7: `0pass` scales linearly with amount of data, with a slope smaller than one, meaning that `0pass` can operate in real-time.
Supplemental Material on the Relationships between Sorted Spikes and Local Field Potentials

E.1 Supplemental VB Updates and Equations

The VB update for $q(A^*_k)$ is dependent only on $x_b^{-k}$ and $y_b^{-k}$, $q(\alpha)$, and $q(\gamma)$, and the distribution over $A^*_k$ is judged to be $q(A_k) = \mathcal{N}(\text{vec}(A_k); \text{vec}(\hat{a}_{k1}, \ldots, \hat{a}_{kB}), \Lambda_k)$.

First, let $\Lambda_k = \Sigma^{-1}$ and let $\Lambda_0$ the the prior precision induced by the autoregressive prior, which is a block tri-diagonal matrix. Let $\Lambda_{kbb'}$ be the block in $\Lambda_k$ that corresponds to bins $b$ and $b'$. Each $\Lambda_{kbb'}$ is updated by $\Lambda_{kbb'} = \Lambda_{0bb'} + \gamma R_{kbb'}$. The mean of the variational distribution can be acquired by either solving $\Lambda_k \text{vec}(\hat{a}_{k1}, \ldots, \hat{a}_{kB}) = \text{vec}(v_{k1}^{-k}, \ldots, v_{kB}^{-k})$, or using the forward filtering backwards smoothing algorithm Roberts and Penny (2002), both of which can be solved in $\mathcal{O}(L^3B)$ because of the structure of the precision matrix.

The necessary calculations for the novel terms in Eq. 6.7 are shown below. The hyper parameter $\alpha$ is approximated as a point estimate, and previous experiments have shown that point estimates low-dimensional hyperparameters don’t have a significant impact on the variational lower bound Kim et al. (2012).
\[ E_q[\log p(A_k|\alpha, \gamma) - \log q(A_k)] = \text{const} + 0.5 \log |\Lambda_k| + \frac{\hat{\gamma}}{2(1-\alpha^2)} ||a_{k1}||^2 \quad (E.1) \]

\[ - \frac{\hat{\gamma}}{2} \sum_{b=2}^{B} (||\hat{\alpha}_{kb} - \hat{\alpha}_{k,b-1}||^2 + \hat{\alpha}^2 \text{cov}(a_{k,b-1}, a_{k,b})) \]

Because the matrix \( \Lambda_k \) is a block diagonal matrix, the determinant can be calculated in \( O(L^3B) \).

For Eq. 6.7, we also need to calculate:

\[ E_q[p(x|A_1,...,K, \zeta, \tau)] = \text{const} - \frac{\hat{\tau}}{2} \sum_b \left( ||x_b - \sum_k y_{kb} * \hat{\alpha}_{kb}||^2 + \sum_k T_b \text{trace}(R_{kb} \text{cov}(a_{kb})) \right) \quad \text{E.2} \]

The rest of the equations are standard and can be found in Roberts and Penny (2002); Hughes and Sudderth (2013).

E.2 INLA for the non-clustering dynamic model

To fit the non-clustering dynamic model in the INLARue et al. (2009) framework, the full model likelihood is rewritten with \( z_j = \delta_j \), and \( \zeta_j = 1 \) (so that \( A_j = A^*_j \)). This likelihood can be written as:

\[ p(x, \Theta) = \prod_{b=1}^{B} [p(x_b|\Theta)] \prod_{j=1}^{J} [p(A^*_j|\alpha, \gamma)] p(\alpha, \gamma) \quad \text{E.3} \]

After reformulation, this observation likelihood can be combined with the prior to get the MAP estimate for a given \{\alpha, \gamma\} by standard Kalman filter EM methods. INLA calculates an approximate posterior jointly over the hyperparameters with the latent variables integrated out, \( p(\alpha, \gamma|x) \). To approximate the marginal posterior, this requires the calculation of:

\[ p(\alpha, \gamma|x) \propto \frac{p(x, \{A^*_1, \ldots, A^*_J\}, \alpha, \gamma)}{p(\{A^*_1, \ldots, A^*_J\}|x, \alpha, \gamma)} \frac{\{A^*_1, \ldots, A^*_J\} = \{A^*_1, \ldots, A^*_J\}^{MAP}}{\{A^*_1, \ldots, A^*_J\}^{MAP}_{(\alpha, \gamma)}} \quad \text{E.4} \]
Letting $\tilde{R}_b$ be a block matrix with block row \( j \) represented by \([R_{j1b}, \ldots, R_{jJb}]\), $\tilde{\nu}_b = \text{vex}([\nu_{1b}, \ldots, \nu_{Jb}])$, and $\tilde{d}_b = \text{vex}([a_{1b}^*, \ldots, a_{Jb}^*])$, the full measurement model can be equivalently written:

$$
\log p(x_b|y_{jb}, d_{jb}, j = 1, \ldots, J) \approx \text{const} - \frac{\tau T_b}{2} (d_b^T \tilde{R}_b d_b - 2 \tilde{\nu}_b^T \tilde{d}_b) \tag{E.5}
$$

After reformulation, this observation likelihood can be combined with the prior to get the MAP estimate for a given $\{\alpha, \gamma\}$ by standard Kalman filter EM methods. To calculate the full posterior over $\{\alpha, \gamma\}$, the INLA method is used which allows for analysis of model uncertainty and also the calculation of the Bayesian MMSE estimator. INLA requires the calculation of:

$$
p(\alpha, \gamma|x) \propto p(x, \tilde{D}, \alpha, \gamma) \left| \frac{p(D|x, \alpha, \gamma)}{p(D^{MAP}|\alpha, \gamma)} \right|_{D=D^{MAP}} \tag{E.6}
$$

$D^{MAP}$ is found by forward filtering-backwards smoothing, and the conditional posterior can be calculated by $p(D = \tilde{D}^{MAP}(\alpha, \gamma)|x, \alpha, \gamma) = p(d_B|x, \alpha, \gamma) \prod_{b=1}^{B-1} p(d_b|d_{b+1}|x, \alpha, \gamma) = \text{const} + (1/2) \log |\Phi_B^{-1}| + \sum_{b=1}^{B-1} (1/2) \log |\Phi_B^{-1} + \gamma I|$, where $I$ denotes the identity matrix, $\Phi_1^{-1} = ((1 + \alpha^2)^{-1} \gamma I + \tau T_1 \tilde{R}_1)$, and $\Phi_b^{-1} = ((\gamma I + \tilde{\alpha}^{-2} \Phi_{b-1}^{-1}) + \tau T_b \tilde{R}_b)$. Speed improvements on this calculations are achieved by using the near sparsity of $\tilde{R}_b$, and linear algebra identities are utilized so that only $\Phi_b^{-1}$ needs to be calculated and store and the inverse is never calculated.

The procedure for calculating the approximate posterior is as follows. First, the MAP for $\{\alpha, \gamma\}$ is found by the EM maximization, and the curvature around the MAP point is estimated. A set of grid points is defined based upon the SVD of the curvature using the smart gridding algorithm shown in Rue et al. (2009). This quantity is evaluated at grid points around $\{\alpha, \gamma\}^{MAP}$ determined by the approximate curvature around the MAP estimate, and sample points are normalized to get the
Figure E.1: This is the posterior found from a single cell in INLA

Table E.1: Hold-out RFE predicting the Nucleus Accumbens region with the novel environment dataset

<table>
<thead>
<tr>
<th>Animal</th>
<th>Time-Invariant</th>
<th>Non-Clustering</th>
<th>Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0111</td>
<td>0.0502</td>
<td>0.0562</td>
</tr>
<tr>
<td>2</td>
<td>0.0739</td>
<td>0.1664</td>
<td>0.1589</td>
</tr>
<tr>
<td>3</td>
<td>0.0055</td>
<td>0.2251</td>
<td>0.2152</td>
</tr>
<tr>
<td>4</td>
<td>0.0605</td>
<td>0.2260</td>
<td>0.1942</td>
</tr>
<tr>
<td>7</td>
<td>0.0873</td>
<td>0.1777</td>
<td>0.1724</td>
</tr>
<tr>
<td>8</td>
<td>0.0638</td>
<td>0.1781</td>
<td>0.1599</td>
</tr>
<tr>
<td>9</td>
<td>0.0733</td>
<td>0.2256</td>
<td>0.1939</td>
</tr>
<tr>
<td>10</td>
<td>0.0272</td>
<td>0.0795</td>
<td>0.1359</td>
</tr>
<tr>
<td>11</td>
<td>0.0283</td>
<td>0.0797</td>
<td>0.0866</td>
</tr>
<tr>
<td>12</td>
<td>0.0358</td>
<td>0.0796</td>
<td>0.0881</td>
</tr>
</tbody>
</table>

posterior estimate. Letting $i$ denote $i^{th}$ sample point, then Bayesian MMSE estimator can be evaluated as:

$$\hat{D}_{MMSE} = \sum_i \hat{p}(\alpha^{(i)}, \gamma^{(i)} | x)\mathbb{E}_{p(\hat{D}|x, \alpha^{(i)}, \gamma^{(i)})}[\hat{D}] = \sum_i \hat{p}(\alpha^{(i)}, \gamma^{(i)} | x)\hat{D}_{MAP}^{(E.7)}$$

Additionally, the time-invariant model that Rasch et al. (2009) used for a single neuron and is generalized to multi-neuron setup is a special case of this inference when $B = 1$.  

163
Table E.2: Hold-out RFE predicting the Prelimbix Cortex region with the novel environment dataset

<table>
<thead>
<tr>
<th>Animal</th>
<th>Time-Invariant</th>
<th>Non-Clustering</th>
<th>Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0128</td>
<td>0.0518</td>
<td>0.0597</td>
</tr>
<tr>
<td>2</td>
<td>0.1116</td>
<td>0.2110</td>
<td>0.2047</td>
</tr>
<tr>
<td>3</td>
<td>0.0132</td>
<td>0.2424</td>
<td>0.2265</td>
</tr>
<tr>
<td>5</td>
<td>0.0822</td>
<td>0.1576</td>
<td>0.1689</td>
</tr>
<tr>
<td>6</td>
<td>0.0987</td>
<td>0.1669</td>
<td>0.1651</td>
</tr>
<tr>
<td>7</td>
<td>0.1065</td>
<td>0.2014</td>
<td>0.1966</td>
</tr>
<tr>
<td>8</td>
<td>0.0904</td>
<td>0.2163</td>
<td>0.2016</td>
</tr>
<tr>
<td>9</td>
<td>0.0903</td>
<td>0.2513</td>
<td>0.2211</td>
</tr>
<tr>
<td>10</td>
<td>0.0355</td>
<td>0.0888</td>
<td>0.1467</td>
</tr>
<tr>
<td>11</td>
<td>0.0391</td>
<td>0.0912</td>
<td>0.1000</td>
</tr>
<tr>
<td>12</td>
<td>0.0313</td>
<td>0.0724</td>
<td>0.1177</td>
</tr>
</tbody>
</table>

Table E.3: Hold-out RFE predicting the Thalamus region with the novel environment dataset

<table>
<thead>
<tr>
<th>Animal</th>
<th>Time-Invariant</th>
<th>Non-Clustering</th>
<th>Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1016</td>
<td>0.1518</td>
<td>0.1612</td>
</tr>
<tr>
<td>2</td>
<td>0.1300</td>
<td>0.2257</td>
<td>0.2182</td>
</tr>
<tr>
<td>3</td>
<td>0.0327</td>
<td>0.3429</td>
<td>0.3863</td>
</tr>
<tr>
<td>4</td>
<td>0.1286</td>
<td>0.3470</td>
<td>0.3184</td>
</tr>
<tr>
<td>5</td>
<td>0.2146</td>
<td>0.2880</td>
<td>0.2965</td>
</tr>
<tr>
<td>6</td>
<td>0.1260</td>
<td>0.1904</td>
<td>0.1949</td>
</tr>
<tr>
<td>7</td>
<td>0.1204</td>
<td>0.2205</td>
<td>0.2143</td>
</tr>
<tr>
<td>8</td>
<td>0.2362</td>
<td>0.3765</td>
<td>0.3618</td>
</tr>
<tr>
<td>9</td>
<td>0.0861</td>
<td>0.3765</td>
<td>0.3444</td>
</tr>
<tr>
<td>10</td>
<td>0.1605</td>
<td>0.1864</td>
<td>0.2556</td>
</tr>
<tr>
<td>11</td>
<td>0.0325</td>
<td>0.0774</td>
<td>0.0869</td>
</tr>
<tr>
<td>12</td>
<td>0.0715</td>
<td>0.1111</td>
<td>0.1229</td>
</tr>
</tbody>
</table>
Table E.4: Hold-out RFE predicting the Ventral Tegmental Area region with the novel environment dataset

<table>
<thead>
<tr>
<th>Animal</th>
<th>Time-Invariant</th>
<th>Non-Clustering</th>
<th>Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0991</td>
<td>0.1489</td>
<td>0.1583</td>
</tr>
<tr>
<td>2</td>
<td>0.1012</td>
<td>0.1918</td>
<td>0.1848</td>
</tr>
<tr>
<td>3</td>
<td>0.0624</td>
<td>0.2598</td>
<td>0.2564</td>
</tr>
<tr>
<td>4</td>
<td>0.1061</td>
<td>0.3086</td>
<td>0.2825</td>
</tr>
<tr>
<td>5</td>
<td>0.1081</td>
<td>0.1763</td>
<td>0.1811</td>
</tr>
<tr>
<td>6</td>
<td>0.1048</td>
<td>0.1683</td>
<td>0.1690</td>
</tr>
<tr>
<td>7</td>
<td>0.1158</td>
<td>0.2143</td>
<td>0.2088</td>
</tr>
<tr>
<td>8</td>
<td>0.1410</td>
<td>0.2569</td>
<td>0.2427</td>
</tr>
<tr>
<td>9</td>
<td>0.0594</td>
<td>0.2723</td>
<td>0.2460</td>
</tr>
<tr>
<td>10</td>
<td>0.0962</td>
<td>0.1323</td>
<td>0.1891</td>
</tr>
<tr>
<td>11</td>
<td>0.0333</td>
<td>0.0803</td>
<td>0.0884</td>
</tr>
<tr>
<td>12</td>
<td>0.0568</td>
<td>0.0960</td>
<td>0.1042</td>
</tr>
</tbody>
</table>
Bibliography


167


Biography

David Carlson was born in Williamsburg, Virginia in 1988. He received his B.S.E in Electrical and Computer Engineering from Duke University in May 2010, and a Master of Science in Electrical and Computer Engineering in December 2014, and expects to receive the Ph.D. in Electrical and Computer Engineering in May 2015. He received the Pratt-Gardner Fellowship from Duke University to support his graduate research. He is a member of Eta Kappa Nu.