Statistical Machine Learning Methods for High-dimensional Neural Population Data Analysis

Yuanjun Gao

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2017

ABSTRACT

Statistical Machine Learning Methods for High-dimensional Neural Population Data Analysis

Yuanjun Gao

Advances in techniques have been producing increasingly complex neural recordings, posing significant challenges for data analysis. This thesis discusses novel statistical methods for analyzing high-dimensional neural data. Part one discusses two extensions of state space models tailored to neural data analysis. First, we propose using a flexible count data distribution family in the observation model to faithfully capture over-dispersion and under-dispersion of the neural observations. Second, we incorporate nonlinear observation models into state space models to improve the flexibility of the model and get a more concise representation of the data. For both extensions, novel variational inference techniques are developed for model fitting, and simulated and real experiments show the advantages of our extensions. Part two discusses a fast region of interest (ROI) detection method for large-scale calcium imaging data based on structured matrix factorization. Part three discusses a method for sampling from a maximum entropy distribution with complicated constraints, which is useful for hypothesis testing for neural data analysis and many other applications related to maximum entropy formulation. We conclude the thesis with discussions and future works.

Table of Contents

List of Figures **iv**

List of Tables **viii**

1 Introduction **1**

1.1 Neuroscience and statistics 2

1.2 Dimensionality reduction for neural data 3

1.3 Latent variable models and state space models 5

1.4 Statistical inference for latent variable models 7

1.5 Overview of the thesis 14

I Neural Population Data Analysis with Latent Variable Models **16**

2 Generalized Count Linear Dynamical System **17**

2.1 Introduction 18

2.2 Generalized count distributions 20

2.3 Generalized count linear dynamical system model formulation 23

2.4 Inference and learning in GCLDS 25

2.4.1 E-step: variational inference with dual optimization 26

2.4.2 M-step: analytical form 28
2.4.3 Practical concerns 29
2.4.4 Dual optimization for E-step 30
2.5 Model evaluation by leave-one-neuron-out error 34
2.6 Experiments 35
2.6.1 Simulation examples 35
2.6.2 Real data analysis 36
2.7 Discussion 41

3 Linear Dynamical Neural Population Models Through Nonlinear Embeddings 43
3.1 Introduction 44
3.2 Notation and overview of neural data 45
3.3 Latent LDS neural population models with a linear rate function . . . 46
3.4 Nonlinear latent variable models for neural populations 48
3.5 Inference by Auto-encoding variational Bayes 49
3.6 Experiments 53
3.6.1 Simulation examples 54
3.6.2 Real data analysis 57
3.7 Discussion 62

II Region of Interest Detection for Calcium Imaging Data 64

4 Region of Interest Detection for Calcium Imaging Data 65
4.1 Introduction 66
4.2 Algorithm 68
4.2.1 Problem formulation 68

4.2.2 Greedy algorithm 70
4.2.3 Shape fine-tuning 72
4.2.4 Other details 74
4.3 Experiments 75
4.3.1 Simulation examples 75
4.3.2 Real data analysis 76
4.4 Discussion 80

III Maximum Entropy Flow Networks 81

5 Maximum Entropy Flow Network 82
5.1 Introduction 83
5.2 Background 86
5.2.1 Maximum entropy modeling and Gibbs distribution 86
5.2.2 Normalizing flows 87
5.3 Maximum entropy flow network (MEFN) algorithm 88
5.3.1 Formulation 88
5.3.2 Algorithm 89
5.4 Experiments 91
5.4.1 A maximum entropy problem with known solution 91
5.4.2 Risk-neutral asset pricing 94
5.4.3 Modeling images of textures 97
5.5 Discussion 102

6 Conclusion and discussion 104

Bibliography 107

List of Figures

1.1 Graphical model representation of state space model 6

2.1 *Left panel*: mean firing rate and variance of neurons in primate motor cortex during the peri-movement period of a reaching experiment (see §2.6.2). The data exhibit under-dispersion, especially for high firing-rate neurons. The two marked neurons will be analyzed in detail in Figure 2.2. *Right panel*: the expectation and variance of the GC distribution with different choices of the function g 22

2.2 Examples of fitting result for selected high-firing neurons. Each row corresponds to one neuron as marked in left panel of Figure 2.1 – *left column*: fitted $g(\cdot)$ using GCLDS and PLDS; *middle and right column*: fitted mean and variance of PLDS and GCLDS. See text for details. . 38

2.3 Goodness-of-fit for monkey data during the reaching period – *left panel*: percentage reduction of mean-squared-error (MSE) compared to the baseline (homogeneous Poisson process); *middle panel*: percentage reduction of predictive negative log likelihood (NLL) compared to the baseline; *right panel*: fitted variance of PLDS and GCLDS for all neurons compared to the observed data. Each point gives the observed and fitted variance of a single neuron, averaged across time. 39

2.4 Goodness-of-fit for monkey data during the preparatory period – *Left panel:* Temporal cross-covariance averaged over all 81 units during the preparatory period, compared to the fitted cross-covariance by PLDS and GCLDS-full. *Right panel*: fitted variance of PLDS and GCLDS-full for all neurons compared to the observed data (averaged across time). 41

3.1 Sample simulation result with "grid cell" type response. *Left panel:* Fitted latent variable compared to true latent variable; *Upper right panel:* Fitted rate compared to the true rate for 4 sample neurons; *Bottom right panel:* Inferred trace of the latent variable compared to true latent trace. Note that the latent trajectory for a 1-dimensional latent variable is identifiable up to multiplicative constant, here we scale the latent variables to lie between 0 and 1. 57

3.2 Results for fits to Macaque V1 data (single orientation) (a) Comparing true firing rate (black line) with fitted rate from PLDS (blue) and PfLDS (red) with 2 dimensional latent space for selected neurons (orientation 0°, averaged across all 120 training trials); (b)(c) 2D latent-space embeddings of 10 sample training trials, color denotes phase of the grating stimulus (orientation 0°); (d)(e) Predictive mean square error (MSE) and predictive negative log likelihood (NLL) reduction with one-step-ahead prediction, compared to a baseline model (homogeneous Poisson process). Results are averaged across 12 orientations. 59

3.3 Macaque V1 data fitting result (full data) (a)(b) Predictive MSE and NLL reduction. (c) 3D embedding of the mean latent trajectory of the neuron activity during 300ms to 500ms after stimulus onset across grating orientations $0°, 5°, ..., 175°$, here we use PfLDS with 4 latent dimensions and then project the result on the first 3 principal components. 60

3.4 Macaque center-out reaching data analysis: (a) 5 sample reaching trajectory for each of the 14 target locations. Directions are coded by different color, and distances are coded by different marker size; (b)(c) 2D embeddings of neuron activity extracted by PLDS and PfLDS, circles represent 50ms before movement onset and triangles represent 340ms after movement onset. Here 5 training reaches for each target location are plotted; (d) Predictive negative log likelihood (NLL) reduction with one-step-ahead prediction. 61

4.1 Simulated calcium data. 77

4.2 Real calcium data. 78

4.3 ROI detection for the full Misha data, each sub-figure represents a z-slice. 79

5.1 Example results from the ME problem with known Dirichlet ground truth. *Left panel*: The normal density p_0 (purple) and iid samples from p_0 (red points). *Middle panel*: The MEFN transforms p_0 to the desired maximum entropy distribution p_{ϕ^*} on the simplex (calculated density p_{ϕ^*} in purple). Truly iid samples are easily drawn from p_{ϕ^*} (red points) by drawing from p_0 and mapping those points through f_{ϕ^*}. Shown in the middle panel are the same points in the top left panel mapped through f_{ϕ^*}. Samples corresponding to training the same network as MEFN to simply match the specified moments (ignoring entropy) are also shown (dark green points; see text). *Right panel*: The ground truth (in this example, known to be Dirichlet) distribution in purple, and iid samples from it in red. 93

5.2 Constructing risk-neutral measure from observed option price. *Left panel*: fitted risk-neutral measure by Gibbs and MEFN method. *Middle panel*: Q-Q plot for the quantiles from the distributions on the left panel. *Right panel*: observed and fitted option price for different strikes. 96

5.3 Analysis of texture synthesis experiment. See text for description. . . 99

5.4 Random samples (first 5 columns) and the mean image of 20 random samples (last column) from texture net (upper row) and MEFN (bottom row) for the stone example. 100

5.5 Brick example result. First row gives the raw input. The bottom 3 rows give 5 random samples (first 5 columns) and the mean image of 20 random samples (last column) from texture net (row 2) and MEFN with large initial texture cost penalty (row 3) and smaller initial texture cost penalty (bottom row) for the brick example. 101

List of Tables

2.1 Special cases of GCGLM. For all models, the GCGLM parametrization for θ is only associated with the slope $\theta(x) = \beta x$, and the intercept α is absorbed into the $g(\cdot)$ function. In all cases we have $g(k) = -\infty$ outside the stated support of the distribution. Whenever unspecified, the support of the distribution and the domain of the $g(\cdot)$ function are non-negative integers $\mathbb{N}$. 24

2.2 Simulation result for PLDS and GCLDS. Showing the leave-one-neuron-out mean square error (MSE) and negative log likelihood (NLL) for PLDS and GCLDS, as well as the improvement of GCLDS over PLDS. Results are averaged across 50 independent repeats with standard error showing in parentheses. 37

3.1 Simulation results with a linear observation model: Each column contains results for a distinct experiment. For each generative model and inference algorithm (one per row), we report the one-step-ahead predictive log likelihood (PLL) and computation time (in minutes) of the model fit to each dataset. 56

5.1 Quantitative measure of image diversity using 20 randomly sampled images 100

Acknowledgments

Five years ago, when I first arrived at New York, I expected that the following few years can be tough. What I did not expect is how this five years would expand my mind so much and make me such a different person. This thesis would be impossible without the support and help of many people from Columbia statistics department, my friends and my family.

First I would like to thank my committee members. My academic advisor Dr. John Cunningham, who is organized, energetic, supportive and caring, provided me with valuable guidance and insights. It is his constant encouragement that makes me productive enough to finish several nice projects during my Ph.D. study. Dr. Liam Paninski aroused my interest in computational neuroscience by showing fascinating animations of decoding the primate motor cortex in a student seminar. I explored many interesting projects with him and was constantly amazed by his knowledge and depth of thinking. Dr. Tian Zheng was my mentor in my first year of PhD study. She gave me confidence in studying and researching, developed my interest in applied statistics and introduced me to many interesting research areas. I would also like to thank Drs. John Paisley and Mark Churchland for being in my committee and for careful reading of my thesis manuscript.

During my study in Columbia, I had the opportunity to participate a few other interesting research projects which I was not able to put in this thesis. I thank Drs. Andrew Gelman, Rahul Mazumder, Matthew Connelly, Shawn Simpson and Lauren

Hannah for bringing me with nice projects that expanded my horizon and developed my skills significantly.

I was very lucky to be in an active research group. I would like to thank Evan Archer, Daniel Soudry, Josh Merel, Eftychios Pnevmatikakis, Ari Pakman, Uygar Sumbul, Gabriel Loaiza-Ganem, Lars Buesing, Xuexin Wei, Christian Andersson Naesseth, Scott Linderman and David Pfau, among others, for the helpful discussion and collaboration. I appreciate the opportunities to learn from them.

I would like to thank many researchers for providing data for the the research. Krishna V. Shenoy, Byron Yu, Gopal Santhanam and Stephen Ryu provided the macaque motor cortical data. Arnulf Graf, Adam Kohn, Tony Movshon, and Mehrdad Jazayeri provided the macaque V1 data. Misha Ahrens provided the zebrafish data.

I would like to thank my friends Shuaiwen Wang, Haolei Weng, Yuting Ma, Lu Meng, Jingjing Zou, Lisha Qiu, Yixin Wang, Yilong Zhang, Qiao Feng, Tianchen Qian, Guohui Guan, Tiantian Nie, Liang Liang, Mingsi Long, Xufei Wang, Shiman Ding and Yinting Hu, among others, for cheering me up when I was down, and for the helpful and enlightening discussions about both research and life.

Finally I would like to thank my parents, Ying Li and Zhi Gao, who made me a smart and hard-working kid and constantly supported me during my Ph.D. study. I owe so much to them for their love and support.

To My Parents

Chapter 1

Introduction

Until recently, neural data analysis techniques focused primarily on the analysis of single neurons and small populations. However, new experimental techniques have enabled the simultaneous recording of ever-larger neural populations [Robinson *et al.*, 2012; Ahrens and Keller, 2013; Prevedel *et al.*, 2014]. The abundance of data provides both opportunities and challenges for neural data analysis, and has spurred a search for new statistical methods [Stevenson and Kording, 2011; Cunningham and Yu, 2014; Gao and Ganguli, 2015] . Indeed, statistical models have provided principled ways to performing signal processing, exploratory analysis, statistical modeling, scientific hypothesis testing, etc. This thesis introduces a set of methods related to high-dimensional neural data analysis.

The rest of this chapter provides high level motivation and background for the thesis, and provides an overview for the rest of the thesis.

1.1 Neuroscience and statistics

Neurons communicate by generating temporally fast ($\sim$ 1ms) electrical signals called action potentials, or "spikes". The temporal sequence of action potentials generated by a single neuron is called its "spike train", which can be represented by a one-dimensional point process. The spike trains encode external stimuli and intentions, allowing humans or animals to understand complex environments and perform complicated tasks. Understanding how the billions of neurons in the brain respond to external stimulus, process and transmit information, and control the behavior is an important question. And statistics has been playing a significant role in the neuroscience community in many aspects [Kass *et al.*, 2005; Paninski *et al.*, 2007]. We give a brief overview for the main contributions of statistical methods in neuroscience below.

To begin with, converting noisy observations from various neural recording techniques into clean signal requires specific statistical models. For electrophysiological data, clustering, mixture models and factor analysis techniques have been extensively applied to the detection and classification of spikes from recorded voltage signals, also known as "spike sorting". See Lewicki [1998] for a review. For calcium imaging data, many statistical methods exist for region of interest detection and calcium deconvolution [Mukamel *et al.*, 2009; Vogelstein *et al.*, 2009; Pnevmatikakis *et al.*, 2016; Friedrich and Paninski, 2016].

Many statistical methods are also highly needed for exploring and understanding the structure of the neural data. Early attempts include using summary statistics such as peristimulus time histogram (PSTH) [Gerstein and Kiang, 1960] and spike triggered average [de Boer and Kuyper, 1968; Theunissen *et al.*, 2001] to visualize single neuron activities given certain stimuli. Supervised learning techniques such as

generalized linear models (GLM) [Paninski, 2004; Truccolo *et al.*, 2005; Stevenson *et al.*, 2008] provide statistical formulations that link the spiking data to stimuli, spiking history and interneuron interactions. Unsupervised learning techniques such as principle component analysis (PCA) [Churchland *et al.*, 2012] and state space models [Lawhern *et al.*, 2010; Macke *et al.*, 2011] provide useful data tools for visualizing and understanding high-dimensional neural data.

Scientific hypotheses of neural data are formulated and tested under statistical frameworks, providing better understanding of the neural data structure [Olshausen and Field, 1997; Schneidman *et al.*, 2006; Churchland *et al.*, 2012]. Applications such as neural prosthetics [Shenoy *et al.*, 2011; Gilja *et al.*, 2012] and optimal experiment design [Nelken *et al.*, 1994; Lewi *et al.*, 2011] have also benefited greatly from statistical methods.

Recent developments in technology enabled simultaneous recordings of neuron populations, which can be represented as a high-dimensional time series. Statistical methods that capturing the key structure of the high-dimensional neural activities allow better understanding of the underlying mechanism of neural activities and are becoming more important in computational neuroscience [Cunningham and Yu, 2014]. Below we introduce dimensionality reduction techniques.

1.2 Dimensionality reduction for neural data

Many studies and theories in neuroscience posit that high-dimensional neural spike trains are a noisy observation of some underlying, low-dimensional, and time-varying signal of interest. A line of research has focused on developing dimensionality reduction techniques for neural data that captures the key structure of the data. As discussed in Cunningham and Yu [2014], dimensionality reduction techniques enable

better data visualization for the neural activity, facilitate single trial data analysis, and shed light on the structure of neural population response.

Denote $X \in \mathbb{R}^{T \times n}$ as the n-dimensional data with T observations. Dimensionality reduction methods aim to identify a reduced version of the data $Z \in \mathbb{R}^{T \times m}$ ($m \ll n$) that captures the key features of the data. Linear dimensionality reduction methods, such as principal component analysis (PCA) and factor analysis (FA), in general takes the form of matrix factorization, where we aim at approximating the data by a low rank matrix

$$X \approx Z \cdot C, \tag{1.1}$$

where $C \in \mathbb{R}^{m \times n}$ links the reduced data to the full data. Nonlinear dimensionality reduction methods usually try to identify a nonlinear mapping that relates the reduced data Z with the full data X [Roweis and Saul, 2000; Tenenbaum *et al.*, 2000; Lawrence, 2004].

In the spike train setting, X represents (maybe a transformed or smoothed version of) the spike counts of n neurons in T time bins. Matrix Z represents a learnt low-dimensional latent intensity that captures the main variability of the data and can be used to provide visualization for neuron activities. Matrix C describes how the low-dimensional intensity is linked to the observation and can be used to summarize the behavior of each neuron. In calcium imaging setting, X represents the n-dimensional vectorized image recorded in T time bins, which can be decomposed as a product of spatial components Z representing shape of neurons (or other regions of interest) and temporal components C representing the activity of each neuron.

Building low-dimensional models for neural spike train data setting is complicated by the discrete observation and the temporal structure. The spike count data does not conform to the commonly used Gaussian assumption and requires count distribu-

tion families (Poisson distribution, for example) to describe its distribution [Paninski, 2004; Truccolo *et al.*, 2005; Macke *et al.*, 2011; Pfau *et al.*, 2013]. The spike train also exhibits rich temporal dynamics, and incorporating the temporal structure in the model can help de-noise the data and more faithfully capture the structure [Yu *et al.*, 2009; Macke *et al.*, 2011]. Among the various formulations for dimensionality reduction, state space models, or more generally latent variable models, provide a popular framework for neural data modeling due to its generative nature and flexibility. We briefly introduce the main idea of this framework in the next section.

1.3 Latent variable models and state space models

Latent variable models are a class of probabilistic models that models the generative process of the observation by hidden variables that are linked to the observation. Latent variable modeling provides a natural and principled way of modeling the structure of the data that are affected by unseen hidden variables, and is useful for summarizing the data, handling missing data, making predictions and so on.

Formally, latent variable models assume that the observation $\mathbf{x}$ is affected by unobserved variables $\mathbf{z}$ and propose a probability distribution family $p_\theta(\mathbf{x}, \mathbf{z})$ parameterized by parameter θ. Model fitting involves identifying optimal model parameters θ as well as the latent variables $\mathbf{z}$, both of which are of interest in data analysis. Latent variable modeling is natural in neural data analysis since the observed neural activities are highly coupled with unobserved neurons, intention, behavior and external stimuli [Sahani, 1999; Kulkarni and Paninski, 2007; Macke *et al.*, 2011].

State space model is a class of latent variable models that models time series data $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$ ($\mathbf{x}_t \in \mathbb{R}^n$ for $t = 1, ..., T$) by assuming a hidden time series $\mathbf{z} = \{\mathbf{z}_1, ..., \mathbf{z}_T\}$ ($\mathbf{z}_t \in \mathbb{R}^m$ for $t = 1, ..., T$) with a Markovian structure that are

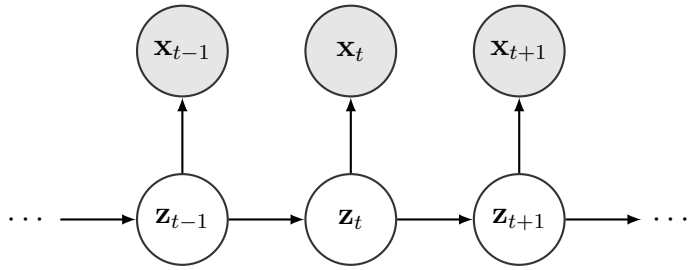

Figure 1.1: Graphical model representation of state space model

coupled with the observation. The generative model is specified by

- initial latent state distribution: $p(\mathbf{z}_1)$,
- dynamic model specifying the evolution of the latent states: $p(\mathbf{z}_{t+1}|\mathbf{z}_t)$ for $t = 1, ..., T-1$,
- observation model $p(\mathbf{x}_t|\mathbf{z}_t)$ for $t = 1, ..., T$.

Figure 1.1 gives the graphical model representation of state space models. The joint distribution is therefore of the form

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}_1) \prod_{t=1}^{T-1} p(\mathbf{z}_{t+1}|\mathbf{z}_t) \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{z}_t). \tag{1.2}$$

The latent variables encode a rich dependency structure through both the dynamic model and the observation model. And the Markovian structure of the latent variable helps make the model interpretable and the inference tractable (in certain cases). Those advantages lead to natural applications of state space model in neural data [Brown *et al.*, 2001; Paninski *et al.*, 2010; Macke *et al.*, 2011]. The model is especially related to the dynamical view of the motor cortex, which states that neural activities in motor system reflect both the outputs to drive the motion and the internal processes that helps to generate motion but is poorly described by the motion [Churchland *et al.*, 2012; Shenoy *et al.*, 2013]. This dynamical system view has been essential for

building robust algorithms for neural prosthetics [Shenoy *et al.*, 2011; Gilja *et al.*, 2012; Kao *et al.*, 2015]

The most commonly used state space model assumes a linear Gaussian structure,

$$p(\mathbf{z}_1) \sim \mathcal{N}(\mu_1, Q_1), \tag{1.3}$$

$$p(\mathbf{z}_{t+1}|\mathbf{z}_t) \sim \mathcal{N}(A\mathbf{z}_t, Q), \tag{1.4}$$

$$p(\mathbf{x}_t|\mathbf{z}_t) \sim \mathcal{N}(C\mathbf{z}_t, \Sigma), \tag{1.5}$$

where $\mu_1 \in \mathbb{R}^m$ and $Q_1 \in \mathbb{R}^{m\times m}$ give the expectation and covariance of the initial states, $A \in \mathbb{R}^{m\times m}$ models the relation of the states of two nearby time points, and $Q \in \mathbb{R}^{m\times m}$ is the noise covariance for the latent states. $C \in \mathbb{R}^{n\times m}$ links the observation with the states and Σ is the covariance of the observation noise. Under the linear Gaussian assumption, inference is fairly easy since both the posterior $p(\mathbf{z}|\mathbf{x})$ and the likelihood $\int p(\mathbf{x}, \mathbf{z})d\mathbf{z}$ are analytical. However, in real data analysis both the linearity assumption and the Gaussian assumption can break, which calls for more general assumptions that lose the tractability. Below we discuss the inference techniques for latent variable models.

1.4 Statistical inference for latent variable models

A common model fitting procedure for statistical models is maximum likelihood estimation (MLE), which optimizes the log likelihood function

$$\hat{\theta} = \arg\max l(\theta), \tag{1.6}$$

where the log likelihood function $l(\theta)$ is the marginal log density of observation $\mathbf{x}$ given parameter θ

$$l(\theta) = \log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}. \tag{1.7}$$

Then the latent variable can be estimated by the posterior distribution of $\mathbf{z}$ given observations and model parameters

$$p_{\hat{\theta}}(\mathbf{z}|\mathbf{x}) = p_{\hat{\theta}}(\mathbf{x}, \mathbf{z}) / \int p_{\hat{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}. \tag{1.8}$$

A key challenge in model fitting for latent variable models is that in most cases computing the log likelihood function (Equation (1.7)) and posterior (Equation (1.8)) involves an intractable integration. The difficulty hinders the application of complicated latent variable models that represent the data more faithfully.

The classic and powerful way of fitting latent variable models is Expectation-Maximization (EM) algorithm proposed in Dempster *et al.* [1977]. EM algorithm tries to get the (local) maximum likelihood estimator by iteratively optimizing the posterior distribution (E-step) and the model parameters (M-step). Specifically, for iteration k, given the current parameter estimator $\theta^{(k)}$, EM algorithm proceeds by

- E-step: getting the posterior distribution given the current parameter estimation $q_k(\mathbf{z}) = p_{\theta^{(k)}}(\mathbf{z}|\mathbf{x})$ and compute the expected value of log-likelihood given the posterior distribution

$$Q(\theta|\theta^{(k)}) = E_{q_k(\mathbf{z})} \log p_\theta(\mathbf{x}, \mathbf{z}); \tag{1.9}$$

- M-step: maximizing this conditional expectation

$$\theta^{(k+1)} = \arg\max_\theta Q(\theta|\theta^{(k)}). \tag{1.10}$$

And stop until certain convergence criteria are met.

By decomposing the complicated likelihood term in Equation (1.7) into two easier steps, EM algorithm facilitates the inference for a large class of latent variable models. However, the tractability of EM algorithm depends on the tractability of $Q(\theta|\hat{\theta})$. When model lacks conjugacy, it is usually hard to compute both the posterior distribution $p_{\theta^{(k)}}(\mathbf{z}|\mathbf{x})$ and $Q(\theta|\theta^{(k)})$. Extensions of EM algorithm have been proposed that approximates E-step by Laplace approximation [Shun and McCullage, 1995] or Markov chain Monte Carlo (MCMC) algorithms [Wei and Tanner, 1990]. Laplace approximation approximates the posterior by a multivariate Gaussian distribution with mean as the mode of the log likelihood and variance as the inverse Hessian of the log density at mode, which can be inaccurate when true posterior is skewed or has a heavy tail. Also, integrating the log likelihood with respect to a Gaussian distribution can still be hard for complicated models. MCMC algorithms construct Markov chains whose stationary distribution is the posterior distribution, and use a Monte Carlo estimator to estimate $Q(\theta|\hat{\theta})$. The method is generic but can be computationally intensive when latent variable is of high dimension or evaluating log likelihood is hard. It is also hard to diagnose the mixing of the chain.

Variational inference is a flexible inference framework that alleviates these issues [Wainwright *et al.*, 2008]. The idea is to approximate the posterior distribution by a tractable distribution family $q(\mathbf{z}) \in \mathcal{Q}$ called variational distribution family, and optimize an objective function that is a lower bound of the log-likelihood called evidence lower bound (ELBO), which is a function of both the variational distribution q and the model parameter θ. The vanilla variational inference tries to maximize the following ELBO,

$$\mathrm{ELBO}(q,\theta) = E_q[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z})] = \log p_\theta(\mathbf{x}) - \mathrm{KL}(q(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})) \leq l(\theta), \quad (1.11)$$

where $\mathrm{KL}(q(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})) = E_{q(\mathbf{z})}[\log q(\mathbf{z}) - \log p_\theta(\mathbf{z}|\mathbf{x})]$, the KL-divergence between $q(\mathbf{z})$ and $p_\theta(\mathbf{z}|\mathbf{x})$, is the gap between ELBO and the log likelihood. If we allow $\mathcal{Q}$ to be any arbitrary distribution, then the optimum will coincide with the true posterior, in which case maximizing ELBO is equivalent to maximizing the likelihood.

One way of optimizing the ELBO is by (block) coordinate ascent, where q and θ are optimized iteratively, leading to Variational Bayes Expectation Maximization (VBEM) algorithm. Given the current parameter estimator $\theta^{(k)}$ and posterior approximation $q^{(k)}$, VBEM algorithm proceeds by

- E-step: optimizing the ELBO with respect to q

$$q^{(k+1)} = \arg\max_{q \in \mathcal{Q}} \mathrm{ELBO}(q, \theta^{(k)}); \tag{1.12}$$

- M-step: optimizing the ELBO with respect to θ

$$\theta^{(k+1)} = \arg\max_{\theta} \mathrm{ELBO}(q^{(k+1)}, \theta). \tag{1.13}$$

Note that when $\mathcal{Q}$ is assumed to be all the distributions, we recover the classic EM algorithm. Clever choice of $\mathcal{Q}$ is important for conducting variational inference. Larger set of $\mathcal{Q}$ would allow for a more accurate approximation of the posterior, usually at the expense of more computational burden. A common choice is to approximate posterior by the family of all independent distributions, which is also called mean-field approximation. For certain conjugate models, mean-field approximation can have analytical solution [Wainwright *et al.*, 2008]. Another common approximation is multivariate Gaussian distribution. An application and extension of VBEM is discussed in chapter 2

The flexibility of variational inference has spurred a huge amount of interest in

the past few years. Efforts have been made to allow variational inference to handle nonconjugacy [Emtiyaz Khan *et al.*, 2013; Blei *et al.*, 2012], scale to large dataset by incorporating stochastic optimization ideas [Hoffman *et al.*, 2013; Kingma and Welling, 2013], allow for richer class of variational distribution family [Rezende and Mohamed, 2015; Kingma *et al.*, 2016], and explore variants of the ELBO formulation [Burda *et al.*, 2015; Li and Turner, 2016]. Here we introduce Auto-encoding variational inference (AEVB) framework [Kingma and Welling, 2013; Rezende *et al.*, 2014; Titsias and Lázaro-Gredilla, 2014], a recently proposed variational inference technique that is flexible and scalable.

Auto-encoding variational inference uses both a generative model (or the probabilistic decoder) $p_\theta(\mathbf{x}, \mathbf{z})$ parameterized by θ, which models the generative process of the data through latent variables, and a recognition model $q_\phi(\mathbf{z}|\mathbf{x})$ (or the probabilistic encoder) parameterized by ϕ, which maps the observation to an approximate posterior distribution of the latent variables. The inference procedure involves jointly optimizing the parameters θ and ϕ by optimizing the ELBO.

$$\max_{\theta,\phi} \mathrm{ELBO}(\phi, \theta), \tag{1.14}$$

where

$$\mathrm{ELBO}(\phi, \theta) = \max_{\theta,\phi} E_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right]. \tag{1.15}$$

Two ideas makes AEVB attractive for large-scale data analysis with complicated models. The first idea is amortized inference enabled by stochastic optimization. Considering an example where the dataset $\mathbf{x} = \{\mathbf{x}^{(i)} \in \mathbb{R}^n\}_{i=1}^N$ consists of N i.i.d. continuous observation, we assume that each of the $\mathbf{x}^{(i)}$ are related to a continuous latent variable $\mathbf{z}^{(i)} \in \mathbb{R}^m$ following a prior distribution $p_\theta(\mathbf{z}^{(i)})$ with conditional dis-

tribution $p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})$ (both $p_\theta(\mathbf{z}^{(i)})$ and $p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})$ are shared across i). The joint distribution has the form

$$\log p_\theta(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}) = \sum_{i=1}^{N} \left[\log p_\theta(\mathbf{z}^{(i)}) + \log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})\right]. \tag{1.16}$$

AEVB parameterizes the posterior $p_\theta(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$ by mapping $\mathbf{x}^{(i)}$ to a distribution of $\mathbf{z}^{(i)}$, resulting in a recognition model $q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$. An example for the recognition model would be a multivariate Gaussian distribution whose mean and variance are functions of the observation $\mathbf{x}^{(i)}$,

$$q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)}) = \mathcal{N}(\mu_\phi(\mathbf{x}^{(i)}), \Sigma_\phi(\mathbf{x}^{(i)})). \tag{1.17}$$

where $\mu_\phi : \mathbb{R}^n \to \mathbb{R}^m$ and $\Sigma_\phi : \mathbb{R}^n \to \mathbb{R}^{m\times m}$ can be neural networks with parameter ϕ. In this case the ELBO can be decomposed into a summation of the ELBO for each observation,

$$\text{ELBO}(\phi, \theta) = \sum_{i=1}^{N} \text{ELBO}_i(\phi, \theta), \tag{1.18}$$

where

$$\text{ELBO}_i(\phi, \theta) = E_{q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})} \left[\log p_\theta(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}) - \log q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})\right]. \tag{1.19}$$

This leads naturally to an unbiased approximation of the full ELBO using a sub-sample of the data

$$\text{ELBO}(\phi, \theta) \approx \frac{N}{M} \sum_{j=1}^{M} \text{ELBO}_{i_j}(\phi, \theta), \tag{1.20}$$

Where $i_1, ..., i_M \in \{1, ..., N\}$ is a set of randomly selected index. Therefore, a gradient of the sub-sampled version of the ELBO would also be an unbiased estimator of

the gradient of the full ELBO, leading naturally to the application of stochastic optimization [Robbins and Monro, 1951; Zeiler, 2012; Kingma and Ba, 2014].

The second idea is the "reparameterization trick", a generic way of getting an unbiased gradient of ELBO. For all but the simplest cases, computing the gradient of ELBO, which involves integrating over q_ϕ, is intractable. While there exists a large area of research on getting a low-variance Monte Carlo estimate of the gradient [Burda *et al.*, 2015; Ranganath *et al.*, 2013], the reparameterization trick has been popular due to its good empirical performance and ease of implementation. The idea is to write $\mathbf{z}^{(i)}$ as the transformation of an easy to sample distribution $\epsilon^{(i)} \sim q_\epsilon$ parameterized by ϕ and $\mathbf{x}^{(i)}$, $\mathbf{z}^{(i)} = g_\phi(\epsilon^{(i)}; \mathbf{x}^{(i)})$. Now ELBO_i can be written as an expectation over $\epsilon^{(i)}$, which is independent of ϕ,

$$\text{ELBO}_i = E_{\epsilon^{(i)} \sim q_\epsilon} \left[\log p_\theta(\mathbf{x}^{(i)}, g_\phi(\epsilon^{(i)}; \mathbf{x}^{(i)})) - \log q_\phi(g_\phi(\epsilon^{(i)}; \mathbf{x}^{(i)}) | \mathbf{x}^{(i)}) \right] \tag{1.21}$$

When optimizing ELBO with gradient methods, equation (1.21) allows an unbiased estimator of the gradient by a Monte Carlo sample

$$\nabla \text{ELBO}_i \approx \frac{1}{L} \sum_{l=1}^{L} \left[\nabla \log p_\theta(\mathbf{x}^{(i)}, g_\phi(\epsilon^{(i,l)}; \mathbf{x}^{(i)})) - \nabla \log q_\phi(g_\phi(\epsilon^{(i,l)}; \mathbf{x}^{(i)}) | \mathbf{x}^{(i)}) \right], \tag{1.22}$$

where $\epsilon^{(i,l)}$ for $l = 1, ..., L$ are samples from q_ϵ. When $\mathbf{z}$ is assumed to be multivariate Gaussian (Equation (1.17)), a commonly used parameterization is $g_\phi(\epsilon; \mathbf{x}^{(i)}) = \mu_\phi(\mathbf{x}^{(i)}) + \Sigma_\phi^{1/2}(\mathbf{x}^{(i)})\epsilon^{(i)}$ where $\epsilon^{(i)}$ follows an m-dimensional standard Gaussian, $\mu_\phi : \mathbb{R}^n \to \mathbb{R}^m$ and $\Sigma_\phi^{1/2} : \mathbb{R}^n \to \mathbb{R}^{m \times m}$ are functions parameterized by ϕ. This gives $\mathbf{z}^{(i)} \sim \mathcal{N}\left(\mu_\phi, \Sigma_\phi^{1/2} \cdot (\Sigma_\phi^{1/2})^T\right)$.

Combining the reparameterization trick and the amortized inference idea, AEVB provides a fast and scalable inference scheme. An application and extension of the

AEVB framework is discussed in chapter 3.

1.5 Overview of the thesis

After providing the general background and an introduction of the key models and techniques, here we give an overview of the subsequent chapters of the thesis.

Chapter 2 incorporates a flexible count distribution family in state space models that gives a more faithful representation of the data. The default distribution used for modeling neural spike counts is Poisson distribution, which is simple but assumes the strong assumption that the mean and variance of the counts are the same. Neural data usually violates this assumption due to refractoriness, burstiness and so on. We propose a general count distribution family for neural spike count modeling and proposes variational Bayes expectation maximization method for model fitting. Our model is able to capture both the under-dispersion and the over-dispersion of the the spike counts and outperforms state space models with Poisson assumption.

Chapter 3 investigates the effect of nonlinear observation model in state space models. Most of the existing dimension reduction techniques for neural data use linear models or a limited form of nonlinearities, with the underlying assumption that neural data lie in a low-dimensional linear sub-space. We show that the complicated neural activities may be more concisely represented with nonlinear models. We extend recently proposed auto-encoding variational Bayes method to develop scalable and flexible inference method. Simulated and real data experiments are shown to illustrate the applicability of the methods in neural data analysis.

Chapter 4 introduces a fast method for region-of-interest (ROI) detection for calcium imaging data, a neuroimaging technique that enables whole brain recording on the cellular level. We formulate the ROI detection problem as a structured matrix

factorization problem. The data is represented as a matrix, where each column represents an image at a specific time. The goal is to decompose the matrix into a product of spatial components and temporal components. Each spatial component represents the shape and location of a neuron, and each temporal component represents the neural activity. We incorporate prior knowledge of neuron shape as constraints and regularizations in the matrix factorization and develop a greedy method for matrix factorization which provides fast result for ROI detection.

Chapter 5 develops a method for sampling from a complicated maximum entropy distribution. Maximum entropy principle states that given our partial knowledge of the data, represented as a set of expectation constraints, the distribution with maximum entropy that satisfies the constraints is the least biased distribution that represents our knowledge. The framework provides principled ways for formulating statistical models and creating null distribution for hypothesis testing. Given complicated constraints and high-dimensional space, it is highly non-trivial to obtain the maximum entropy distribution. Here we propose approximating maximum entropy distribution on continuous spaces by learning a smooth and invertible transformation that transforms a simple distribution to the desired maximum entropy distribution. We formulate the problem as a constrained optimization problem and propose stochastic optimization methods for solving the problem. We illustrate the flexibility and applicability of our method on simulated and real data examples.

Chapter 6 discuss methods proposed in the preceding chapters and the future work of modern neural data analysis.

Part I

Neural Population Data Analysis with Latent Variable Models

Chapter 2

Generalized Count Linear Dynamical System

Latent factor models have been widely used to analyze simultaneous recordings of spike trains from large, heterogeneous neural populations. These models assume the signal of interest in the population is a low-dimensional latent intensity that evolves over time, which is observed in high dimension via noisy point-process observations. These techniques have been well used to capture neural correlations across a population and to provide a smooth, denoised, and concise representation of high-dimensional spiking data. One limitation of many current models is that the observation model is assumed to be Poisson, which lacks the flexibility to capture under- and over-dispersion that is common in recorded neural data, thereby introducing bias into estimates of covariance. Here we develop the generalized count linear dynamical system, which relaxes the Poisson assumption by using a more general exponential family for count data. In addition to containing Poisson, Bernoulli, negative binomial, and other common count distributions as special cases, we show that this model can be tractably learned by extending recent advances in variational inference techniques.

We apply our model to data from primate motor cortex and demonstrate performance improvements over state-of-the-art methods, both in capturing the variance structure of the data and in held-out prediction.

This work, which was published as Gao *et al.* [2015], was jointly done with Lars Buesing, John Cunningham and Krishna Shenoy. Code can be found at `https://bitbucket.org/mackelab/pop_spike_dyn`.

2.1 Introduction

Many studies and theories in neuroscience posit that high-dimensional populations of neural spike trains are a noisy observation of some underlying, low-dimensional, and time-varying signal of interest. As such, over the last decade researchers have developed and used a number of methods for jointly analyzing populations of simultaneously recorded spike trains, and these techniques have become a critical part of the neural data analysis toolkit [Cunningham and Yu, 2014]. In the supervised setting, generalized linear models (GLM) have used stimuli and spiking history as covariates driving the spiking of the neural population [Paninski, 2004; Truccolo *et al.*, 2005; Pillow *et al.*, 2008; Stevenson *et al.*, 2008; Vidne *et al.*, 2012]. In the unsupervised setting, latent variable models have been used to extract low-dimensional hidden structure that captures the variability of the recorded data, both temporally and across the population of neurons [Kulkarni and Paninski, 2007; Yu *et al.*, 2009; Macke *et al.*, 2011; Petreska *et al.*, 2011; Pfau *et al.*, 2013; Buesing *et al.*, 2014].

In both these settings, however, a limitation is that spike trains are typically assumed to be conditionally Poisson, given the shared signal [Macke *et al.*, 2011; Pfau *et al.*, 2013; Buesing *et al.*, 2014]. The Poisson assumption, while offering algorithmic conveniences in many cases, implies the property of equal dispersion: the

conditional mean and variance are equal. This well-known property is particularly troublesome in the analysis of neural spike trains, which are commonly observed to be either over-dispersed or under-dispersed (variance greater than or less than the mean) [Churchland *et al.*, 2010b]. No doubly stochastic process with a Poisson observation can capture under-dispersion, and while such a model can capture over-dispersion, it must do so at the cost of erroneously attributing variance to the latent signal, rather than the observation process.

To allow for deviation from the Poisson assumption, some previous work has instead modeled the data as Gaussian [Yu *et al.*, 2009] or using more general renewal process models [Cunningham *et al.*, 2007; Adams *et al.*, 2009; Koyama, 2015]; the former of which does not match the count nature of the data and has been found inferior [Macke *et al.*, 2011], and the latter of which requires costly inference that has not been extended to the population setting. More general distributions like the negative binomial have been proposed [Goris *et al.*, 2014; Scott and Pillow, 2012; Linderman *et al.*, 2015], but again these families do not generalize to cases of under-dispersion. Furthermore, these more general distributions have not yet been applied to the important setting of latent variable models.

Here we employ a count-valued exponential family distribution that addresses these needs and includes much previous work as special cases. We call this distribution the *generalized count* (GC) distribution [del Castillo and Pérez-Casany, 2005], and we offer here four main contributions: *(i)* we introduce the GC distribution and derive a variety of commonly used distributions that are special cases, using the GLM as a motivating example (§2.2); *(ii)* we combine this observation likelihood with a latent linear dynamical systems prior to form a GC linear dynamical system (GCLDS; §2.3); *(iii)* we develop a variational learning algorithm by extending the current state-of-the-art methods from Emtiyaz Khan *et al.* [2013] to the GCLDS setting (§2.4); and

(iv) we show in data from the primate motor cortex that the GCLDS model provides superior predictive performance and in particular captures data covariance better than Poisson models (§2.6.2).

2.2 Generalized count distributions

We define the generalized count distribution as the family of count-valued probability distributions:

$$p_{\mathcal{GC}}(k;\theta,g(\cdot)) = \frac{\exp(\theta k + g(k))}{k!M(\theta,g(\cdot))}, \quad k \in \mathbb{N} \tag{2.1}$$

where $\theta \in \mathbb{R}$ and the function $g : \mathbb{N} \to \mathbb{R}$ parameterizes the distribution, and $M(\theta, g(\cdot)) = \sum_{k=0}^{\infty} \frac{\exp(\theta k+g(k))}{k!}$ is the normalizing constant. The primary virtue of the GC family is that it recovers all common count-valued distributions as special cases and naturally parameterizes many common supervised and unsupervised models (as will be shown); for example, the function $g(k) = 0$ implies a Poisson distribution with rate parameter $\lambda = \exp\{\theta\}$. Generalizations of the Poisson distribution have been of interest since at least Rao [1965], and the paper del Castillo and Pérez-Casany [2005] introduced the GC family and proved two additional properties: first, that the expectation of any GC distribution is monotonically increasing in θ, for a fixed $g(k)$; and second – and perhaps most relevant to this study – concave (convex) functions $g(\cdot)$ imply under-dispersed (over-dispersed) GC distributions. Furthermore, often desired features like zero truncation or zero inflation [Lambert, 1992; Singh, 1978] can also be naturally incorporated by modifying the $g(0)$ value. Thus, with θ controlling the (log) rate of the distribution and $g(\cdot)$ controlling the "shape" of the distribution, the GC family provides a rich model class for capturing the spiking statistics of neural data. Other discrete distribution families do exist, such as the Conway-

Maxwell-Poisson distribution [Sellers and Shmueli, 2010] and ordered logistic/probit regression [Ananth and Kleinbaum, 1997], but the GC family offers a rich exponential family, which makes computation somewhat easier and allows the $g(\cdot)$ functions to be interpretable.

Figure 2.1 demonstrates the relevance of modeling dispersion in neural data analysis. The left panel shows a scatterplot where each point is an individual neuron in a recorded population of neurons from primate motor cortex (experimental details will be described in §2.6.2). Plotted are the mean and variance of spiking activity of each neuron; activity is considered in 20ms bins. For reference, the equi-dispersion line implied by a homogeneous Poisson process is plotted in red, and note further that all doubly stochastic Poisson models would have an implied dispersion *above* this Poisson line. These data clearly demonstrate meaningful under-dispersion, underscoring the need for the present advance. The right panel demonstrates the appropriateness of the GC model class, showing that a convex/linear/concave function $g(k)$ will produce the expected over/equal/under-dispersion. Given the left panel, we expect under-dispersed GC distributions to be most relevant, but indeed many neural datasets also demonstrate over and equi-dispersion [Churchland *et al.*, 2010b], highlighting the need for a flexible observation family.

To illustrate the generality of the GC family and to lay the foundation for our unsupervised learning approach, we consider briefly the case of supervised learning of neural spike train data, where generalized linear models (GLM) have been used extensively [Pillow *et al.*, 2008; Paninski *et al.*, 2007; Scott and Pillow, 2012]. We define GCGLM as that which models a single neuron with count data $x_i \in \mathbb{N}$, and associated covariates $z_i \in \mathbb{R}^p (i = 1, ..., n)$ as

$$x_i \sim \mathcal{GC}(\theta(z_i), g(\cdot)), \quad \text{where} \quad \theta(z_i) = z_i\beta. \tag{2.2}$$

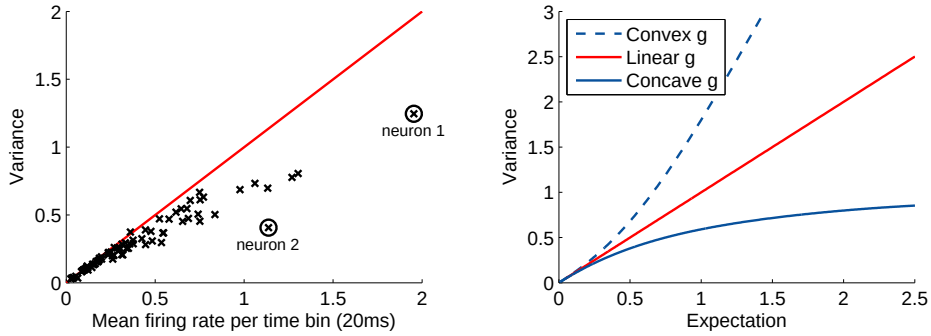

Figure 2.1: *Left panel*: mean firing rate and variance of neurons in primate motor cortex during the peri-movement period of a reaching experiment (see §2.6.2). The data exhibit under-dispersion, especially for high firing-rate neurons. The two marked neurons will be analyzed in detail in Figure 2.2. *Right panel*: the expectation and variance of the GC distribution with different choices of the function g

Here $\mathcal{GC}(\theta, g(\cdot))$ denotes a random variable distributed according to (2.1), $\beta \in \mathbb{R}^p$ are the regression coefficients. This GCGLM model is highly general. Table 1 shows that many of the commonly used count-data models are special cases of GCGLM, by restricting the $g(\cdot)$ function to have certain parametric form. In addition to this convenient generality, one benefit of our parametrization of the GC model is that the curvature of $g(\cdot)$ directly measures the extent to which the data deviate from the Poisson assumption, allowing us to meaningfully interrogate the form of $g(\cdot)$. Note that (2.2) has no intercept term because it can be absorbed in the $g(\cdot)$ function as a linear term αk (see Table 2.1).

Unlike previous GC work del Castillo and Pérez-Casany [2005], our parameterization implies that maximum likelihood parameter estimation (MLE) is a tractable convex program, which can be seen by considering:

$$(\hat{\beta}, \hat{g}(\cdot)) = \arg\max_{(\beta, g(\cdot))} \sum_{i=1}^{n} \log p(x_i) = \arg\max_{(\beta, g(\cdot))} \sum_{i=1}^{n} \left[(z_i\beta)x_i + g(x_i) - \log M(z_i\beta, g(\cdot))\right]. \tag{2.3}$$

First note that, although we have to optimize over a function $g(\cdot)$ that is defined on all non-negative integers, we can exploit the empirical support of the distribution to produce a finite optimization problem. Namely, for any k^* that is not achieved by any data point x_i (i.e., the count $\#\{i|x_i = k^*\} = 0$), the MLE for $g(k^*)$ must be $-\infty$, and thus we only need to optimize $g(k)$ for k that have empirical support in the data. Thus $g(k)$ is a finite dimensional vector. To avoid the potential overfitting caused by truncation of $g_i(\cdot)$ beyond the empirical support of the data, we can enforce a large (finite) support and impose a quadratic penalty on the second difference of g(.), to encourage linearity in $g(\cdot)$ (which corresponds to a Poisson distribution). Second, note that we can fix $g(0) = 0$ without loss of generality, which ensures model identifiability. With these constraints, the remaining $g(k)$ values can be fit as free parameters or as convex-constrained (a set of linear inequalities on $g(k)$; similarly for concave case). Finally, problem convexity is ensured as all terms are either linear or linear within the log-sum-exp function $M(\cdot)$, leading to fast optimization algorithms [Boyd and Vandenberghe, 2009].

2.3 Generalized count linear dynamical system model formulation

With the GC distribution in hand, we now turn to the unsupervised setting, namely coupling the GC observation model with a latent, low-dimensional dynamical system. Our model is a generalization of linear dynamical systems with Poisson likelihoods (PLDS), which have been extensively used for analysis of populations of neural spike trains [Macke *et al.*, 2011; Buesing *et al.*, 2014, 2012, 2015]. Denoting x_{rti} as the observed spike-count of neuron $i \in \{1, ..., N\}$ at time $t \in \{1, ..., T\}$ on experimental

Table 2.1: Special cases of GCGLM. For all models, the GCGLM parametrization for θ is only associated with the slope $\theta(x) = \beta x$, and the intercept α is absorbed into the $g(\cdot)$ function. In all cases we have $g(k) = -\infty$ outside the stated support of the distribution. Whenever unspecified, the support of the distribution and the domain of the $g(\cdot)$ function are non-negative integers $\mathbb{N}$.

Model Name	**Typical Parameterization**	**GCGLM Parametrization**
Logistic regression (e.g. Ananth and Kleinbaum [1997])	$P(x=k) = \frac{\exp(k(\alpha + z\beta))}{1+\exp(\alpha + x\beta)}$	$g(k) = \alpha k; k = 0, 1$
Poisson regression (e.g., Pillow *et al.* [2008]; Paninski *et al.* [2007])	$P(x=k) = \frac{\lambda^k}{k!}\exp(-\lambda);$ $\lambda = \exp(\alpha + z\beta)$	$g(k) = \alpha k$
Adjacent category regression (e.g., Ananth and Kleinbaum [1997])	$\frac{P(x=k+1)}{P(x=k)} = \exp(\alpha_k + z\beta)$	$g(k) = \sum_{i=1}^{k}(\alpha_{i-1} + \log i);$ $k = 0, 1, ..., K$
Negative binomial regression (e.g., Scott and Pillow [2012]; Linderman *et al.* [2015])	$P(x=k) = \frac{(k+r-1)!}{k!(r-1)!}(1-p)^r p^k$ $p = \exp(\alpha + z\beta)$	$g(k) = \alpha k + \log(k+r-1)!$
COM-Poisson regression (e.g., Sellers and Shmueli [2010])	$P(x=k) = \frac{\lambda^k}{(k!)^\nu} / \sum_{j=1}^{+\infty}\frac{\lambda^j}{(j!)^\nu}$ $\lambda = \exp(\alpha + z\beta)$	$g(k) = \alpha k + (1-\nu)\log k!$

trial $r \in \{1, ..., R\}$, the PLDS assumes that the spike activity of neurons is a noisy Poisson observation of an underlying low-dimensional latent state $\mathbf{z}_{rt} \in \mathbb{R}^p$,(where $p \ll N$), such that:

$$x_{rti}|\mathbf{z}_{rt} \sim \text{Poisson}\left(\exp\left\{c_i^\top \mathbf{z}_{rt} + \mathbf{d}_i\right\}\right). \tag{2.4}$$

Here $C = \begin{bmatrix} c_1 & ... & c_N \end{bmatrix}^\top \in \mathbb{R}^{N\times p}$ is the factor loading matrix mapping the latent state $\mathbf{z}_{rt}$ to a log rate, with time and trial invariant baseline log rate $\mathbf{d} \in \mathbb{R}^N$. Thus the vector $C\mathbf{z}_{rt} + \mathbf{d}$ denotes the vector of log rates for trial r and time t. Critically, the latent state $\mathbf{z}_{rt}$ can be interpreted as the underlying signal of interest that acts as the "common input signal" to all neurons, which is modeled *a priori* as a linear

Gaussian dynamical system (to capture temporal correlations):

$$\begin{aligned} \mathbf{z}_{r1} &\sim \mathcal{N}(\mu_1, Q_1) \\ \mathbf{z}_{r(t+1)}|\mathbf{z}_{rt} &\sim \mathcal{N}(A\mathbf{z}_{rt} + \mathbf{b}_t, Q), \end{aligned} \tag{2.5}$$

where $\mu_1 \in \mathbb{R}^p$ and $Q_1 \in \mathbb{R}^{p\times p}$ parameterize the initial state. The transition matrix $A \in \mathbb{R}^{p\times p}$ and innovations covariance $Q \in \mathbb{R}^{p\times p}$ parameterize the dynamical state update. The optional term $\mathbf{b}_t \in \mathbb{R}^p$ allows the model to capture a time-varying firing rate that is fixed across experimental trials. The PLDS has been widely used and has been shown to outperform other models in terms of predictive performance, including in particular the simpler Gaussian linear dynamical system [Macke *et al.*, 2011].

The PLDS model is naturally extended to what we term the generalized count linear dynamical system (GCLDS) by modifying equation (2.4) using a GC likelihood:

$$x_{rti}|\mathbf{z}_{rt} \sim \mathcal{GC}\left(c_i^\top \mathbf{z}_{rt}, g_i(\cdot)\right). \tag{2.6}$$

Where $g_i(\cdot)$ is the $g(\cdot)$ function in (2.1) that models the dispersion for neuron i. Similar to the GLM, for identifiability, the baseline rate parameter $\mathbf{d}$ is dropped in (2.6) and we can fix $g(0) = 0$. As with the GCGLM, one can recover preexisting models, such as an LDS with a Bernoulli observation, as special cases of GCLDS (see Table 2.1).

2.4 Inference and learning in GCLDS

As is common in LDS models, we use expectation-maximization to learn parameters $\Theta = \{A, \{\mathbf{b}_t\}_t, Q, Q_1, \mu_1, \{g_i(\cdot)\}_i, C\}$. Because the required expectations do not admit a closed form as in previous similar work [Macke *et al.*, 2011; Lawhern *et al.*,

2010], we required an additional approximation step, which we implemented via a variational lower bound. Below we detailed the VBEM algorithm we use for the model

2.4.1 E-step: variational inference with dual optimization

First, each E-step requires calculating $p(\mathbf{z}_r|\mathbf{x}_r, \Theta)$ for each trial $r \in \{1, ..., R\}$ (the conditional distribution of the latent trajectories $\mathbf{z}_r = \{\mathbf{z}_{rt}\}_{1\leq t\leq T}$, given observations $\mathbf{x}_r = \{x_{rti}\}_{1\leq t\leq T, 1\leq i\leq N}$ and parameter Θ). For ease of notation below we drop the trial index r. These posterior distributions are intractable, and in the usual way we make a normal approximation $p(\mathbf{z}|\mathbf{x}, \Theta) \approx q(\mathbf{z}) = \mathcal{N}(\mathbf{m}, V)$.

One simple way to achieve this is by Laplace approximation, i.e. we approximate posterior mean by the mode of the joint distribution $\mathbf{m} = \arg\max_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \Theta)$ and approximate posterior variance by the negative inverse Hessian of the log-likelihood evaluated at the mode $V = -(\nabla^2 \log p(\mathbf{z}|\mathbf{x}, \Theta))^{-1}|_{\mathbf{z}=\mathbf{m}}$. Laplace approximation is simple and fast, but does not have a theoretical guarantee and can be inaccurate.

Here we identify the optimal $(\mathbf{m}, V)$ by maximizing a variational Bayesian lower bound (the so-called evidence lower bound or "ELBO") over the variational parameters $\mathbf{m}, V$ as:

$$\begin{aligned}\mathcal{L}(\mathbf{m}, V) &= E_{q(\mathbf{z})}\left[\log\left(\frac{p(\mathbf{z}|\Theta)}{q(\mathbf{z})}\right)\right] + E_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z}, \Theta)] \qquad (2.7)\\ &= \frac{1}{2}\left(\log|V| - \mathrm{tr}[\Sigma^{-1}V] - (\mathbf{m}-\mu)^T\Sigma^{-1}(\mathbf{m}-\mu)\right) + \sum_{t,i} E_{q(\mathbf{z}_t)}[\log p(x_{ti}|\mathbf{z}_t)] + \text{const},\end{aligned}$$

which is the usual form to be maximized in a variational Bayesian EM (VBEM) algorithm [Buesing *et al.*, 2014]. Here $\mu \in \mathbb{R}^{pT}$ and $\Sigma \in \mathbb{R}^{pT\times pT}$ are the expectation and variance of $\mathbf{z}$ given by the LDS prior in (2.5). The first term of (2.7) is the

negative Kullback-Leibler divergence between the variational distribution and prior distribution, encouraging the variational distribution to be close to the prior. The second term involving the GC likelihood encourages the variational distribution to explain the observations well. The integrations in the second term are intractable (this is in contrast to the PLDS case, where all integrals can be calculated analytically [Buesing *et al.*, 2014]). Below we use the ideas of Emtiyaz Khan *et al.* [2013] to derive a tractable, further lower bound. Here the term $E_{q(\mathbf{z}_t)}[\log p(x_{ti}|\mathbf{z}_t)]$ can be reduced to:

$$\begin{aligned} &E_{q(\mathbf{z}_t)}[\log p(x_{ti}|\mathbf{z}_t)] = E_{q(\eta_{ti})}\left[\log p_{\mathcal{GC}}(x|\eta_{ti}, g_i(\cdot))\right] \\ &= E_{q(\eta_{ti})}\left[x_{ti}\eta_{ti} + g_i(x_{ti}) - \log x_{ti}! - \log \sum_{k=0}^{K} \frac{1}{k!}\exp(k\eta_{ti} + g_i(k))\right], \end{aligned} \tag{2.8}$$

where $\eta_{ti} = c_i^T \mathbf{z}_t$. Denoting $\nu_{tik} = k\eta_{ti} + g_i(k) - \log(k!) = kc_i^T\mathbf{z}_t + g_i(k) - \log k!$, (2.8) is reduced to $E_{q(\nu)}[\nu_{tix_{ti}} - \log(\sum_{0\le k\le K}\exp(\nu_{tik}))]$. Since ν_{tik} is a linear transformation of $\mathbf{z}_t$, under the variational distribution ν_{tik} is also normally distributed $\nu_{tik} \sim \mathcal{N}(h_{tik}, \rho_{tik})$. We have $h_{tik} = kc_i^T\mathbf{m}_t + g_i(k) - \log k!, \rho_{tik} = k^2 c_i^T V_t c_i$, where $(\mathbf{m}_t, V_t)$ are the expectation and covariance matrix of $\mathbf{z}_t$ under variational distribution. Now we can derive a lower bound for the expectation by Jensen's inequality:

$$E_{q(\nu_{ti})}\left[\nu_{tix_{ti}} - \log \sum_k \exp(\nu_{tik})\right] \geq h_{tix_{ti}} - \log \sum_{k=1}^{K} \exp(h_{tik} + \rho_{tik}/2) =: f_{ti}(\mathbf{h}_{ti}, \rho_{ti}). \tag{2.9}$$

Combining (2.7) and (2.9), we get a tractable variational lower bound:

$$\mathcal{L}(\mathbf{m}, V) \geq \mathcal{L}^*(\mathbf{m}, V) = E_{q(\mathbf{z})}\left[\log\left(\frac{p(\mathbf{z}|\Theta)}{q(\mathbf{z})}\right)\right] + \sum_{t,i} f_{ti}(\mathbf{h}_{ti}, \rho_{ti}). \tag{2.10}$$

For computational efficiency, we complete the E-step by maximizing the new evidence lower bound $\mathcal{L}^*$ via its dual [Emtiyaz Khan *et al.*, 2013]. Full details are derived in section 2.4.4.

2.4.2 M-step: analytical form

The M-step then requires maximization of $\mathcal{L}^*$ over Θ. We have two sets of parameters to optimize in the M-step. One set is for the dynamical system $(A, \{\mathbf{b}_t\}_{t=1}^T, Q, Q_1, \mu_1)$, the other is for the observation $(C, \{g_i(\cdot)\}_i)$. Similar to the PLDS case, the set of parameters involving the latent Gaussian dynamics $(A, \{\mathbf{b}_t\}_{t=1}^T, Q, Q_1, \mu_1)$ can be optimized analytically [Macke *et al.*, 2011]. Then, the parameters involving the GC likelihood $(C, \{g_i\}_i)$ can be optimized efficiently via convex optimization techniques [Boyd and Vandenberghe, 2009].

The part of the likelihood about the dynamical system has the form

$$\begin{aligned}\mathcal{L}_2(A, Q, Q_1, \mu_1) = \sum_{r=1}^{R} E_{q(\mathbf{z}_r)} \Bigg[& -\frac{1}{2}(\mathbf{z}_{r1} - \mu_1)^T Q_1^{-1}(\mathbf{z}_{r1} - \mu_1) \\ & -\frac{1}{2}\sum_{t=1}^{T-1}(\mathbf{z}_{r(t+1)} - A\mathbf{z}_{rt} - \mathbf{b}_t)^T Q^{-1}(\mathbf{z}_{r(t+1)} - A\mathbf{z}_{rt} - \mathbf{b}_t) \\ & -\frac{1}{2}\log|Q_1| - \frac{T-1}{2}\log|Q| \Bigg]\end{aligned}$$

Since everything is quadratic with respect to $\mathbf{z}$, the expectation can be calculated analytically. Moreover, all the parameters can be optimized analytically in close

form.

$$
\begin{aligned}
\hat{\mu}_t =& \frac{1}{R}\sum_{r=1}^{R} E(\mathbf{z}_{r1}), t = 1, ..., T \\
\hat{Q}_0 =& \frac{1}{R}\sum_{r=1}^{R} \left[E(\mathbf{z}_{r1}) + (E(\mathbf{z}_{r1}) - \hat{\mu}_1)(E(\mathbf{z}_{r1}) - \hat{\mu}_1)^T\right] \\
\hat{A}^T =& \{\sum_{r=1}^{R}\sum_{t=1}^{T-1} E\left[(\mathbf{z}_{rt} - \hat{\mu}_t)(\mathbf{z}_{rt} - \hat{\mu}_t)^T\right]\}^{-1} \sum_{r=1}^{R}\sum_{t=1}^{T-1} E\left[(\mathbf{z}_{r(t+1)} - \hat{\mu}_{t+1})(\mathbf{z}_{rt} - \hat{\mu}_t)^T\right] \\
\hat{\mathbf{b}}_t =& \hat{\mu}_{t+1} - \hat{A}\hat{\mu}_t, t = 1, ..., T-1 \\
\hat{Q} =& \frac{1}{R(T-1)}\sum_{r=1}^{R}\sum_{t=1}^{T-1} E[(\mathbf{z}_{r(t+1)} - \hat{A}\mathbf{z}_{rt} - \hat{\mathbf{b}}_t)(\mathbf{z}_{r(t+1)} - \hat{A}\mathbf{z}_{rt} - \hat{\mathbf{b}}_t)^T]
\end{aligned}
$$

The part of the likelihood about the observation can be written as

$$
\begin{aligned}
\mathcal{L}_1(C, g) = \sum_{i=1}^{N} \Bigg[& \sum_{\substack{t=1,...,T \\ r=1,...,R}} y_{rti}(c_i^T m_{rt}) + g_i(y_{rti}) \\
& - \log(1 + \sum_{k=1}^{K} \frac{1}{k!} \exp(k(c_i^T m_{rt}) + g_i(k) + \frac{1}{2}k^2 c_i^T V_{rt} c_i)) \Bigg]
\end{aligned}
$$

This part is concave and can be optimized efficiently using convex optimization techniques.

2.4.3 Practical concerns

In practice we initialize our VBEM algorithm with a Laplace-EM algorithm, and we initialize each E-step in VBEM with a Laplace approximation, which empirically gives substantial runtime advantages, and always produces a sensible optimum. With the above steps, we have a fully specified learning and inference algorithm, which we now use to analyze real neural data.

2.4.4 Dual optimization for E-step

Below we detail the dual optimization we used in the E-step. This part is rather technical and can be skipped for dis-interested readers.

We first introduce the "vectorized" notation for the GCLDS model. Note that in the E-step the inference is separable across trials, so for ease of notation, we only consider one single trial and drop the trial index r. We assume N neurons observed during T time bins. Denote $\mathbf{z}_t$ as the p-dimensional latent variable and and $\mathbf{x}_t$ as the N-dimensional observation, respectively.

$$\mathbf{z} := \begin{pmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_T \end{pmatrix}, \mathbf{x} := \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{pmatrix}$$

The prior can be summarized as a multi-variate Gaussian distribution:

$$p(\mathbf{z}) = \mathcal{N}(\mu, \Sigma)$$

where

$$\mu = \begin{pmatrix} \mu_1 \\ A\mu_1 + b_1 \\ \vdots \\ A^{T-1}\mu_1 + \sum_{t=1}^{T-1} A^{T-1-t} b_t \end{pmatrix},$$

$$\Sigma^{-1} = \begin{pmatrix} Q_0^{-1} + A^T Q^{-1} A & A^T Q^{-1} & & \\ Q^{-1} A & Q^{-1} + A^T Q^{-1} A & A^T Q^{-1} & \\ & \ddots & \ddots & \ddots \end{pmatrix}.$$

The likelihood has the form

$$
\begin{aligned}
p(\mathbf{x}|\mathbf{z}) =& \prod_{t,i} p(x_{ti}|\eta_{ti}) \\
p(x_{ti}|\eta_{ti}) =& \mathcal{GC}(x_{ti}|\eta_{ti}, g_i(\cdot)) \\
\eta :=& W\mathbf{z} \\
W =& \text{blk-diag}(C, ..., C),
\end{aligned}
$$

where we stack all the η_{ti} in $\eta = (\eta_{11}, ..., \eta_{1N}, ..., \eta_{T1}, ..., \eta_{TN}) \in \mathbb{R}^{NT}$. The log likelihood reads:

$$
\begin{aligned}
\log p(\mathbf{z}, \mathbf{x}) \propto & -\frac{1}{2}(\mathbf{z}-\mu)^T\Sigma^{-1}(\mathbf{z}-\mu) + \sum_{t,i}[x_{ti}\eta_n + g_i(x_{ti}) - \log(\sum_k \frac{1}{k!}\exp(k\eta_{ti} + g_i(k)))] \\
& - \sum_{t,i}\log(x_{ti}!) - \frac{1}{2}\log|\Sigma|
\end{aligned}
$$

In the E-step we make a Gaussian approximation to the posterior:

$$
p(\mathbf{z}|\mathbf{x}) \approx q(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{m}, V).
$$

The variational lower bound reads:

$$
\begin{aligned}
\mathcal{L}(\mathbf{m}, V) =& \int q(\mathbf{z}) \log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} d\mathbf{z} \\
=& \frac{1}{2}(\log|V| - \text{tr}[\Sigma^{-1}V] - (\mathbf{m}-\mu)^T\Sigma^{-1}(\mathbf{m}-\mu)) \\
& + \sum_{t,i} E_{q(\eta_{ti})}[\log p(x_{ti}|\eta_{ti})] - \frac{1}{2}\log|\Sigma| + \frac{dT}{2}.
\end{aligned}
$$

Defining $\nu_{tik} = k\eta_{ti} + g_i(k) - \log k!$, we know that ν_{tik} is also normally distributed

under the variational distribution

$$\nu_{tik} \sim \mathcal{N}(h_{tik}, \rho_{tik}).$$

Therefore we can re-write the term $E_{q(x)}[\log p(x_{ti}|\eta_{ti})]$ and find a lower bound of the term by

$$\begin{aligned}
&E_{q(\eta_{ti})}\left[\log p(x_{ti}|\eta_{ti})\right] \\
=&E_{q(\eta_{ti})}\left[x_{ti}\eta_{ti} + g_i(x_{ti}) - \log(x_{ti}!) - \log(\sum_k \frac{1}{k!}\exp(k\eta_{ti} + g_i(k)))\right] \\
=&E_{q(\nu_{ti})}\left[\nu_{tix_{ti}} - \log(\sum_{k=0}^{K}\exp(\nu_{tik}))\right] \\
\geq&h_{tix_{ti}} - \log(\sum_{k=0}^{K} E_{q(\nu_{ti})}(\exp(\nu_{tik}))) \\
=&h_{tix_{ti}} - \log(\sum_{k=0}^{K}\exp(h_{nk} + \rho_{nk}/2))
\end{aligned}$$

where $\nu_{ti} = (\nu_{ti1}, ..., \nu_{tiK})$. We always have $\nu_{ti0} = \rho_{ti0} = 0$. For the other variables define

$$\nu = (\nu_{11}, \nu_{12}, ..., \nu_{1N}, ..., \nu_{T1}, ..., \nu_{TN})^T,$$

and define $\mathbf{h}$ and ρ similarly. We then have the constraints

$$\begin{aligned}
\mathbf{h} :=& \tilde{W}\mathbf{m} + \tilde{\mathbf{d}} \\
\rho :=& \text{diag}(\tilde{W}V\tilde{W}^T)
\end{aligned}$$

where

$$\tilde{W} = W \otimes (1, 2, ..., K)^T$$
$$\tilde{\mathbf{d}} = \mathbf{1}_{T\times 1} \otimes (g_1(1) - \log 1!, ..., g_1(K) - \log K!, ..., g_N(1) - \log 1!, ..., g_N(K) - \log K!)^T$$

where $\otimes$ is the Kronecker product. Applying this lower bound and setting $\nu_{ti0} = \rho_{ti0} = 0$, we get the evidence lower bound (ELBO)

$$\begin{aligned}\mathcal{L}^*(\mathbf{m}, V, \mathbf{h}, \rho) =& \frac{1}{2}(\log|V| - \mathrm{tr}[\Sigma^{-1}V] - (\mathbf{m}-\mu)^T\Sigma^{-1}(\mathbf{m}-\mu)) \\ &+ \sum_{t,i}\left[\mathbf{1}_{\{x_{ti}>0\}}h_{tix_{ti}} - \log(1 + \sum_{k=1}^{K}\exp(h_{tik} + \rho_{tik}/2))\right]\end{aligned}$$

the variational inference can now be cast as the optimization problem:

$$\begin{aligned}\max_{\mathbf{m},V,\mathbf{h},\rho} \quad & \mathcal{L}^*(\mathbf{m}, V, \mathbf{h}, \rho) \\ \text{subject to} \quad & V \succeq 0 \\ & \mathbf{h} = \tilde{W}\mathbf{m} + \tilde{\mathbf{d}} \\ & \rho = \mathrm{diag}(\tilde{W}V\tilde{W}^T)\end{aligned}$$

Following Emtiyaz Khan *et al.* [2013], we can solve the dual problem

$$\min_{\alpha,\lambda}\max_{\mathbf{m},V,\mathbf{h},\rho} L(\mathbf{m}, V, \mathbf{h}, \rho) + \alpha^T(\mathbf{h} - \tilde{W}\mathbf{m} - \tilde{\mathbf{d}}) + \frac{1}{2}\lambda^T(\rho - \mathrm{diag}(\tilde{W}V\tilde{W}^T)),$$

where $\alpha, \lambda \in \mathbb{R}^{TNK}$ are the Lagrange multipliers. The unique maximizer with respect to $(\mathbf{m}, V)$ is given by

$$\begin{aligned}\mathbf{m}^* =& \mu - \Sigma\tilde{W}^T\alpha \\ V^* =& B_\lambda^{-1} := (\Sigma^{-1} + \tilde{W}^T(\mathrm{diag}\lambda)\tilde{W})^{-1}\end{aligned}$$

Maximization over $(\mathbf{h}, \rho)$ is also available in close form. Collecting the term containing $(\mathbf{h}, \rho)$. for f^* to be finite, we need to enforce the constraint $\alpha_{tik} = \lambda_{tik} - \mathbf{1}_{\{x_{ti}=k\}}$. Therefore, we can express everything in terms of λ

$$\begin{aligned} f^*_{ti}(\lambda_{ti}) =& \max_{\mathbf{h},\rho} \alpha_{ti}^T \mathbf{h}_{ti} + \lambda_{ti}^T \rho_{ti}/2 + \left[\mathbf{1}_{\{x_{ti}>0\}} h_{tix_{ti}} - \log(1 + \sum_{k=1}^{K} \exp(h_{tik} + \rho_{tik}/2) \right] \\ =& \sum_{k=1}^{K} \lambda_{tik} \log \lambda_{tik} + (1 - \sum_{k=1}^{K} \lambda_{tik}) \log(1 - \sum_{k=1}^{K} \lambda_{tik}). \end{aligned}$$

Denoting $\tilde{y}_{ti} = (\mathbf{1}_{\{x_{ti}=1\}}, \mathbf{1}_{\{x_{ti}=2\}}, ..., \mathbf{1}_{\{x_{ti}=K\}})$ and $\tilde{y} = (\tilde{y}_{11}, ..., \tilde{y}_{1N}, ..., \tilde{y}_{T1}, ..., \tilde{y}_{TN})$, the dual problem is reduced to

$$\begin{aligned} \min_{\lambda} \quad & D(\lambda) \\ \text{subject to} \quad & \lambda_{tik} > 0 \\ & \sum_{k=1}^{K} \lambda_{tik} < 1, \ t = 1, ..., T, n = 1, ..., N, k = 1, ..., K \end{aligned}$$

where

$$D(\lambda) := \frac{1}{2}(\lambda - \tilde{y})^T \tilde{W} \Sigma \tilde{W}^T (\lambda - \tilde{y}) - (\tilde{W}\mu + \tilde{\mathbf{d}})^T (\lambda - \tilde{y}) - \frac{1}{2} \log |B_\lambda| + \sum_{t,i} f^*_{ti}(\lambda_{ti})$$

and the gradient of the dual reads

$$D'(\lambda) = \tilde{W} \Sigma \tilde{W}^T (\lambda - \tilde{y}) - \tilde{W}\mu - \tilde{\mathbf{d}} - \frac{1}{2} \text{diag}(W B_\lambda^{-1} W^T) - \sum_n f^{*\prime}_{ti}(\lambda_{ti})$$

2.5 Model evaluation by leave-one-neuron-out error

To evaluate the goodness-of-fit of the LDS models, we use leave-one-neuron-out predictive error. The idea is to split the data into training trials and testing trials. We

first use training trials to learn the model parameter Θ. Then for test data, each time we drop one neuron, use the other neurons to get the posterior distribution of the latent variables, and then compute the posterior distribution of the left out data. Specifically, denote $\mathbf{x}_r^i = (x_{r1i}, ..., x_{rTi})$ as the spike train of neuron i for trial r, $\mathbf{x}_r^{-i} = (\mathbf{x}_r^1, ..., \mathbf{x}_r^{i-1}, \mathbf{x}_r^{i+1}, ..., \mathbf{x}_r^n)$ as the spike train for trial r with neuron i left out. We compute the posterior distribution

$$p(x_{rti}|\mathbf{x}_r^{-i}) = \int p(\mathbf{x}_{rti}|\mathbf{z}_r)p(\mathbf{z}_r|\mathbf{x}_r^{-i})\mathbf{z}_r \tag{2.11}$$

where $p(\mathbf{z}_r|\mathbf{x}_r^{-i})$ is approximated with variational distribution and the integration can be reduced to a one-dimensional numerical integration since $\mathbf{x}_{rti}$ only depend on $\mathbf{z}_r$ through $c_i^T\mathbf{z}_{rt}$. We compute the mean square error of the predicted firing rate and the predictive log likelihood of the predicted firing rate.

2.6 Experiments

2.6.1 Simulation examples

To demonstrate the generality of the GCLDS and verify our algorithmic implementation, we first simulated four sets of data with binary, (nearly) Poisson, under-dispersed and over-dispersed observations by generating GCLDS model with 4 different g functions for the GC distribution.

- Binary: $g(k) = -1.9k$ for k=0,1;
- Nearly Poisson: $g(k) = -1.9k$ for k =0,1...,10;
- Underdispersed: $g(k) = -0.4k^2 + 1.5k$ for k =0,...,5;

- Overdispersed: $g(k) = 0.2k^2 - 2.1k$ for k=0,...,5.

Here we set $g(k) = -\infty$ when k is out of the scope of the definition, implying zero probability of generating an observation of k. Those g functions are selected to generate a small firing rate, which mimics the real neural data setting.

For each scenario we perform 50 independent experiments, each with 50 training trials and 10 testing trials. We randomly generate the model parameters and make sure that the latent signals are strong enough. For all simulations we use 30 neurons with 3 latent dimensions. For each neuron, we perturb its own g function by a random linear function to obtain a variable baseline rate. Here we compare the performance of PLDS and GCLDS, both with 3 latent dimensions. For GCLDS we restrict the $g_i(\cdot)$ functions to be the same across all neurons up to a linear function.

Table 2.2 shows the leave-one-neuron-out performance. We observe that GCLDS and PLDS has comparable MSE although GCLDS outperforms a little, which makes sense since even though PLDS has model specification, it can still capture the mean firing rate well as long as the data can be explained by low-dimensional subspace. In terms of log likelihood, GCLDS significantly outperforms PLDS for all but the nearly Poisson case.

2.6.2 Real data analysis

We analyze recordings of populations of neurons in the primate motor cortex during a reaching experiment (`G20040123`), details of which have been described previously [Yu *et al.*, 2009; Macke *et al.*, 2011]. In brief, a rhesus macaque monkey executed 56 cued reaches from a central target to 14 peripheral targets. Before the subject was cued to move (the *go* cue), it was given a preparatory period to plan the upcoming reach. Each trial was thus separated into two temporal epochs, each of which has been

Setting		Binary	Poisson	Underdispersed	Overdispersed
MSE	PLDS	0.124(0.001)	0.199(0.004)	0.156(0.001)	0.259(0.008)
	GCLDS	0.123(0.001)	0.198(0.004)	0.156(0.001)	0.246(0.004)
	Improve	0.001(0.000)	0.001(0.001)	0.000(0.000)	0.013(0.007)
NLL	PLDS	0.434(0.002)	0.481(0.003)	0.453(0.002)	0.517(0.003)
	GCLDS	0.397(0.002)	0.481(0.003)	0.449(0.002)	0.512(0.003)
	Improve	0.037(0.002)	-0.000(0.000)	0.004(0.000)	0.005(0.000)

Table 2.2: Simulation result for PLDS and GCLDS. Showing the leave-one-neuron-out mean square error (MSE) and negative log likelihood (NLL) for PLDS and GCLDS, as well as the improvement of GCLDS over PLDS. Results are averaged across 50 independent repeats with standard error showing in parentheses.

suggested to have their own meaningful dynamical structure [Petreska *et al.*, 2011; Churchland *et al.*, 2012]. We separately analyze these two periods: the preparatory period (1200ms period preceding the go cue), and the reaching period (50ms before to 370ms after the movement onset). We analyzed data across all 14 reach targets, and results were highly similar; in the following for simplicity we show results for a single reaching target (one 56 trial dataset). Spike trains were simultaneously recorded from 96 electrodes (using a Blackrock multi-electrode array). We bin neural activity at 20ms. To include only units with robust activity, we remove all units with mean rates less than 1 spike per second on average, resulting in 81 units for the preparatory period, and 85 units for the reaching period. As we have already shown in Figure 2.1, the reaching period data are strongly under-dispersed, even absent conditioning on the latent dynamics (implying further under-dispersion in the observation noise). Data during the preparatory period are particularly interesting due to its clear cross-correlation structure.

To fully assess the GCLDS model, we analyze four LDS models – *(i)* GCLDS-full: a separate function $g_i(\cdot)$ is fitted for each neuron $i \in \{1, ..., N\}$; *(ii)* GCLDS-simple:

a single function $g(\cdot)$ is shared across all neurons (up to a linear term modulating the baseline firing rate); *(iii)* GCLDS-linear: a truncated linear function $g_i(\cdot)$ is fitted, which corresponds to truncated-Poisson observations; and *(iv)* PLDS: the Poisson case is recovered when $g_i(\cdot)$ is a linear function on all nonnegative integers. In all cases we use the learning and inference of §2.4. We initialize the PLDS using nuclear norm minimization [Pfau *et al.*, 2013], and initialize the GCLDS models with the fitted PLDS. For all models we vary the latent dimension p from 2 to 8.

Analysis of the reaching period. Figure 2.2 compares the fits of the two neural units highlighted in Figure 2.1. These two neurons are particularly high-firing (during the reaching period), and thus should be most indicative of the differences between the PLDS and GCLDS models. The left column of Figure 2.2 shows the fitted $g(\cdot)$ functions the for four LDS models being compared. It is apparent in both the GCLDS-full and GCLDS-simple cases that the fitted g function is concave (though it was not constrained to be so), agreeing with the under-dispersion observed in Figure 2.1.

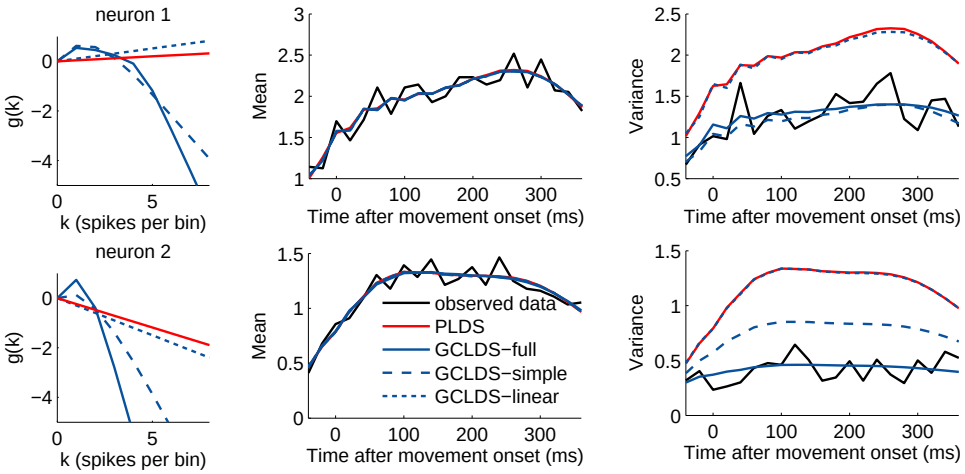

Figure 2.2: Examples of fitting result for selected high-firing neurons. Each row corresponds to one neuron as marked in left panel of Figure 2.1 – *left column*: fitted $g(\cdot)$ using GCLDS and PLDS; *middle and right column*: fitted mean and variance of PLDS and GCLDS. See text for details.

The middle column of Figure 2.2 shows that all four cases produce models that fit the mean activity of these two neurons very well. The black trace shows the empirical mean of the observed data, and all four lines (highly overlapping and thus not entirely visible) follow that empirical mean closely. This result is confirmatory that the GCLDS matches the mean and the current state-of-the-art PLDS.

More importantly, we have noted the key feature of the GCLDS is matching the dispersion of the data, and thus we expect it should outperform the PLDS in fitting variance. The right column of Figure 2.2 shows this to be the case: the PLDS significantly overestimates the variance of the data. The GCLDS-full model tracks the empirical variance quite closely in both neurons. The GCLDS-linear result shows that only adding truncation does not materially improve the estimate of variance and dispersion: the dotted blue trace is quite far from the true data in black, and indeed it is quite close to the Poisson case. The GCLDS-simple still outperforms the PLDS case, but it does not model the dispersion as effectively as the GPLDS-full case where each neuron has its own dispersion parameter (as Figure 2.1 suggests). The

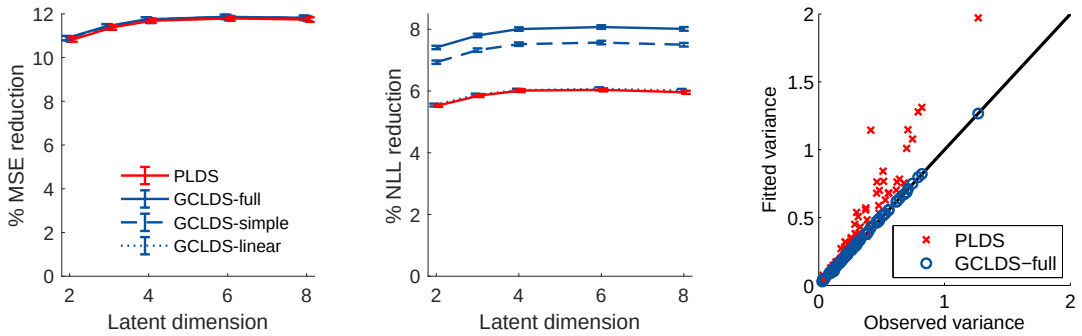

Figure 2.3: Goodness-of-fit for monkey data during the reaching period – *left panel*: percentage reduction of mean-squared-error (MSE) compared to the baseline (homogeneous Poisson process); *middle panel*: percentage reduction of predictive negative log likelihood (NLL) compared to the baseline; *right panel*: fitted variance of PLDS and GCLDS for all neurons compared to the observed data. Each point gives the observed and fitted variance of a single neuron, averaged across time.

natural next question is whether this outperformance is simply in these two illustrative neurons, or if it is a population effect. Figure 2.3 shows that indeed the population is much better modeled by the GCLDS model than by competing alternatives. The left and middle panels of Figure 2.3 show leave-one-neuron-out prediction error of the LDS models. For each reaching target we use 4-fold cross-validation and the results are averaged across all 14 reaching targets. Critically, these predictions are made for all neurons in the population. To give informative performance metrics, we defined baseline performance as a straightforward, homogeneous Poisson process for each neuron, and compare the LDS models with the baseline using percentage reduction of mean-squared-error and negative log likelihood (thus higher error reduction numbers imply better performance). The mean-squared-error (MSE; left panel) shows that the GCLDS offers a minor improvement (reduction in MSE) beyond what is achieved by the PLDS. Though these standard error bars suggest an insignificant result, a paired t-test is indeed significant ($p < 10^{-8}$). Nonetheless this minor result agrees with the middle column of Figure 2.2, since predictive MSE is essentially a measurement of the mean.

In the middle panel of Figure 2.3, we see that the GCLDS-full significantly outperforms alternatives in predictive log likelihood across the population ($p < 10^{-10}$, paired t-test). Again this largely agrees with the implication of Figure 2.2, as negative log likelihood measures both the accuracy of mean and variance. The right panel of Figure 2.3 shows that the GCLDS fits the variance of the data exceptionally well across the population, unlike the PLDS.

Analysis of the preparatory period. To augment the data analysis, we also considered the preparatory period of neural activity. When we repeated the analyses of Figure 2.3 on this dataset, the same results occurred: the GCLDS model produced

concave (or close to concave) g functions and outperformed the PLDS model both in predictive MSE (minority) and negative log likelihood (significantly). For brevity we do not show this analysis here. Instead, we here compare the temporal cross-covariance, which is also a common analysis of interest in neural data analysis [Macke *et al.*, 2011; Goris *et al.*, 2014; Cohen and Kohn, 2011] and, as noted, is particularly salient in preparatory activity. Figure 2.4 shows that GCLDS model fits both the temporal cross-covariance (left panel) and variance (right panel) considerably better than PLDS, which overestimates both quantities.

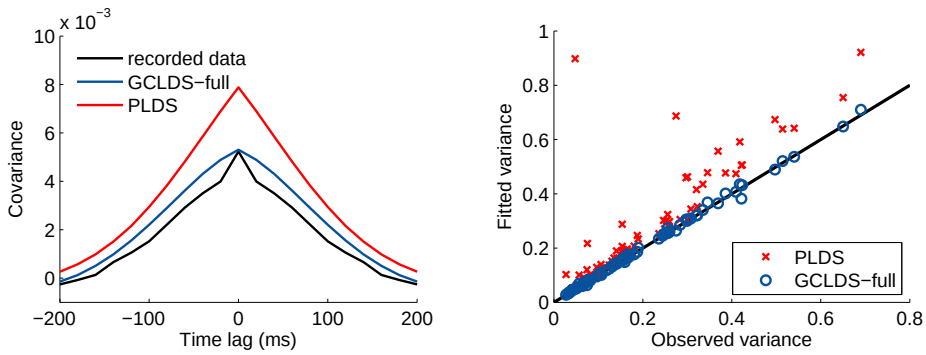

Figure 2.4: Goodness-of-fit for monkey data during the preparatory period – *Left panel:* Temporal cross-covariance averaged over all 81 units during the preparatory period, compared to the fitted cross-covariance by PLDS and GCLDS-full. *Right panel*: fitted variance of PLDS and GCLDS-full for all neurons compared to the observed data (averaged across time).

2.7 Discussion

In this paper we showed that the GC family better captures the conditional variability of neural spiking data, and further improves inference of key features of interest in the data. We note that it is straightforward to incorporate external stimuli and spike history in the model as covariates, as has been done previously in the Poisson case [Macke *et al.*, 2011]. Beyond the GCGLM and GCLDS, the GC family is also

extensible to other models that have been used in this setting, such as exponential family PCA [Pfau *et al.*, 2013] and subspace clustering [Buesing *et al.*, 2014]. The cost of this performance, compared to the PLDS, is an extra parameterization (the $g_i(\cdot)$ functions) and the corresponding algorithmic complexity. While we showed that there seems to be no empirical sacrifice to doing so, it is likely that data with few examples and reasonably Poisson dispersion may cause GCLDS to overfit.

Chapter 3

Linear Dynamical Neural Population Models Through Nonlinear Embeddings

A body of recent work in modeling neural activity focuses on recovering low-dimensional latent features that capture the statistical structure of large-scale neural populations. Most such approaches have focused on linear generative models, where inference is computationally tractable. Here, we propose fLDS, a general class of nonlinear generative models that permits the firing rate of each neuron to vary as an arbitrary smooth function of a latent, linear dynamical state. This extra flexibility allows the model to capture a richer set of neural variability than a purely linear model, but retains an easily visualizable low-dimensional latent space. To fit this class of non-conjugate models we propose a variational inference scheme, along with a novel approximate posterior capable of capturing rich temporal correlations across time. We show that our techniques permit inference in a wide class of generative models.We also show in application to two neural datasets that, compared to state-of-the-art neural pop-

ulation models, fLDS captures a much larger proportion of neural variability with a small number of latent dimensions, providing superior predictive performance and interpretability.

This work, which was published as Gao *et al.* [2016], was jointly done with Evan Archer, Liam Paninski, and John Cunningham. A Python/Theano [Bastien *et al.*, 2012; Bergstra *et al.*, 2010] implementation of our algorithms is available at `http://github.com/earcher/vilds`.

3.1 Introduction

Access to these high-dimensional neural data has spurred a search for new statistical methods. One recent approach has focused on extracting latent, low-dimensional dynamical trajectories that describe the activity of an entire population [Yu *et al.*, 2009; Macke *et al.*, 2011; Pfau *et al.*, 2013; Gao *et al.*, 2015]. The resulting models and techniques permit tractable analysis and visualization of high-dimensional neural data. Further, applications to motor cortex [Sadtler *et al.*, 2014; Churchland *et al.*, 2012, 2010a] and visual cortex [Goris *et al.*, 2014; Okun *et al.*, 2012; Ecker *et al.*, 2014] suggest that the latent trajectories recovered by these methods can provide insight into underlying neural computations.

Previous work for inferring latent trajectories has considered models with a latent linear dynamics that couple to observations either linearly, or through a restricted nonlinearity [Macke *et al.*, 2011; Gao *et al.*, 2015; Archer *et al.*, 2014]. When the true data generating process is nonlinear (for example, when neurons respond nonlinearly to a common, low-dimensional unobserved stimulus), the observation may lie in a low-dimensional nonlinear subspace that can not be captured using a mismatched observation model, hampering the ability of latent linear models to recover the low-

dimensional structure from the data. Here, we propose fLDS, a new approach to inferring latent neural trajectories that generalizes several previously proposed methods. As in previous methods, we model a latent dynamical state with a linear dynamical system (LDS) prior. But, under our model, each neuron's spike rate is permitted to vary as an arbitrary smooth nonlinear function of the latent state. By permitting each cell to express its own, private non-linear response properties, our approach seeks to find a nonlinear embedding of a neural time series into a linear-dynamical state space.

To perform inference in this nonlinear model we adapt recent advances in variational inference [Kingma and Welling, 2013; Titsias and Lázaro-Gredilla, 2014; Rezende *et al.*, 2014]. Using a novel approximate posterior that is capable of capturing rich correlation structure in time, our techniques can be applied to a large class of latent-LDS models. We show that our variational inference approach, when applied to learn generative models that predominate in the neural data analysis literature, perform comparably to inference techniques designed for a specific model. More interestingly, we show in both simulation and application to two neural datasets that our fLDS modeling framework yields higher prediction performance with a more compact latent representation, as compared to state-of-the-art neural population models. By freeing the latent space from representing the nonlinear response properties of each neuron, we are able to recover much cleaner and more structured representations of neural dynamics.

3.2 Notation and overview of neural data

Neuronal signals take the form of temporally fast ($\sim$ 1 ms) spikes that are typically modeled as discrete events. Although the spiking response of individual neurons has been the focus of intense research, modern experimental techniques make it possi-

ble to study the simultaneous activity of large numbers of neurons. In real data analysis, we usually discretize time into small bins of duration Δt and represent the response of a population of n neurons at time t by a vector $\mathbf{x}_t$ of length n, whose i^{th} entry represents number of spikes recorded from neuron i in time bin t, where $i \in \{1, \ldots, n\}$, $t \in \{1, \ldots, T\}$. Additionally, because spike responses are variable even under identical experimental conditions, it is commonplace to record many repeated trials, $r \in \{1, \ldots, R\}$, of the same experiment.

Here, we denote $\mathbf{x}_{rt} = (x_{rt1}, ..., x_{rtn})^\top \in \mathbb{N}^n$ as spike counts of n neurons for time t and trial r. When the time index is suppressed, we refer to a data matrix $\mathbf{x}_r = (\mathbf{x}_{r1}, ..., \mathbf{x}_{rT}) \in \mathbb{N}^{T \times n}$. We also use $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_R) \in \mathbb{N}^{T \times n \times R}$ to denote all the observations. We use analogous notation for other temporal variables; for instance $\mathbf{z}_r$ and $\mathbf{z}$.

3.3 Latent LDS neural population models with a linear rate function

Latent factor models are popular tools in neural data analysis, where they are used to infer low-dimensional, time-evolving latent trajectories (or factors) $\mathbf{z}_{rt} \in \mathbb{R}^m, m \ll n$ that capture a large proportion of the variability present in a neural population recording. Many recent techniques follow this general approach, with distinct noise models [Gao *et al.*, 2015], different priors on the latent factors [Yu *et al.*, 2009; Zhao and Park, 2016], extra model structure [Buesing *et al.*, 2014] and so on.

We focus upon one thread of this literature that takes its inspiration directly from the classical Kalman filter. Under this approach, the dynamics of a population of n neurons are modulated by an unobserved, linear dynamical system (LDS) with an

m-dimensional latent state $\mathbf{z}_{rt}$ that evolves according to,

$$\mathbf{z}_{r1} \sim \mathcal{N}(\mu_1, Q_1) \tag{3.1}$$

$$\mathbf{z}_{r(t+1)}|\mathbf{z}_{rt} \sim \mathcal{N}(\mathbf{A}\mathbf{z}_{rt}, Q), \tag{3.2}$$

where $\mathbf{A}$ is an $m \times m$ linear dynamics matrix, and the matrices Q_1 and Q are the covariances of the initial states and Gaussian innovation noise, respectively. The spike count observation is then related to the latent state via an observation model,

$$x_{rti}|\mathbf{z}_{rt} \sim \mathcal{P}_{\lambda,\ell}\left(\lambda = \ell([f(\mathbf{z}_{rt})]_i)\right). \tag{3.3}$$

where $[f(\mathbf{z}_{rt})]_i$ is the ith element of a deterministic "rate" function $f(\mathbf{z}_{rt}) : \mathbb{R}^m \to \mathbb{R}^n$, and $\mathcal{P}_{\lambda,\ell}(\lambda)$ is a noise model with parameter λ and link function ℓ. Each choice among the ingredients f and $\mathcal{P}_{\lambda,\ell}$ leads to a model with distinct characteristics. When $\mathcal{P}_{\lambda,\ell}$ is a Gaussian distribution with identity link and mean parameter λ, and f is linear, the model reduces to the classical Kalman filter. All operations in the Kalman filter are conjugate, and inference may be performed in closed form. However, any non-Gaussian noise model $\mathcal{P}_{\lambda,\ell}$ or nonlinear rate function f breaks conjugacy and necessitates the use of approximate inference techniques. This is generally the case for neural models, where the discrete, positive nature of spikes suggests the use of discrete noise models with positive link [Macke *et al.*, 2011; Gao *et al.*, 2015].

Examples of latent LDS models with a linear rate function: When $\mathcal{P}_{\lambda,\ell}$ is chosen to be Poisson with exponential link and $f(\mathbf{z}_{rt})$ couples linearly to the latent state, we recover the *Poisson linear dynamical system* model (PLDS) [Macke *et al.*,

2011],

$$x_{rti}|\mathbf{z}_{rt} \sim \text{Poisson}\left(\lambda_{rti} = \exp(c_i\mathbf{z}_{rt} + d_i)\right), \tag{3.4}$$

where c_i is the ith row of the $n \times m$ observation matrix $\mathbf{C}$ and $d_i \in \mathbb{R}$ is the baseline firing rate of neuron i. With $\mathcal{P}_{\lambda,\ell}$ chosen to be a generalized count (GC) distribution, identity link ℓ, and linear rate f, the model is called the *generalized count linear dynamical system* (GCLDS) [Gao *et al.*, 2015],

$$x_{rti}|\mathbf{z}_t \sim \mathcal{GC}\left(\lambda_{rti} = c_i\mathbf{z}_{rt}, g_i(\cdot)\right). \tag{3.5}$$

where $\mathcal{GC}(\lambda, g(\cdot))$ is a distribution family parameterized by $\lambda \in \mathbb{R}$ and a function $g(\cdot) : \mathbb{N} \to \mathbb{R}$, distributed as,

$$p_{\mathcal{GC}}(k; \lambda, g(\cdot)) = \frac{\exp(\lambda k + g(k))}{k!M(\lambda, g(\cdot))}. \quad k \in \mathbb{N} \tag{3.6}$$

where $M(\lambda, g(\cdot)) = \sum_{k=0}^{\infty} \frac{\exp(\lambda k + g(k))}{k!}$ is the normalizing constant. The GC model can flexibly capture under- and over-dispersed count distributions.

3.4 Nonlinear latent variable models for neural populations

We relax the linear assumptions of the previous LDS-based neural population models by incorporating a per-neuron rate function. We retain the latent LDS of Equation (3.1) and Equation (3.2), but select an observation model such that each neuron has

a separate nonlinear dependence upon the latent variable,

$$x_{rti}|\mathbf{z}_{rt} \sim \mathcal{P}_{\lambda,\ell}\left(\lambda_{rti} = \ell([f_\psi(\mathbf{z}_{rt})]_i)\right), \tag{3.7}$$

where $\mathcal{P}_{\lambda,\ell}(\lambda)$ is a noise model with parameter λ and link function ℓ; $f_\psi : \mathbb{R}^m \to \mathbb{R}^n$ is an arbitrary continuous function from the latent state into the spike rate; and $[f_\psi(\mathbf{z}_{rt})]_i$ is the ith element of $f_\psi(\mathbf{z}_{rt})$. In principle, the rate functions may be represented using any technique for function approximation. Here, we represent $f_\psi(\cdot)$ through a parametric neural network model. The parameters ψ then amount to the weights and biases of all units across all layers. For the remainder of the text, we use θ to denote all generative model parameters: $\theta = (\mu_1, Q_1, A, Q, \psi)$. We refer to this class of models as fLDS.

To refer to an fLDS with a given noise model $\mathcal{P}_{\lambda,\ell}$, we append the noise model to the acroynm. For the examples of fLDS we consider in the experiments, the link function ℓ may be absorbed into the rate function f; we omit it from the subsequent discussion. In the experiments, we will consider both PfLDS (taking $\mathcal{P}_{\lambda,\ell}$ to be Poisson) and GCfLDS (taking $\mathcal{P}_{\lambda,\ell}$ to be a generalized count distribution).

3.5 Inference by Auto-encoding variational Bayes

Our goal is to learn the model parameters θ and to infer the posterior distribution over the latent variables $\mathbf{z}$. Ideally, we would perform maximum likelihood estimation on the parameters, $\hat{\theta} = \arg\max_\theta \log p_\theta(\mathbf{x}) = \arg\max_\theta \sum_{r=1}^{R} \log \int p_\theta(\mathbf{x}_r, \mathbf{z}_r) \mathrm{d}\mathbf{z}_r$, and compute the posterior $p_{\hat{\theta}}(\mathbf{z}|\mathbf{x})$. However, under a fLDS neither the $p_\theta(\mathbf{z}|\mathbf{x})$ nor $p_\theta(\mathbf{x})$ are computationally tractable (both due to the noise model $\mathcal{P}$ and the nonlinear observation model $f_\psi(\cdot)$). As a result, we pursue a stochastic variational inference

approach to simultaneously learn parameters θ and infer the distribution of $\mathbf{z}$.

The strategy of variational inference is to approximate the intractable posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ by a tractable distribution $q_\phi(\mathbf{z}|\mathbf{x})$, which carries its own parameters ϕ.[1] With an approximate posterior[2] in hand, we learn both $p_\theta(\mathbf{z}, \mathbf{x})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ simultanously by maximizing the *evidence lower bound* (ELBO) of the marginal log likelihood:

$$\log p_\theta(\mathbf{x}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}) = \sum_{r=1}^{R} \mathcal{L}(\theta, \phi; \mathbf{x}_r) = \sum_{r=1}^{R} \mathbb{E}_{q_\phi(\mathbf{z}_r|\mathbf{x}_r)} \left[\log \frac{p_\theta(\mathbf{x}_r, \mathbf{z}_r)}{q_\phi(\mathbf{z}_r|\mathbf{x}_r)} \right]. \tag{3.8}$$

We optimize $\mathcal{L}(\theta, \phi; \mathbf{x})$ by stochastic gradient ascent, using a Monte Carlo estimate of the gradient $\nabla\mathcal{L}$. It is well-documented that Monte Carlo estimates of $\nabla\mathcal{L}$ are typically of very high variance, and strategies for variance reduction are an active area of research [Blei *et al.*, 2012; Ranganath *et al.*, 2013; Burda *et al.*, 2015].

Here, we take an auto-encoding variational Bayes (AEVB) approach [Kingma and Welling, 2013; Rezende *et al.*, 2014; Titsias and Lázaro-Gredilla, 2014] to estimate $\nabla\mathcal{L}$. In AEVB, we choose an easy-to-sample random variable $\epsilon \sim p(\epsilon)$ and sample $\mathbf{z}$ through a transformation of random sample ϵ parameterized by observations $\mathbf{x}$ and parameters ϕ: $\mathbf{z} = h_\phi(\mathbf{x}, \epsilon)$ to get a rich set of variational distributions $q_\phi(\mathbf{z}|\mathbf{x})$. We then use the unbiased gradient estimator on minibatches consisting of a randomly selected single trials $\mathbf{x}_r$,

$$\nabla\mathcal{L}(\theta, \phi; \mathbf{x}) \approx R\nabla\mathcal{L}(\theta, \phi; \mathbf{x}_r) \tag{3.9}$$

$$\approx R \left[\frac{1}{L} \sum_{l=1}^{L} \nabla \log p_\theta(\mathbf{x}_r, h_\phi(\mathbf{x}_r, \epsilon^l)) - \nabla\mathbb{E}_{q_\phi(\mathbf{z}_r|\mathbf{x}_r)} \left[\log q_\phi(\mathbf{z}_r|\mathbf{x}_r) \right] \right], \tag{3.10}$$

[1]Here, we consider a posterior $q_\phi(\mathbf{z}|\mathbf{x})$ that is conditioned explicitly upon $\mathbf{x}$. However, this is not necessary for variational inference.

[2]The approximate posterior is also sometimes called a "recognition model".

where ϵ^l are iid samples from $p(\epsilon)$. In practice, we evaluate the gradient in Equation (3.9) using a single sample from $p(\epsilon)$ ($L = 1$) and use ADADELTA for stochastic optimization [Zeiler, 2012].

The AEVB approach to inference is appealing in its generality: it is well-defined for a large class of generative models $p_\theta(\mathbf{x}, \mathbf{z})$ and approximate posteriors $q_\phi(\mathbf{z}|\mathbf{x})$. In practice, however, the performance of the algorithm has a strong dependence upon the particular structure of these models. In our case, we use an approximate posterior that is designed explicitly to parameterize a temporally correlated approximate posterior [Archer *et al.*, 2015]. We use a Gaussian approximate posterior,

$$q_\phi(\mathbf{z}_r|\mathbf{x}_r) = \mathcal{N}\left(\mu_\phi(\mathbf{x}_r), \Sigma_\phi(\mathbf{x}_r)\right), \tag{3.11}$$

where $\mu_\phi(\mathbf{x}_r)$ is a $mT \times 1$ mean vector and $\Sigma_\phi(\mathbf{x}_r)$ is a $mT \times mT$ covariance matrix. Both $\mu_\phi(\mathbf{x}_r)$ and $\Sigma_\phi(\mathbf{x}_r)$ are parameterized by observations $\mathbf{x}$ through a structured neural network. We can sample from this approximate by setting $p(\epsilon) \sim \mathcal{N}(0, I)$ and $h_\phi(\epsilon; \mathbf{x}_r) = \mu_\phi(\mathbf{x}_r) + \Sigma_\phi^{1/2}(\mathbf{x}_r)\epsilon$, where $\Sigma_\phi^{1/2}$ is the Cholesky decomposition of Σ_ϕ.

This approach is similar to that of Kingma and Welling [2013], except that we impose a block-tridiagonal structure upon the precision matrix ${\Sigma_\phi}^{-1}$ (rather than a diagonal covariance), which can express rich temporal correlations across time (essential for the posterior to capture the smooth, correlated trajectories typical of LDS posteriors), while remaining tractable with a computational complexity that scales linearly with T, the length of a trial.

Below we detail the parameterization for the recognition model $q_\phi(\mathbf{z}|\mathbf{x})$ we used

in fitting fLDS. We construct $q_\phi(\mathbf{z}|\mathbf{x})$ as a product of factors across time,

$$q_\phi(\mathbf{z}_r|\mathbf{x}_r) \propto q_\phi(\mathbf{z}_{r1}) \prod_{t=1}^{T-1} q_\phi(\mathbf{z}_{rt}|\mathbf{z}_{r(t-1)}) \prod_{t=1}^{T} q_\phi(\mathbf{z}_{rt}|\mathbf{x}_{rt}). \tag{3.12}$$

such that:

$$q_\phi(\mathbf{z}_{r1}) \sim \mathcal{N}(\tilde{\mu}_1, \tilde{Q}_1), \tag{3.13}$$

$$q_\phi(\mathbf{z}_{rt}|\mathbf{z}_{r(t-1)}) \sim \mathcal{N}(\tilde{A}\mathbf{z}_{r(t-1)}, \tilde{Q}), \tag{3.14}$$

$$q_\phi(\mathbf{z}_{rt}|\mathbf{x}_{rt}) \sim \mathcal{N}(m_{\tilde{\psi}}(\mathbf{x}_{rt}), c_{\tilde{\psi}}(\mathbf{x}_{rt})). \tag{3.15}$$

In our experiments we take $\tilde{\mu}_1 = 0$ although learning $\tilde{\mu}_1$ is also straightforward. The parameters $\tilde{A}$, $\tilde{Q}$ and $\tilde{Q}_1$ are $m \times m$ matrices that control the smoothness of the posterior, and are analogous to the LDS parameters appearing in Equation (3.1) and Equation (3.2). Functions $m_{\tilde{\psi}}(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ and $c_{\tilde{\psi}}(\cdot) : \mathbb{R}^n \to \mathbb{R}^{m\times m}$ are nonlinear functions of observations $\mathbf{x}_t \in \mathbb{R}^n$, parameterized by $\tilde{\psi}$. To ensure non-negative definiteness of $c_{\tilde{\psi}}(\mathbf{x}_{rt})$, we first map the observations $\mathbf{x}_t$ to the square root of the precision matrix. We parameterize a matrix-valued function $r_{\tilde{\psi}}(\cdot) : \mathbb{R}^n \to \mathbb{R}^{m\times m}$ by a feed-forward neural network, and set $c_{\tilde{\psi}}(\mathbf{x}_{rt}) = \left(r_{\tilde{\psi}}(\mathbf{x}_{rt}) r_{\tilde{\psi}}(\mathbf{x}_{rt})^T\right)^{-1}$. To summarize, the recognition model is parameterized by $\phi = (\tilde{\mu}_1, \tilde{A}, \tilde{Q}, \tilde{Q}_1, \tilde{\psi})$.

This product of Gaussian factors also has a Gaussian functional form, with block-tridiagonal inverse covariance. Normalizing recovers the multivariate Gaussian representation of Equation (3.11), where

$$\Sigma_\phi(\mathbf{x}_r) = \left(\mathbf{D}^{-1} + \mathbf{C}_\phi^{-1}(\mathbf{x}_r)\right)^{-1} \tag{3.16}$$

$$\mu_\phi(\mathbf{x}_r) = \left(\mathbf{D}^{-1} + \mathbf{C}_\phi^{-1}(\mathbf{x}_r)\right)^{-1} \mathbf{C}_\phi^{-1}(\mathbf{x}_r)\mathbf{M}_\phi(\mathbf{x}_r). \tag{3.17}$$

here $\mathbf{D} = (I - \tilde{\mathbf{A}})^{-\mathrm{T}}\tilde{\mathbf{Q}}(I - \tilde{\mathbf{A}})^{-1}$, where

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \tilde{Q}_1 & & & \\ & \tilde{Q} & & \\ & & \ddots & \\ & & & \tilde{Q} \end{bmatrix}, \quad \tilde{\mathbf{A}} = \begin{bmatrix} 0 & & & & \\ \tilde{A} & 0 & & & \\ & \tilde{A} & 0 & & \\ & & \ddots & \ddots & \\ & & & \tilde{A} & 0 \end{bmatrix}, \tag{3.18}$$

and

$$\mathbf{C}_{\tilde{\psi}}(\mathbf{x}_r) = \begin{bmatrix} c_{\tilde{\psi}}(\mathbf{x}_{r1}) & & & \\ & c_{\tilde{\psi}}(\mathbf{x}_{r2}) & & \\ & & \ddots & \\ & & & c_{\tilde{\psi}}(\mathbf{x}_{rT}) \end{bmatrix}, \quad \mathbf{M}_{\tilde{\psi}}(\mathbf{x}) = \begin{bmatrix} m_{\tilde{\psi}}(\mathbf{x}_{r1}) \\ \vdots \\ m_{\tilde{\psi}}(\mathbf{x}_{rT}) \end{bmatrix} \in \mathbb{R}^{mT} \tag{3.19}$$

3.6 Experiments

In all our experiments with PfLDS and GCfLDS, we parameterize $f_\phi(\cdot) : \mathbb{R}^m \to \mathbb{R}^n$ using a feed-forward neural network with 2 hidden layers, each containing 60 nodes using tanh nonlinearity. For PfLDS we transform the final output layer by an exponential function to ensure the positivity of the rate.

For the approximate posterior described in section 3.5. We parameterize $m_\phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ and $r_\phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^{m\times m}$ by a neural network with two hidden layers, each containing 60 nodes using tanh nonlinearity. Here the hidden layers are shared for $m_\phi(\cdot)$ and $r_\phi(\cdot)$.

3.6.1 Simulation examples

Linear dynamical system models with shared, fixed rate function: Our AEVB approach in principle permits inference in any latent LDS model. To illustrate this flexibility, we simulate 3 datasets from several previously-proposed models of neural responses. In our simulations, each data-generating model has a latent LDS state of $m = 2$ dimensions, as described by Equation (3.1) and Equation (3.2). Further, in all data-generating models, spike rates depend linearly on the latent state variable through a fixed link function f that is common across neurons. Each data-generating model has a distinct observation model (Equation (3.3)): Bernoulli, Poisson, or negative-binomial. We use the logistic link function for Bernoulli observations and use exponential link functions for the Poisson and negative-binomial distributions.

We compare PLDS and GCLDS model fits to each of these three datasets, using both our AEVB algorithm and two EM-based inference algorithms: LapEM (which approximates the conditional distribution of latent variable $p(\mathbf{z}|\mathbf{x})$ with a multivariate Gaussian by Laplace approximation in the E-step [Macke *et al.*, 2011; Gao *et al.*, 2015]) and VBDual (which approximates $p(\mathbf{z}|\mathbf{x})$ with a multivariate Gaussian by variational inference, through optimization in the dual space [Emtiyaz Khan *et al.*, 2013; Gao *et al.*, 2015]). Additionally, we fit PfLDS and GCfLDS models with our AEVB algorithm. On this linear simulated data we do not expect these nonlinear techniques to outperform linear methods. In all simulation studies we generate 20 training trials and 20 testing trials, with 100 simulated neurons and 200 time bins for each trial. In all our numerical experiments with PfLDS and GCfLDS, we use a neural network with 2 hidden layers, each containing 60 nodes using tanh nonlinearity, to parameterize $f_\psi(\cdot)$. Results are averaged across 10 independent repeats.

When training a model using the AEVB algorithm, we run 500 epochs before stopping. For LapEM and VBDual, we initialize with nuclear norm minimization [Pfau *et al.*, 2013] and stop either after 200 iterations or when the ELBO (scaled by number of time bins) increases by less than $\epsilon = 10^{-9}$ after one iteration.

We compare the predictive performance and running times of the algorithms in Table 3.1. For both PLDS and GCLDS, our AEVB algorithm gives results comparable to, though slightly worse than, the LapEM and VBEM algorithms. Although PfLDS and GCfLDS assume a much more complicated generative model, both provide comparable predictive performance and running time. We note that while LapEM is competitive in running time in this relatively small-data setting, the AEVB algorithm may be more desirable in a large data setting, where it can learn model parameters even before seeing the full dataset. In constrast, both LapEM and VBDual require a full pass through the data in the E-step before the M-step parameter updates.

Simulation with "grid cell" type response: A grid cell is a type of neuron that is activated when an animal occupies any vertex of a grid spanning the environment [Hafting *et al.*, 2005]. When an animal moves along a one-dimensional line in the space, grid cells exhibit oscillatory responses. Motivated by the response properties of grid cells, we simulated a population of 100 spiking neurons with oscillatory link functions and a shared, one-dimensional input $\mathbf{x}_{rt} \in \mathbb{R}$ given by,

$$\mathbf{x}_{r1} = 0, \tag{3.20}$$

$$\mathbf{x}_{r(t+1)} \sim \mathcal{N}(0.99\mathbf{x}_t, 0.01). \tag{3.21}$$

Table 3.1: Simulation results with a linear observation model: Each column contains results for a distinct experiment. For each generative model and inference algorithm (one per row), we report the one-step-ahead predictive log likelihood (PLL) and computation time (in minutes) of the model fit to each dataset.

		Bernoulli		Poisson		Negative-binomial	
Model	Inference	PLL	Time	PLL	Time	PLL	Time
PLDS	LapEM	-0.446	3	-0.385	5	-0.359	5
	VBDual	-0.446	157	-0.385	170	-0.359	138
	AEVB	-0.445	50	-0.387	55	-0.363	53
PfLDS	AEVB	-0.445	56	-0.387	58	-0.362	50
GCLDS	LapEM	-0.389	40	-0.385	97	-0.359	101
	VBDual	-0.389	131	-0.385	126	-0.359	127
	AEVB	-0.390	69	-0.386	75	-0.361	73
GCfLDS	AEVB	-0.390	72	-0.386	76	-0.361	68

The log firing rate of each neuron, indexed by i, is coupled to the latent variable $\mathbf{x}_{rt}$ through a sinusoid with a neuron-specific phase ϕ_i and frequency ω_i

$$y_{rit} \sim \text{Poisson}\left(\lambda_{rit} = \exp(2\sin(\omega_i x_{rt} + \phi_i) - 2)\right). \tag{3.22}$$

We generated ϕ_i uniformly at random in the region $[0, 2\pi]$ and set $\omega_i = 1$ for neurons with index $i \leq 50$ and $\omega_i = 3$ for neurons with index $i > 50$. We simulated 150 training and 20 testing trials, each with $T = 120$ time bins. We repeated this simulated experiment 10 times.

We compare performance of PLDS with PfLDS, both with 1-dimensional latent variable. As shown in Figure 3.1, PLDS is not able to adapt to the nonlinear and non-monotonic link function, and cannot recover the true latent variable (left panel and bottom right panel) or spike rate (upper right panel). On the other hand the PfLDS

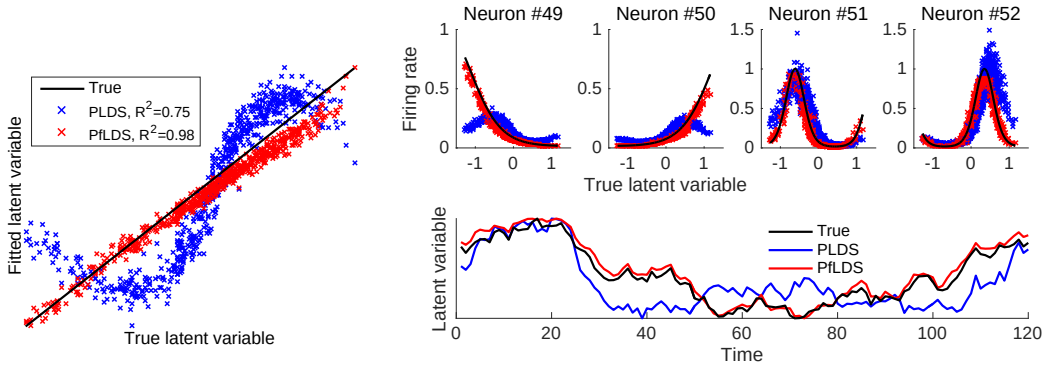

Figure 3.1: Sample simulation result with "grid cell" type response. *Left panel:* Fitted latent variable compared to true latent variable; *Upper right panel:* Fitted rate compared to the true rate for 4 sample neurons; *Bottom right panel:* Inferred trace of the latent variable compared to true latent trace. Note that the latent trajectory for a 1-dimensional latent variable is identifiable up to multiplicative constant, here we scale the latent variables to lie between 0 and 1.

model captures the nonlinearity well, recovering the true latent trajectory. The one-step-ahead predictive log likelihood (PLL) on a held-out dataset for PLDS is -0.622 (se=0.006), for PfLDS is -0.581 (se=0.006). A paired t-test for PLL is significant ($p < 10^{-6}$).

3.6.2 Real data analysis

We analyze two multi-neuron spike-train datasets, recorded from primary visual cortex and primary motor cortex of the Macaque brain, respectively. We find that fLDS models outperform PLDS in terms of predictive performance on held out data. Further, we find that the latent trajectories uncovered by fLDS are lower-dimensional and more structured than those recovered by PLDS.

Macaque V1 with drifting grating stimulus with single orientation: The dataset consists of 148 neurons simultaneously recorded from the primary visual cortex (area V1) of an anesthetized macaque, as described in Graf *et al.* [2011] (array 5).

Data were recorded while the monkey watched a 1280ms movie of a sinusoidal grating drifting in one of 72 orientations: (0°, 5°, 10°,...). Each of the 72 orientations was repeated $R = 50$ times. We analyze the spike activity from 300ms to 1200ms after stimulus onset. We discretize the data at $\Delta t = 10$ms, resulting in $T = 90$ timepoints per trial. Following Graf *et al.* [2011], we consider the 63 neurons with well-behaved tuning-curves. We performed both single-orientation and whole-dataset analysis.

We first use 12 equal spaced grating orientation (0°, 30°, 60°,...) and analyze each orientation separately. To increase sample size, for each orientation we pool data from the 2 neighboring orientations (e.g. for orientation 0°, we include data from orientation 5°and 355°), thereby getting 150 trials for each dataset (we find similar, but more variable, results when we do not include neighboring orientations). For each orientation, we divide the data into 120 training trials and 30 testing trials. For PfLDS we further divide the 120 training trials into 110 trials for fitting and 10 trials for validation (we use the ELBO on validation set to determine when to stop training). We do not include a stimulus model, but rather perform unsupervised learning to recover a low-dimensional representation that combines both internal and stimulus-driven dynamics.

We take orientation 0°as an example (the other orientations exhibit a similar pattern) and compare the fitted result of PLDS and PfLDS with a 2-dimensional latent space, which should in principle adequately capture the oscillatory pattern of the neural responses. We find that PfLDS is able to capture the nonlinear response charateristics of V1 complex cells (Figure 3.2(a), black line), while PLDS can only reliably capture linear responses (Figure 3.2(a), blue line). In Figure 3.2(b)(c) we project all trajectories onto the 2-dimensional latent manifold described by the PfLDS. We find that both techniques recover a manifold that reveals the rotational structure of the data; however, by offsetting the nonlinear features of the data into the observation

model, PfLDS recovers a much cleaner latent representation(Figure 3.2(c)).

We assess the model fitting quality by one-step-ahead prediction on a held-out dataset; we compare both percentage mean squared error (MSE) reduction and negative predictive log likelihood (NLL) reduction. We find that PfLDS recovers more compact representations than the PLDS, for the same performance in MSE and NLL. We illustrate this in Figure 3.2(d)(e), where PLDS requires approximately 10 latent dimensions to obtain the same predictive performance as an PfLDS with 3 latent dimensions. This result makes intuitive sense: during the stimulus-driven portion of the experiment, neural activity is driven primarily by a low-dimensional, oscillatory stimulus drive (the drifting grating). We find that the highly nonlinear generative models used by PfLDS lead to *lower*-dimensional and hence *more* interpretable latent-variable representations.

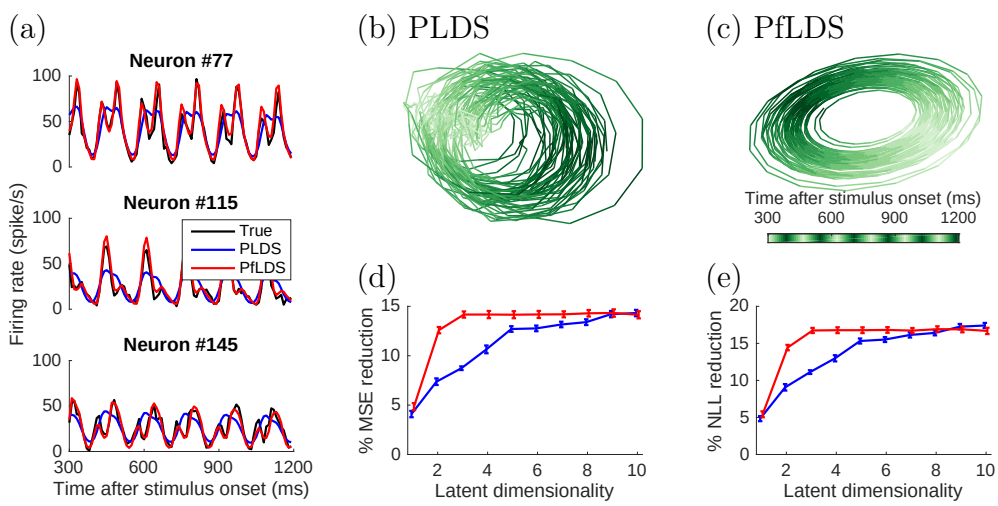

Figure 3.2: Results for fits to Macaque V1 data (single orientation) (a) Comparing true firing rate (black line) with fitted rate from PLDS (blue) and PfLDS (red) with 2 dimensional latent space for selected neurons (orientation 0°, averaged across all 120 training trials); (b)(c) 2D latent-space embeddings of 10 sample training trials, color denotes phase of the grating stimulus (orientation 0°); (d)(e) Predictive mean square error (MSE) and predictive negative log likelihood (NLL) reduction with one-step-ahead prediction, compared to a baseline model (homogeneous Poisson process). Results are averaged across 12 orientations.

To compare the performance of PLDS and PfLDS on the whole dataset, we use 10 trials from each of the 72 grating orientations (720 trials in total) as a training set, and 1 trial from each orientation as a test set. For PfLDS we further divide the 720 trials into 648 for fitting and 72 for validation. We observe in Figure 3.3(a)(b) that PfLDS again provides much better predictive performance with a small number of latent dimensions. We also find that for PfLDS with 4 latent dimensions, when we projected the observation into the latent space and take the first 3 principal components, the trajectory forms a torus (Figure 3.3(c)). Once again, this result has an intuitive appeal: just as the sinusoidal stimuli (for a fixed orientation, across time) are naturally embedded into a 2D ring, stimulus variation in orientation (at a fixed time) also has a natural circular symmetry. Taken together, the stimulus has a natural toroidal topology. We find that fLDS is capable of uncovering this latent structure. A video for the 3D embedding can be found at `https://www.dropbox.com/s/cluev4fzfsob4q9/video_fLDS.mp4?dl=0`

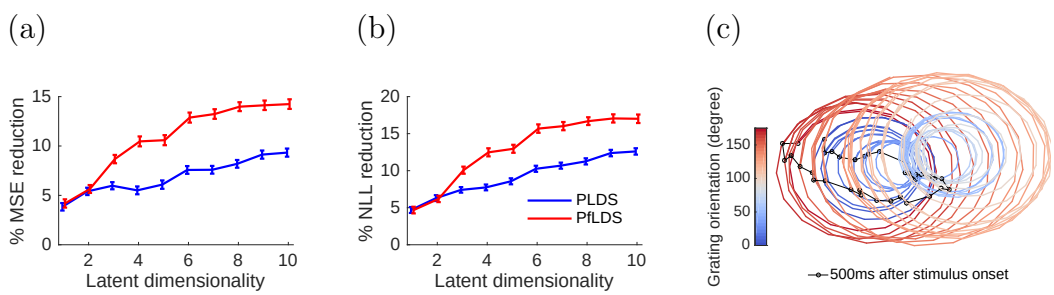

Figure 3.3: Macaque V1 data fitting result (full data) (a)(b) Predictive MSE and NLL reduction. (c) 3D embedding of the mean latent trajectory of the neuron activity during 300ms to 500ms after stimulus onset across grating orientations $0°, 5°, ..., 175°$, here we use PfLDS with 4 latent dimensions and then project the result on the first 3 principal components.

Macaque center-out reaching data: We analyzed the neural population data recorded from the Macaque motor cortex(`G20040123`), details of which can be found

in Yu *et al.* [2009]; Macke *et al.* [2011]. Briefly, the data consist of simultaneous recordings of 105 neurons for 56 cued reaches from the center of a screen to 14 peripheral targets. We analyze the reaching period (50ms before and 370ms after movement onset) for each trial. We discretize the data at $\Delta t = 20$ms, resulting in $T = 21$ timepoints per trial. For each target we use 50 training trials and 6 testing trials and fit all the 14 reaching targets together (making 700 training trials and 84 testing trials). We use both Poisson and GC noise models, as GC has the flexibility to capture the noted under-dispersion of the data [Gao *et al.*, 2015]. We compare both PLDS and PfLDS as well as GCLDS and GCfLDS fits. For both PfLDS and GCfLDS we further divide the training trials into 630 for fitting and 70 for validation.

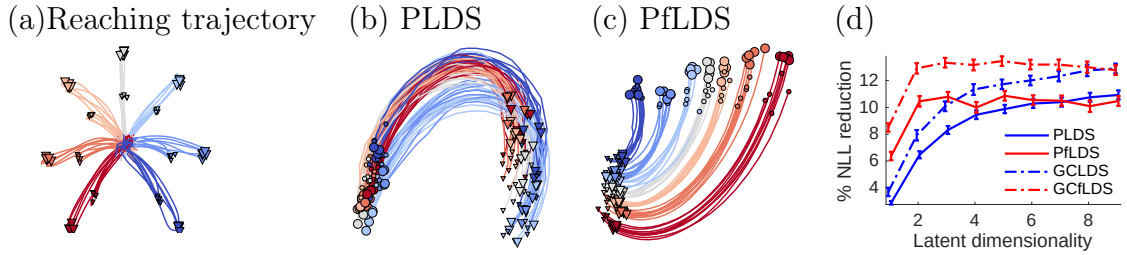

Figure 3.4: Macaque center-out reaching data analysis: (a) 5 sample reaching trajectory for each of the 14 target locations. Directions are coded by different color, and distances are coded by different marker size; (b)(c) 2D embeddings of neuron activity extracted by PLDS and PfLDS, circles represent 50ms before movement onset and triangles represent 340ms after movement onset. Here 5 training reaches for each target location are plotted; (d) Predictive negative log likelihood (NLL) reduction with one-step-ahead prediction.

As is shown in figure Figure 3.4(d), PfLDS and GCfLDS with latent dimension 2 or 3 outperforms their linear counterparts with much larger latent dimensions. We also find that GCLDS and GCfLDS models give much better predictive likelihood than their Poisson counterparts.On figure Figure 3.4(b)(c) we project the neural activities on the 2 dimensional latent space. We find that PfLDS (Figure 3.4(c)) clearly separates the reaching trajectories and orders them in exact correspondence with the

true the spatial location of the targets.

3.7 Discussion

We have proposed fLDS, a modeling framework for high-dimensional neural population data that extends previous latent, low-dimensional linear dynamical system models with a flexible, nonlinear observation model. Additionally, we described an efficient variational inference algorithm suitable for fitting a broad class of LDS models – including several previously-proposed models.

We illustrate in both simulation and application to real data that, even when a neural population is modulated by a low-dimensional linear dynamics, a latent variable model with a linear rate function fails to capture the true low-dimensional structure. In constrast, a fLDS can recover the low-dimensional structure, providing better predictive performance more interpretable latent-variable representations.

Our approach is distinct from related manifold learning methods [Roweis and Saul, 2000; Tenenbaum *et al.*, 2000].While most manifold learning techniques rely primarily on the notion of nearest neighbors, we exploit the temporal structure of the data by imposing strong prior assumption about the dynamics of our latent space. Further, in contrast to most manifold learning approaches, our approach includes an explicit generative model that lends itself naturally to inference and prediction, and allows for count-valued observations that account for the discrete nature of neural data.

Further, while an arbitrary nonlinear rate provides great flexibility, our inference approach also permits more structured generative models designed to account for applications with more background knowledge on the data generating process. Future work includes relaxing the latent linear dynamical system assumption to incorporate more flexible latent dynamics (for example, by using a Gaussian process prior [Zhao

and Park, 2016] or by incorporating a nonlinear dynamical phase space [Frigola *et al.*, 2014]). We also anticipate our approach may be useful in applications to neural decoding and prosthetics: once trained, our approximate posterior may be evaluated in close to real-time.

Part II

Region of Interest Detection for Calcium Imaging Data

Chapter 4

Region of Interest Detection for Calcium Imaging Data

The previous chapters aim at building statistical models to describe the structure of neural activities based on given spike train data. In real data analysis, extracting neural signal from noisy, indirect observations of the signal is an important and highly non-trivial first step.

Calcium imaging data is an optical imaging method that enables simultaneous recording of large neural populations at cellular resolution [Ahrens and Keller, 2013; Prevedel *et al.*, 2014]. The observation can be represented as (2D or 3D) movie data, where the neuron activity of a single neuron is represented as a spatially-localized fluorescent signal that varies across time. Manually identifying neurons from calcium imaging data can be time-consuming when a large neural population is recorded (currently on the scale of hundreds of thousands of neurons), and is also complicated by the overlapping of neural signals. Therefore, a principled way of automatically de-mixing neuron activities is highly desirable.

This chapter discusses a fast greedy method for detecting regions of interest (ROI)

from calcium imaging data based on structured matrix factorization formulation. The method is fast and effective, and can serve both as a stand-alone ROI detection method or as a good initialization of the more complicated method.

Part of the work described in this chapter is published as part of a larger joint work [Pnevmatikakis *et al.*, 2016] in collaboration with Eftychios A. Pnevmatikakis, Daniel Soudry, Timothy A. Machado, Josh Merel, David Pfau, Thomas Reardon, Yu Mu, Clay Lacefield, Weijian Yang, Misha Ahrens, Randy Bruno, Thomas M. Jessell, Darcy S. Peterka, Rafael Yuste, Liam Paninski. Code for the joint work can be found at `https://github.com/epnev/ca_source_extraction`

4.1 Introduction

The basic principle of calcium imaging is that the spiking activity of a neuron induce a transient increase in calcium concentration, which can be indirectly observed by recording the fluorescent properties of certain calcium indicators [Ahrens and Keller, 2013; Prevedel *et al.*, 2014]. The technique allows simultaneous recording from hundreds of thousands of neurons, providing crucial datasets for understanding the neural population behavior.

Recovering spike trains from calcium imaging data involves two inter-linked steps: ROI detection and calcium deconvolution. ROI detection refers to detecting the regions from the image data that correspond to neurons, while calcium deconvolution refers to recovering spike times from noisy calcium observation data. Here we focus on ROI detection, and propose a method that decomposes the calcium imaging data into neuron location and the corresponding calcium activity.

Due to the recent popularity of calcium imaging techniques, many methods have been proposed for ROI detection. One line of research directly uses the fact that pixels

belonging to the same neuron should have high temporal correlation, and identifies a pixel to be in the region of interest (ROI) when it's highly correlated with adjacent pixels [Smith and Häusser, 2010; Portugues *et al.*, 2014]. Intuitive and fast as this approach is, further steps are usually needed to separate individual neurons in the ROI. When the neurons are densely packed in the image, the identified ROI may almost cover the whole image, which doesn't provide much information.

Another line of research exploits the fact that neurons tend to have similar size and shape. Pachitariu *et al.* [2013] proposes a generative model which assumes that the shape of each neuron can be generated by a linear combination of several (localized) basis functions. Neuron identification is then performed by alternating between identifying location of the neurons by matching pursuit using given basis and tuning the shape of the bases using K-SVD or gradient descent. Though successfully exploiting the localized shape of neurons, the algorithm can only be used to analyze a single image (the mean image of the calcium imaging data) instead of the whole video, and therefore does not exploit the temporal structure of the data. When the neurons are densely packed in the image, the performance of the algorithm will deteriorate. Also, assuming a parametric form of the neuron shape can be too stringent.

Several attempts have been made for ROI detection using dictionary learning or matrix factorization techniques. Denote $X \in \mathbb{R}^{N \times T}$ to be the calcium imaging movie, where each column is a (vectorized) image with N pixels, the general idea is to decompose the matrix into $X \approx DA^T$, where each column of $D \in \mathbb{R}^{N \times K}$ contains the shape of a neuron and each column of $A \in \mathbb{R}^{T \times K}$ is the calcium activity of the corresponding neuron. Mukamel *et al.* [2009] proposes the PCA-ICA pipeline, which first uses principle component analysis (PCA) to de-noise the data and then uses independent component analysis (ICA) (together with other ad-hoc post-processing) to extract neuron locations. Diego *et al.* [2013] uses (online) matrix factorization

technique with sparsity penalties to learn the neuron shape and calcium activity. Maruyama *et al.* [2014] uses non-negative matrix factorization technique. By exploiting the fact that pixels belonging to the same neuron tend to fluoresce together, these algorithms achieve a certain amount of success. One common problem with this kind of approaches is that the extracted features are usually not localized. In real data, correlated neurons that tends to fire together are prevalent, and as a result, each dictionary element can contain several neurons, and one neuron may also appear in several elements. Another problem is the ability to determine the number of neurons automatically. Though certain heuristics have been proposed, the task is generally hard.

In this chapter we propose a greedy algorithm that combines matching pursuit with matrix factorization. In each iteration, we use the full temporal data to identify the region that has the most significant calcium activity over time. Then we apply regularized matrix factorization to the small patch to fine-tune the shape of the neuron. Experiments on simulated and real data show that the algorithm can automatically infer the location of neurons.

4.2 Algorithm

4.2.1 Problem formulation

Recall that X is a $N \times T$ matrix representing calcium imaging video, where each column is a (vectorized) frame that contains N pixels (when converted to the 2D images, the column has dimension $n_x \times n_y$). Assume there are K neurons in the

video, then we assume that the data can be decomposed into

$$X = DA^T + \epsilon,$$

where $D \in \mathbb{R}^{N \times K}$ represents the neuron shape, and $A \in \mathbb{R}^{T \times K}$ represents the calcium activities. $\epsilon \in \mathbb{R}^{N \times T}$ is the random noise. This leads to a natural optimization problem based on matrix factorization:

$$\min_{D \in \mathcal{D}, A \in \mathcal{A}} \|X - DA^T\|_2^2 + f_D(D) + f_A(A).$$

Here $\| \cdot \|_2$ is the Euclidean norm, $\mathcal{D}$ and $\mathcal{A}$ are the feasible sets for D and A that enforce certain hard constraint to the neuron shape and calcium activity. For example, Maruyama *et al.* [2014] restricts $\mathcal{D}$ and $\mathcal{A}$ to be non-negative, which reduces the problem into the well-established non-negative matrix factorization technique [Lee and Seung, 1999]. $f_D(\cdot)$ and $f_A(\cdot)$ are penalties that encourages certain structure in neuron shape and calcium trace. For example, Diego *et al.* [2013] enforces sparsity by adding Lasso penalty.

Natural and flexible as it is, current algorithms using matrix factorization usually ignore that the shape of each neuron is localized and smooth. As a result, each learnt dictionary element usually contains several neurons and post-processing is usually needed to separate the neurons. To enforce localization in matrix factorization technique, one possible way would be to add certain penalty to each column of D, as is proposed by Jenatton *et al.* [2009]. However, experiments shows that the learnt elements are usually not sparse enough and the algorithm tends to be unstable.

Instead of using penalty terms to enforce localization, we want to directly constrain the nonzero elements of each dictionary elements to be in a small region. Specifically,

denote $\mathcal{D}_w^+ \subset \mathcal{R}^N$ to be the set of all non-negative vectors whose nonzero elements lie in some $w \times w$ square (when converted to the 2D image), where w are set to be a little larger than the typical neuron diameter and are usually much smaller than the full image size. Now we would like to solve the optimization problem,

$$\begin{aligned} \underset{D,A}{\text{minimize}} \quad & \|X - DA^T\|_2^2 + f_D(D), \\ \text{subject to} \quad & D_k \in \mathcal{D}_w^+; k = 1, \ldots, K, \\ & \|A_k\|_2 \leq c_k, \end{aligned} \tag{4.1}$$

where D_k and A_k are the kth column of D and A, respectively. $f_D(\cdot)$ are some smooth penalty that will be discussed later in section 4.2.3. We restrict $\|A_k\|_2 \leq c_k$ to avoid degenerate solution of D. We can add more biological plausible constraints to A following Pnevmatikakis and Paninski [2013], but here we do not consider those constraints because the data we are dealing with have low temporal resolution, which makes the temporal structure less informative. Details of the constraints and regularizations will be discussed in section 4.2.3.

4.2.2 Greedy algorithm

The formulation of Equation (4.1) is similar to best-subset selection, which aims to choose the most significant covariates to do regression on dependent variable. Here we aim to choose the most significant K neuron locations to explain the variability of the whole movie. As is the case in best-subset selection, solving (4.1) exactly is challenging because the constraint $D_k \in \mathcal{D}_w^+, k = 1, \ldots, K$ is highly non-convex. Instead, we propose to approximate the solution using a greedy algorithm.

A description of the greedy algorithm is given in algorithm 1. For each pixel, we construct a 2D Gaussian kernel centered at that pixel with a standard deviation τ

similar to the size of a neuron, and use the kernel to fit the residue from the last iteration. Then pick the location that explains the most variance of the whole movie. To refine the shape of the identified neuron, we extract the small patch of movie centered at the identified location and do a regularized matrix factorization to be discussed in section 4.2.3. Then we subtract the influence of the chosen neuron to update the residue.

Algorithm 1 Greedy neuron identification (GreedyId)

Require: (Centered) data $X \in \mathbb{R}^{N\times T}$; number of neurons K; standard deviation for Gaussian kernel τ; window size w. **Procedure** GreedyId(Y, K, τ, w)

1: $R = Y$;
2: **for** $k = 1 : K$ **do**
3: Calculate variance explained by each Gaussian kernel: $\rho = G^T R$, $v_p = \sum_{t=1}^{T} \rho_{pt}^2$, where $G = G(\tau, w) \in \mathbb{R}^{N\times N}$ is the Gaussian blur matrix. i.e. $G_{:p}$ is a 2D Gaussian kernel centered at pixel p with standard deviation τ and window size w.
4: Identify neuron center: $p_k = \arg\max_p v_p$
5: Initialize $A_k = G_{:p_k}$, $D_k = \rho_{p_k:}$
6: Fine tune the shape of the identified neuron to get a refined D_k and A_k (see section 4.2.3)
7: $R \leftarrow R - A_k C_k$;
8: **end for**
9: **return** $A = [A_1, ..., A_K]$, $D = [D_1^T, ..., D_K^T]^T$, $P = \{p_k\}_{k=1}^K$.

The greedy algorithm we use is reminiscent of the forward-selection procedure in linear regression. Each time the basis that explains the most variance is identified. The reason we use a Gaussian kernel here is that empirically the shape of a neuron resembles a Gaussian kernel. However, other filters such as a constant two-dimensional square (which makes ρ_{pt} a local average in a certain small region) can also be applied.

By using fast Fourier transformation (FFT), the computation of ρ scales gracefully with the dimension of the image. Also notice that after the first iteration, the update of ρ can be done locally since only a local part of the residual is updated.

Instead of using a fixed Gaussian kernel, more elaborate searching scheme can be used to adapt to the variable shape of the neurons. One possible improvements would be to follow Pachitariu *et al.* [2013], which assumes that the shape of a neuron can be written as a linear combination of several basis. Here we fine-tune the neuron shape in each iteration to adapt to the variable shapes of the neurons. We discuss our method in 4.2.3.

4.2.3 Shape fine-tuning

Since the shape of the neuron is similar to but not exactly the same as a Gaussian kernel, after identifying the neuron center at each iteration, we want to refine the neuron shape. Here we propose a regularized matrix factorization technique to decompose the small movie patch using a one-dimensional matrix factorization. Denote S_k as the set of all the pixels that lie in the small square centered at pixel p_k with width = height = w, then at iteration k, we propose the optimization problem

$$\begin{aligned} \underset{D_k, A_k}{\text{minimize}} \quad & \|R - D_k A_k^T\|^2 + \sum_{i=1}^{3} \lambda_i f_i(D_k), \\ \text{subject to} \quad & D_{kp} \geq 0, p \in S_k, \\ & D_{kp} = 0, p \notin S_k, \\ & \|A_k\|_2 \leq c_k, \end{aligned} \tag{4.2}$$

where we propose the following three penalties to regularize D_k:

$$f_1(D_k) = \sum_p \tau_{(p,p_k)} |D_{kp}|,$$

$$f_2(D_k) = \sum_p (D_{kp} - G_{p_k})^2,$$

$$f_3(D_k) = \sum_{p_1 \text{ and } p_2 \text{ are neighbors}} (D_{kp_1} - D_{kp_2})^2.$$

Here f_1 is a Lasso penalty that serves to de-noise the margin of the neuron to enforce sparsity. Note that we define τ_p so that different pixels get penalized differently. In application, we can set τ_{p,p_k} to be 0 when pixel p is close to the identified center p_k, and set a large τ_{p,p_k} to be larger when p is far from p_k. f_2 encourages the neuron shape to be similar to a Gaussian kernel, which is not necessarily desirable when neurons have variable shapes. f_3 is the fused ridge term that encourages the neuron shape to be smooth, which is generally desirable.

The constraint $\|A_k\|_2 \leq c_k$ is necessary since if we use unconstrained A_k, the optimal solution will send D_k to 0 and A_k to infinity. Here we set c_k to be the initial norm of A_k (see Algorithm 1 for how A_k is initialized). This allows regularization to be applied more or less uniformly across neurons. If we set c_k to be the same across neuron k, then the shape of the neurons with large signals will be penalized more, which is unreasonable. In fact, we may even want to regularize neurons with large signals less since we have more information, but we do not discuss the choices here.

Problem (4.2) can be (approximately) solved efficiently. Since we enforce the nonzero elements of D_k to lie in S_k, when doing matrix factorization we only need to use the data restricted at region S_k. We use block-coordinate descent to solve (4.2) with D_k restricted in S_k. Specifically, we alternate between optimizing D_k and optimizing A_k. When $\lambda_3 = 0$, each block-coordinate descent step can be decomposed

into one-dimensional optimization subproblems that can be analytically solved. For $\lambda_3 > 0$, solving D_k for fixed A_k is trickier due to the interaction brought by the penalty. In this case we do several coordinate descent iterations to each element of D_k to approximate the optimization step.

Since the problem is bi-convex but not jointly convex, the algorithm may converge into local-minimum. However, since we have a reasonable initializer (we use Gaussian kernel as initializer for D_k), empirically we found that the algorithm gives reasonable result. In fact, when the data is noisy and the regularization parameter is small, we observed that doing too many coordinate descent adds noise to the learnt neuron component. And we recommend just a finite small number of coordinate descent (3 for example), for both the computational speed and the quality of the neuron shape.

We note that while the penalties make intuitive sense, we need to be careful about setting the regularization parameters λ_i. When we regularize too much, the identified shape will shrink towards 0, leaving extra signals in the residue and causing one neuron to be identified multiple times. When we have a large enough time domain, we may want to use only a small amount of regularization.

4.2.4 Other details

Background elimination: In real data we usually have constant background calcium activity. When the background is present, we pre-process the data by subtracting each pixel with its temporal median. The rationale is that the neuron spikes rarely and therefore most of the time the signal we observe is purely background with random noise, and therefore taking median gives a robust estimate of the background. In the more complicated cases we can have time-varying background activity, which confounds with the single neuron activity. More advanced method for background

elimination has been proposed after this work was done [Pnevmatikakis *et al.*, 2016; Zhou *et al.*, 2016].

Stopping rule: To complete our algorithm, we need a rule to decide when to stop. One possible way would be to first manually estimate the number of neurons K, and stop at the Kth iteration. Another way is to pre-specify a threshold for the minimum explained variance $v_{\min}$ and stop when $\max_p v_p < v_{\min}$. We observe empirically that the latter stopping rule is robust and useful for parallel processing of calcium imaging data.

Handling 3D imaging data: We also have 3-D movie dataset, where calcium activity at different sections of the brain are recorded simultaneously (or to be precise, with very short time lag), therefore giving a 4-D data. Our algorithm can be easily adapted to this case by using a 3D Gaussian kernel or analyzing each z-slice separately.

Recovering spike train from calcium observation: After identifying the neuron shape, we can proceed to get the spike train of each neuron using existing deconvolution methods [Vogelstein *et al.*, 2009; Friedrich and Paninski, 2016].

4.3 Experiments

4.3.1 Simulation examples

We first tested our algorithm on simulated data. We tried two simulations, in both settings, $K = 20$ locations are randomly chosen from the $n_x \times n_y = 200 \times 200$ images to be the center of neurons, and $T = 200$ time bins are generated. For each neuron, the spike train are randomly generated using a Poisson process and the intensity of each neuron is also varying. White noise is added to the signal such that the mean image of the 200 frames are blurred. The only difference in the two simulations is

that in the first simulation, we set the shape of neurons to be Gaussian kernel with standard deviation $\tau = 5$ while in the second simulation we generate the shape of neurons to be the difference of two (scaled) Gaussian densities with different variances $(\tau_1 = 5, \tau_2 = 3.5)$ at the same center, which gives a ring shape.

In both simulations, we use greedy algorithm to infer the neuron center. The number of neurons are set to be the true value and the Gaussian kernel used for greedy search is set to have standard deviation $\tau = 5$ and window size $w = 35$. For fine-tuning the shape of neurons, we only enforce positivity and drop all the other regularization $(\lambda_1 = \lambda_2 = \lambda_3 = 0)$.

Figure 4.3.1 shows the simulation result for both the Gaussian kernel shaped data (top row) and ring shaped data (bottom row). Left column compares the true neuron location (blue '$*$') with the inferred location (numbers indicating the order by which the neuron is added), plotted on the mean image of the simulated data. Right column compares the true neuron shape (top left sub-figure) with the inferred neuron shape (other sub-figures). We see that our algorithm is quite robust to neuron shapes and is able to identify basically all the neuron locations very well. It also recovers the neuron shape reasonably well. When neurons are fairly close (neuron 3, 8, 14), the inferred shape gets distorted, showing the deficiency of the algorithm. Yet we note that this can be solved by doing a global fine-tuning after the initialization, which is elaborated in Pnevmatikakis *et al.* [2016].

4.3.2 Real data analysis

We apply our algorithm on a small patch of the 2-D movie from the brain of a zebrafish [Ahrens and Keller, 2013]. The movie has 100 time bins and each image has size 400-by-150. We apply two variants of our algorithm with different regularization

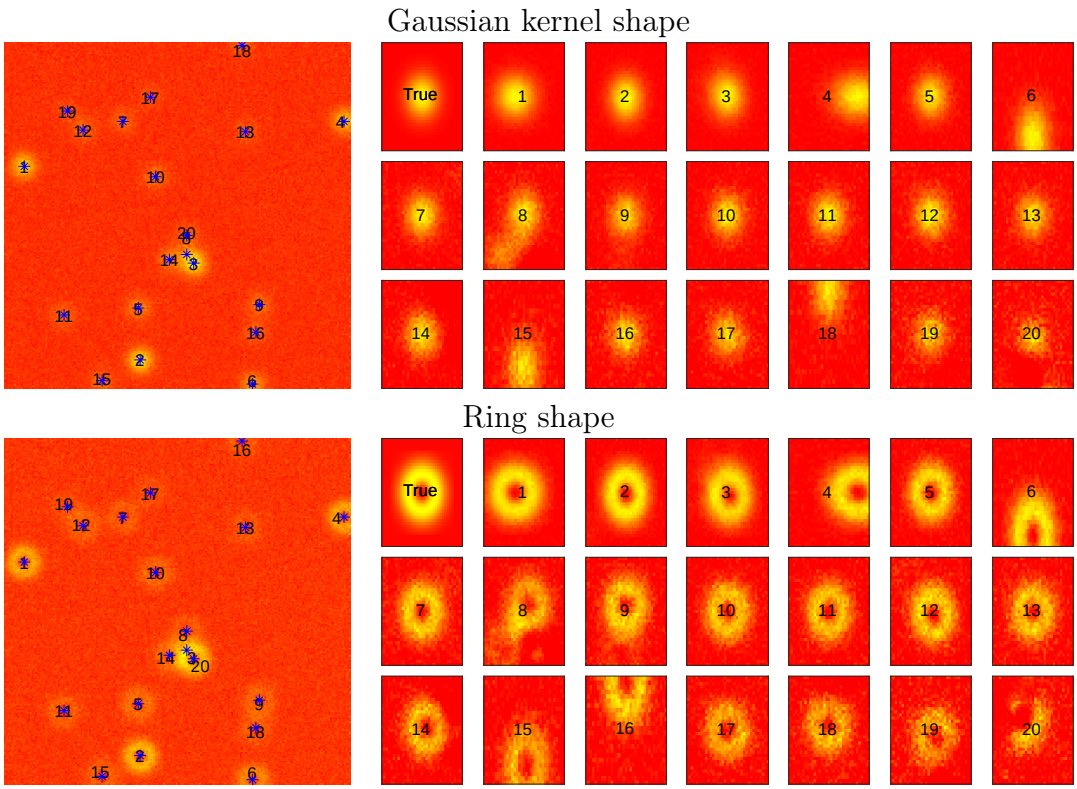

Figure 4.1: Simulated calcium data.

to the data. For the "plain algorithm", we simply enforce the neuron shape to be non-negative. For the "smoothing algorithm", we add penalty to enforce smoothness and localization to the neuron shape ($\lambda_1 > 0, \lambda_3 > 0$). For both algorithms, we manually set the number of neurons to be 30. For both algorithms we preprocess the data by subtracting the median image (for each pixel taking the median across image).

Figure 4.2 shows the result for "plain algorithm" (top row) and "smoothing algorithm" (bottom row). Left column plots the true neuron location inferred location (numbers indicating the order by which the neuron is added) on the mean image of the data. Right column plots the inferred neuron shape. The two algorithms infer similar neuron locations while the "smoothing algorithm" provides more reasonable neuron shapes.

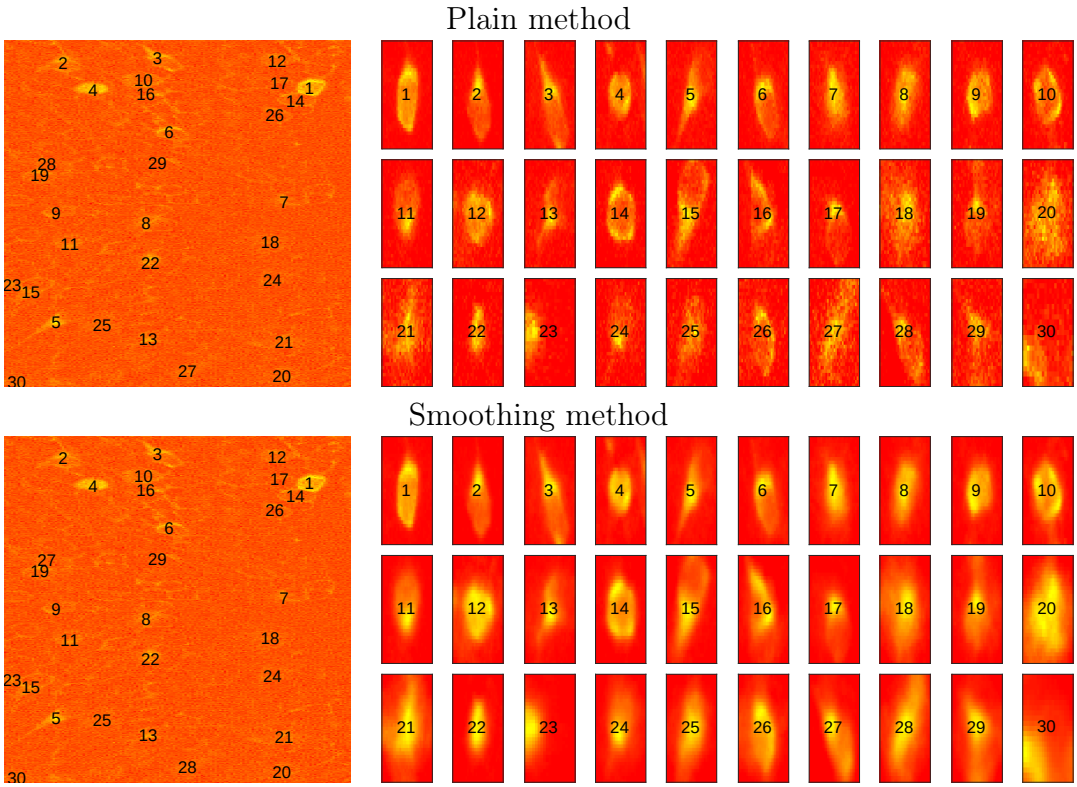

Figure 4.2: Real calcium data.

We then tested our method on a big dataset of [Ahrens and Keller, 2013], which contains the whole-brain data recordings for 1000 time frames. The image is of the dimension $1472 \times 2048 \times 28$. To facilitate computation, we split data into $400 \times 400 \times 1$ patches, with a stride of $(200, 200, 1)$. Here we use the "plain method" since we have a fairly long time domain.

We perform ROI detection on each patch separately and then merge them. We eliminate redundant neurons for overlapping patches when the identified neurons on overlapping patches are spatially close and share similar traces. We plot the inferred neuron locations in image 4.3. While more analysis should be done to measure the quality of the detected neurons, the identified neurons agree well with the shape of the brain of the zebra fish.

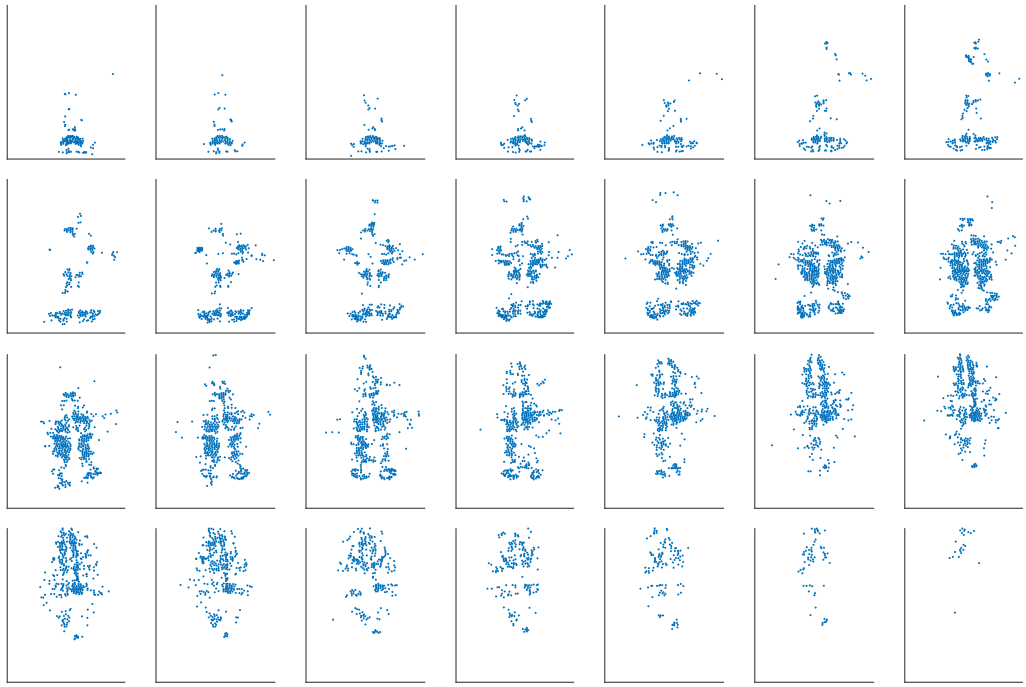

Figure 4.3: ROI detection for the full Misha data, each sub-figure represents a z-slice.

4.4 Discussion

In this chapter we developed a new algorithm for automatic neuron identification in calcium imaging data. By using a variant of matching pursuit for temporal-spatial data, the algorithm encourages the neuron shape to be localized and have biologically plausible shape. By doing matrix factorization locally, we further exploits the spatial-temporal characteristic of neurons. The temporal structure of the data (smoothness and calcium convolution) can be further exploited when the data has high temporal resolution. The ability of the algorithm to handle densely packed regions can be limited due to the greedy fashion of the method. For animals with unconstrained behavior, more advanced motion correction or object tracking methods should be applied as a first step.

After this work was done, several other works have been proposed for calcium imaging processing. Pnevmatikakis *et al.* [2016] used our algorithm as an initialization method and develops methods to refine the neuron shape, add and drop neurons, and also recover spikes from calcium traces. Zhou *et al.* [2016] proposed methods for eliminating noisy background by identifying background with locally low-rank approximation and proposed a new greedy initialization algorithm. Pnevmatikakis and Giovannucci [2017] proposed an online algorithm for calcium imaging processing that also involves a non-rigid motion correction of the data.

Part III

Maximum Entropy Flow Networks

Chapter 5

Maximum Entropy Flow Network

This chapter discusses a method that is related to and motivated by formulating scientific hypothesis testing in neural population data analysis, but can be used in a much broader domain.

With the abundance of large scale neural population recordings, systematic neuroscientists have proposed many population-level analysis techniques and many hypotheses about the neural population structure. One important question to ask is whether those newly identified population structures are meaningful or epiphenomenal? Are those population structures just a consequence of the simpler structures that are already already known, such as the tuning of single neurons or the temporal correlation? This question causes debate for the significance of population structures and a hypothesis testing framework for epiphenomena is important for resolving the debate.

A natural idea for this hypothesis testing framework is to generate a random fake data that follow a null distribution which share the known, simple neural properties with the true observation, and see whether the newly found population structures also appear in fake data. One challenge here is to generate fake data that share the

simpler, known structures and are yet random enough. Maximum entropy modeling is a flexible and popular framework for formulating statistical models given partial knowledge [Jaynes, 1957]. The distribution with the maximum entropy that satisfies certain constraints is regarded as the least biased distribution among all given the constraints. Therefore maximum entropy distributions are often used, in neuroscience or other areas, to build statistical models or for fomulating a null distribution for hypothesis testing [Good, 1963; Schneidman *et al.*, 2006; Tang *et al.*, 2008].

This chapter discusses a way of sampling (approximately) from a continuous maximum entropy distribution given complicated constraint, a challenging computational problem. Rather than the traditional method of optimizing over the continuous density directly, we learn a smooth and invertible transformation that maps a simple distribution to the desired maximum entropy distribution. Doing so is nontrivial in that the objective being maximized (entropy) is a function of the density itself. By exploiting recent developments in normalizing flow networks, we cast the maximum entropy problem into a finite-dimensional constrained optimization, and solve the problem by combining stochastic optimization with the augmented Lagrangian method. Applications to finance and computer vision show the flexibility and accuracy of using maximum entropy flow networks.

This work, which was published as Loaiza-Ganem *et al.* [2017], was done with Gabriel Loaiza-Ganem and John Cunningham. I was heavily involved with every aspect of the paper.

5.1 Introduction

The maximum entropy (ME) principle [Jaynes, 1957] states that subject to some given prior knowledge, typically some given list of moment constraints, the distribution that

makes minimal additional assumptions – and is therefore appropriate for a range of applications from hypothesis testing to price forecasting to texture synthesis – is that which has the largest entropy of any distribution obeying those constraints. First introduced in statistical mechanics by Jaynes [1957], and considered both celebrated and controversial, ME has been extensively applied in areas including natural language processing [Berger *et al.*, 1996], ecology [Phillips *et al.*, 2006], finance [Buchen and Kelly, 1996], computer vision [Zhu *et al.*, 1998], and many more.

Continuous ME modeling problems typically include certain expectation constraints, and are usually solved by introducing Lagrange multipliers, which under typical assumptions yields an exponential family distribution (also called Gibbs distribution) with natural parameters such that the expectation constraints are obeyed. Unfortunately, fitting ME distributions in even modest dimensions poses significant challenges. First, optimizing the Lagrangian for a Gibbs distribution requires evaluating the normalizing constant, which is in general computationally very costly and error prone. Secondly, in all but the rarest cases, there is no way to draw samples independently and identically from this Gibbs distribution, even if one could derive it. Third, unlike in the discrete case where a number of recent and exciting works have addressed the problem of estimating entropy from discrete-valued data [Jiao *et al.*, 2015; Valiant and Valiant, 2013], estimating differential entropy from data samples remains inefficient and typically biased. These shortcomings are critical and costly, given the common use of ME distributions for generating reference data samples for a null distribution of a test statistic. There is thus ample need for a method that can both solve the ME problem and produce a solution that is easy and fast to sample.

In this paper we develop maximum entropy flow networks (MEFN), a stochastic-optimization-based framework and algorithm for fitting continuous maximum entropy models. Two key steps are required. First, conceptually, we replace the idea of max-

imizing entropy over a density directly with maximizing, over the parameter space of an indexed function family, the entropy of the density induced by mapping a simple distribution (a Gaussian) through that optimized function. Modern neural networks, particularly in variational inference [Kingma and Welling, 2013; Rezende and Mohamed, 2015], have successfully employed this same idea to generate complex distributions, and we look to similar technologies. Secondly, unlike most other objectives in this network literature, the entropy objective itself requires evaluation of the target density directly, which is unavailable in most traditional architectures. We overcome this potential issue by learning a smooth, invertible transformation that maps a simple distribution to an (approximate) ME distribution. Recent developments in normalizing flows [Rezende and Mohamed, 2015; Dinh *et al.*, 2016] allow us to avoid biased and computationally inefficient estimators of differential entropy (such as the nearest-neighbor class of estimators like that of Kozachenko-Leonenko; see Berrett *et al.* [2016]). Our approach avoids calculation of normalizing constants by learning a map with an easy-to-compute Jacobian, yielding tractable probability density computation. The resulting transformation also allows us to reliably generate iid samples from the learned ME distribution. We demonstrate MEFN in detail in examples where we can access ground truth, and then we demonstrate further the ability of MEFN networks in equity option prices fitting and texture synthesis.

Primary contributions of this work include: *(i)* addressing the substantial need for methods to sample ME distributions; *(ii)* introducing ME problems, and the value of including entropy in a range of generative modeling problems, to the deep learning community; *(iii)* the novel use of *constrained* optimization for a deep learning application; and *(iv)* the application of MEFN to option pricing and texture synthesis, where in the latter we show significant increase in the diversity of synthesized textures (over current state of the art) by using MEFN.

5.2 Background

5.2.1 Maximum entropy modeling and Gibbs distribution

We consider a continuous random variable $\mathbf{Z} \in \mathcal{Z} \subseteq \mathbb{R}^d$ with density p, where p has differential entropy $H(p) = -\int p(\mathbf{z}) \log p(\mathbf{z}) d\mathbf{z}$ and support $\text{supp}(p)$. The goal of ME modeling is to find, and then be able to easily sample from, the maximum entropy distribution given a set of moment and support constraints, namely the solution to:

$$\begin{aligned} p^* \quad = \quad & \text{maximize} \quad H(p) \\ & \text{subject to} \quad E_{\mathbf{Z}\sim p}[T(\mathbf{Z})] = 0 \\ & \qquad\qquad\quad \text{supp}(p) = \mathcal{Z}, \end{aligned} \tag{5.1}$$

where $T(\mathbf{z}) = (T_1(\mathbf{z}), ..., T_m(\mathbf{z})) : \mathcal{Z} \to \mathbb{R}^m$ is the vector of known (assumed sufficient) statistics, and $\mathcal{Z}$ is the given support of the distribution. Under standard regularity conditions, the optimization problem can be solved by Lagrange multipliers, yielding an exponential family p^* of the form:

$$p^*(\mathbf{z}) \propto e^{\eta^\top T(\mathbf{z})} \mathbb{1}(\mathbf{z} \in \mathcal{Z}) \tag{5.2}$$

where $\eta \in \mathbb{R}^m$ is the choice of natural parameters of p^* such that $E_{p^*}[T(\mathbf{Z})] = 0$. Despite this simple form, these distributions are only in rare cases tractable from the standpoint of calculating η, calculating the normalizing constant of p^*, and sampling from the resulting distribution. There is extensive literature on finding η numerically [Darroch and Ratcliff, 1972; Salakhutdinov *et al.*, 2002; Della Pietra *et al.*, 1997; Dudik *et al.*, 2004; Malouf, 2002; Collins *et al.*, 2002], but doing so requires computing normalizing constants, which poses a challenge even for problems with modest

dimensions. Also, even if η is correctly found, it is still not trivial to sample from p^*. Problem-specific sampling methods (such as importance sampling, MCMC, etc.) have to be designed and used, which is in general challenging (burn-in, mixing time, etc.) and computationally burdensome.

5.2.2 Normalizing flows

Following Rezende and Mohamed [2015], we define a *normalizing flow* as the transformation of a probability density through a sequence of invertible mappings. Normalizing flows provide an elegant way of generating a complicated distribution while maintaining tractable density evaluation. Starting with a simple distribution $\mathbf{Z}_0 \in \mathbb{R}^d \sim p_0$ (usually taken to be a standard multivariate Gaussian), and by applying k invertible and smooth functions $f_i : \mathbb{R}^d \to \mathbb{R}^d (i = 1, ..., k)$, the resulting variable $\mathbf{Z}_k = f_k \circ f_{k-1} \circ \cdots \circ f_1(\mathbf{Z}_0)$ has density:

$$p_k(\mathbf{z}_k) = p_0(f_1^{-1} \circ f_2^{-1} \circ \cdots \circ f_k^{-1}(\mathbf{z}_k)) \prod_{i=1}^{k} |\det(J_i(\mathbf{z}_{i-1}))|^{-1}, \tag{5.3}$$

where J_i is the Jacobian of f_i. If the determinant of J_i can be easily computed, p_k can be computed efficiently.

Rezende and Mohamed [2015] proposed two specific families of transformations for variational inference, namely planar flows and radial flows, respectively:

$$f_i(\mathbf{z}) = \mathbf{z} + \mathbf{u}_i h(\mathbf{w}_i^T \mathbf{z} + b_i) \qquad \text{and} \qquad f_i(\mathbf{z}) = \mathbf{z} + \beta_i h(\alpha_i, r_i)(\mathbf{z} - \mathbf{z}_i'), \tag{5.4}$$

where $b_i \in \mathbb{R}$, $\mathbf{u}_i, \mathbf{w}_i \in \mathbb{R}^d$ and h is an activation function in the planar case, and $\beta_i \in \mathbb{R}$, $\alpha_i > 0$, $\mathbf{z}_i' \in \mathbb{R}^d$, $h(\alpha, r) = 1/(\alpha + r)$ and $r_i = ||\mathbf{z} - \mathbf{z}_i'||$ in the radial. Recently Dinh *et al.* [2016] proposed a normalizing flow with convolutional, multiscale structure

that is suitable for image modeling and has shown promise in density estimation for natural images.

5.3 Maximum entropy flow network (MEFN) algorithm

5.3.1 Formulation

Instead of solving Equation 5.2, we propose solving Equation 5.1 directly by optimizing a transformation that maps a random variable $\mathbf{Z}_0$, with simple distribution p_0, to the ME distribution. Given a parametric family of normalizing flows $\mathcal{F} = \{f_\phi, \phi \in \mathbb{R}^q\}$, we denote $p_\phi(\mathbf{z}) = p_0(f_\phi^{-1}(\mathbf{z}))|\det(J_\phi(\mathbf{z}))|^{-1}$ as the distribution of the variable $f_\phi(\mathbf{Z}_0)$, where J_ϕ is the Jacobian of f_ϕ. We then rewrite the ME problem as:

$$\begin{aligned} \phi^* \quad = \quad & \text{maximize} \quad H(p_\phi) \\ & \text{subject to} \quad E_{\mathbf{Z}_0 \sim p_0}[T(f_\phi(\mathbf{Z}_0))] = 0 \\ & \qquad\qquad \text{supp}(p_\phi) = \mathcal{Z}. \end{aligned} \tag{5.5}$$

When p_0 is continuous and $\mathcal{F}$ is suitably general, the program in Equation 5.5 recovers the ME distribution p_ϕ exactly. With a flexible transformation family, the ME distribution can be well approximated. In experiments we found that taking p_0 to be a standard multivariate normal distribution achieves good empirical performance. Taking p_0 to be a bounded distribution (e.g. uniform distribution) is problematic for learning transformations near the boundary, and heavy tailed distributions (e.g. Cauchy distribution) caused similar trouble due to large numbers of outliers.

5.3.2 Algorithm

We solved Equation 5.5 using the augmented Lagrangian method. Denote $R(\phi) = E(T(f_\phi(\mathbf{Z}_0)))$, the augmented Lagrangian method uses the following objective:

$$L(\phi; \lambda, c) = -H(p_\phi) + \lambda^\top R(\phi) + \frac{c}{2}||R(\phi)||^2 \tag{5.6}$$

where $\lambda \in \mathbb{R}^m$ is the Lagrange multiplier and $c > 0$ is the penalty coefficient. We minimize Equation 5.6 for a non-decreasing sequence of c and well-chosen λ. As a technical note, the augmented Lagrangian method is guaranteed to converge under some regularity conditions [Bertsekas, 2014]. As is usual in neural networks, a proof of these conditions is challenging and not yet available, though intuitive arguments suggest that most of them should hold. We omit a more thorough discussion about them and rely instead on the empirical results of the algorithm to claim that it is indeed solving the optimization problem.

For a fixed (λ, c) pair, we optimize L with stochastic gradient descent. Owing to our choice of network and the resulting ability to efficiently calculate the density $p_\phi(\mathbf{z}^{(i)})$ for any sample point $\mathbf{z}^{(i)}$ (which are easy-to-sample iid draws from the multivariate normal p_0), we compute the unbiased estimator of $H(p_\phi)$ with:

$$H(p_\phi) \approx -\frac{1}{n}\sum_{i=1}^{n} \log p_\phi(f_\phi(\mathbf{z}^{(i)})) \tag{5.7}$$

$R(\phi)$ can also be estimated without bias by taking a sample average of $\mathbf{z}^{(i)}$ draws. The resulting optimization procedure is detailed in Algorithm 2, of which step 9 requires some detail: denoting ϕ_k as the resulting ϕ after i_{max} SGD iterations at the

Algorithm 2 Training the MEFN

1: initialize $\phi = \phi_0$, set $c_0 > 0$ and λ_0.
2: **for** Augmented Lagrangian iteration $k = 1, ..., k_{\max}$ **do**
3: **for** SGD iteration $i = 1, ..., i_{\max}$ **do**
4: Sample $\mathbf{z}^{(1)}, ..., \mathbf{z}^{(n)} \sim p_0$, get transformed variables $\mathbf{z}_\phi^{(i)} = f_\phi(\mathbf{z}^{(i)}), i = 1, ..., n$
5: Update ϕ by descending its stochastic gradient (using e.g. ADADELTA [Zeiler, 2012]):

$$\nabla_\phi L(\phi; \lambda_k, c_k) \approx \frac{1}{n}\sum_{i=1}^{n} \nabla_\phi \log p_\phi(\mathbf{z}_\phi^{(i)}) + \lambda_k \cdot \frac{1}{n}\sum_{i=1}^{n} \nabla_\phi T(\mathbf{z}_\phi^{(i)}) + c_k \frac{2}{n}\sum_{i=1}^{\frac{n}{2}} \nabla_\phi T(\mathbf{z}_\phi^{(i)}) \cdot \frac{2}{n}\sum_{i=\frac{n}{2}+1}^{n} T(\mathbf{z}_\phi^{(i)})$$

6: **end for**
7: Sample $\mathbf{z}^{(1)}, ..., \mathbf{z}^{(\tilde{n})} \sim p_0$, get transformed variables $\mathbf{z}_\phi^{(i)} = f_\phi(\mathbf{z}^{(i)}), i = 1, ..., \tilde{n}$
8: Update $\lambda_{k+1} = \lambda_k + c_k \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} T(\mathbf{z}_\phi^{(i)})$
9: Update $c_{k+1} \geq c_k$ (see text for detail)
10: **end for**

augmented Lagrangian iteration k, the usual update rule for c [Bertsekas, 2014] is:

$$c_{k+1} = \begin{cases} \beta c_k, \text{ if } ||R(\phi_k)|| > \gamma ||R(\phi_{k-1})|| \\ c_k, \text{ otherwise} \end{cases} \tag{5.8}$$

where $\gamma \in (0, 1)$ and $\beta > 1$. What results is a robust and novel algorithm for estimating maximum entropy distributions, while preserving the critical properties of being both easy to calculate densities of particular points, and being trivially able to produce truly iid samples.

5.4 Experiments

§5.4.2 and §5.4.3 applies the MEFN to a financial data application (predicting equity option values) and texture synthesis, respectively, to illustrate the flexibility and practicality of our algorithm. For §5.4.2, We use 10 layers of planar flow with a final transformation g (specified below) that transforms samples to the specified support, and use with ADADELTA [Zeiler, 2012]. For §5.4.3 we use real NVP structure and use ADAM [Kingma and Ba, 2014] with learning rate $= 0.001$. For all our experiments, we use $i_{max} = 3000$, $\beta = 4$, $\gamma = 0.25$. For §5.4.2 we use $n = 300$, $\tilde{n} = 1000$, $k_{\max} = 10$; For §5.4.3 we use $n = \tilde{n} = 2$, $k_{\max} = 8$.

5.4.1 A maximum entropy problem with known solution

Following the setup of the typical ME problem, suppose we are given a specified support $\mathcal{S} = \{\mathbf{z} = (z_1, \ldots, z_{d-1}) : z_i \geq 0 \text{ and } \sum_{k=1}^{d-1} z_k \leq 1\}$ and a set of constraints $E[\log Z_k] = \kappa_k (k = 1, ..., d)$, where $Z_d = 1 - \sum_{k=1}^{d-1} Z_k$. We then write the maximum entropy program:

$$\begin{aligned} p^* \quad = \quad & \text{maximize} \quad H(p) \\ & \text{subject to} \quad E_{\mathbf{Z}\sim p}[\log Z_k - \kappa_k] = 0 \quad \forall k = 1, ..., d \\ & \qquad\qquad \text{supp}(p) = \mathcal{S}. \end{aligned} \tag{5.9}$$

This is a general ME problem that can be solved via the MEFN. Of course, we have particularly chosen this example because, though it may not obviously appear so, the solution has a standard and tractable form, namely the Dirichlet. This choice allows us to consider a complicated optimization program that happens to have known global optimum, providing a solid test bed for the MEFN (and for the Gibbs approach

against which we will compare). Specifically, given a parameter $\alpha \in \mathbb{R}^d$, the Dirichlet has density:

$$p(z_1, \ldots, z_{d-1}) = \frac{1}{B(\alpha)} \prod_{k=1}^{d} z_k^{\alpha_k - 1} \mathbb{1}\left((z_1, \ldots, z_{d-1}) \in \mathcal{S}\right) \tag{5.10}$$

where $B(\alpha)$ is the multivariate Beta function, and $z_d = 1 - \sum_{k=1}^{d-1} z_k$. Note that this Dirichlet is a distribution on $\mathcal{S}$ and not on the $(d-1)$-dimensional simplex $\mathcal{S}^{d-1} = \{(z_1, \ldots, z_d) : z_k \geq 0 \text{ and } \sum_{k=1}^{d} z_k = 1\}$ (an often ignored and seemingly unimportant technicality that needs to be correct here to ensure the proper transformation of measure). Connecting this familiar distribution to the ME problem above, we simply have to choose α such that $\kappa_k = \psi(\alpha_k) - \psi(\alpha_0)$ for $k = 1, ..., d$, where $\alpha_0 = \sum_{k=1}^{d} \alpha_k$ and ψ is the digamma function. We then can pose the above ME problem to the MEFN and to the competitive Gibbs method, and compare performance against ground truth. Before doing so, we must stipulate the transformation g that maps the Euclidean space of the multivariate normal p_0 to the desired support $\mathcal{S}$. Any sensible choice will work well (another point of flexibility for the MEFN); we use the standard transformation $g(z_1, ..., z_{d-1}) = \left(e^{z_1}/(\sum_{k=1}^{d-1} e^{z_k} + 1), ..., e^{z_{d-1}}/(\sum_{k=1}^{d-1} e^{z_k} + 1)\right)$. Note that the MEFN outputs vectors in $\mathbb{R}^{d-1}$, and not $\mathbb{R}^d$, because the Dirichlet is specified as a distribution on $\mathcal{S}$ (and not on the simplex $\mathcal{S}^{d-1}$). Accordingly, the Jacobian is a square matrix and its determinant can be computed efficiently using the matrix determinant lemma. Here, p_0 is set to the $(d-1)$-dimensional standard normal.

We proceed as follows: We choose $\alpha = (1, 2, 3)$ and compute the constraints $\kappa_1, ..., \kappa_d$. We run MEFN pretending we do not know α or the Dirichlet form. Figure 5.4.1 shows an example of the transformation from normal (left panel) to MEFN (middle panel), and comparing that to the ground truth Dirichlet (right panel). The MEFN and ground truth Dirichlet densities shown in purple match closely, and the

samples drawn (red) indeed appear to be iid draws from the same (maximum entropy) distribution in both cases.

Additionally, the middle panel of Figure 5.4.1 shows an important cautionary tale that foreshadows our texture synthesis results (§5.4.3). One might suppose that satisfying the moment matching constraints is adequate to produce a distribution which, if not technically the ME distribution, is still interestingly variable. The middle panel shows the failure of this intuition: in dark green, we show a network trained to simply match the moments specified above, and the resulting distribution quite poorly expresses the variability available to a distribution with these constraints, leading to samples that are needlessly similar. Given the substantial interest in using networks to learn implicit generative models (e.g., Mohamed and Lakshminarayanan [2016]), this concern is particularly relevant and highlights the importance of considering entropy.

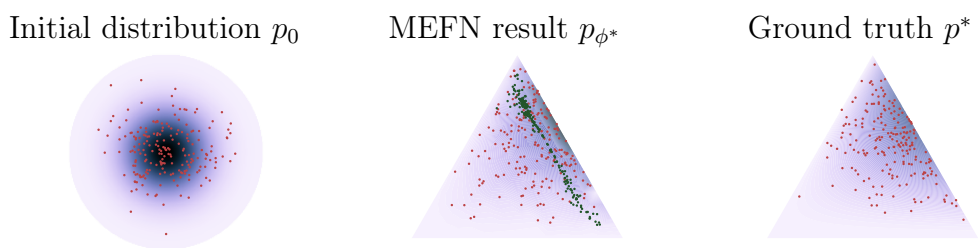

Figure 5.1: Example results from the ME problem with known Dirichlet ground truth. *Left panel*: The normal density p_0 (purple) and iid samples from p_0 (red points). *Middle panel*: The MEFN transforms p_0 to the desired maximum entropy distribution p_{ϕ^*} on the simplex (calculated density p_{ϕ^*} in purple). Truly iid samples are easily drawn from p_{ϕ^*} (red points) by drawing from p_0 and mapping those points through f_{ϕ^*}. Shown in the middle panel are the same points in the top left panel mapped through f_{ϕ^*}. Samples corresponding to training the same network as MEFN to simply match the specified moments (ignoring entropy) are also shown (dark green points; see text). *Right panel*: The ground truth (in this example, known to be Dirichlet) distribution in purple, and iid samples from it in red.

We then take a random sample from the fitted distribution and a random sample from the Dirichlet with parameter α, and compare the two samples using the maxi-

mum mean discrepancy (MMD) kernel two sample test [Gretton *et al.*, 2012], which assesses the fit quality. We take the sample size to be 300 for both distributions and our samples pass the MMD test ($p > 0.05$).

5.4.2 Risk-neutral asset pricing

We apply our method for extracting the risk-neutral asset price probability distribution based on option prices, an active and interesting area for ME models. Here we want to get a distribution on the price of an asset in the future time t_e, and the partial information we have is the prices of options expiring at t_e, which are financial contracts whose prices can be expressed as the expectation of the asset price distribution at t_e. Given those expectation constraints, a natural guess of the distribution of the asset price can be formulated as an ME problem. Below we discuss the mathematical formulation of this problem, interested readers to see Buchen and Kelly [1996] for a more detailed explanation.

Denoting S_t as the price of an asset at time t, the buyer of a European call option for the stock that expires at time t_e with strike price K will receive a payoff of $C_K = (S_{t_e} - K)_+ = \max(S_{t_e} - K, 0)$ at time t_e. Under the efficient market assumption, the risk-neutral probability distribution for the stock price at time t_e satisfies:

$$C_K = D(t_e)E_q[(S_{t_e} - K)_+], \tag{5.11}$$

where $D(t_e)$ is the risk-free discount factor and q is the risk-neutral measure. We also have that, under the risk-neutral measure, the current stock price S_0 is the discounted expected value of S_{t_e}:

$$S_0 = D(t_e)E_q(S_{t_e}). \tag{5.12}$$

When given m options that expire at time t_e with strikes $K_1, ..., K_m$ and prices $C_{K_1}, ..., C_{K_m}$, we get m expectation constraints on $q(S_{t_e})$ from Equation 5.11, together with Equation 5.12, we have $m+1$ expectation constraints in total. With that partial knowledge we can approximate $q(S_{t_e})$, which is helpful for understanding the market expected volatility and identify mispricing in option markets, etc.

Inferring the risk-neutral density of asset price from a finite number of option prices is an important question in finance and has been studied extensively [Buchen and Kelly, 1996; Borwein *et al.*, 2003; Bondarenko, 2003]. One popular method proposed by Buchen and Kelly [1996] estimates the probability density as the maximum entropy distribution satisfying the expectation constraints and a positivity support constraint by fitting a Gibbs distribution, which results in a piece-wise linear log density:

$$p(z) \propto \exp\left\{\eta_0 z + \sum_{i=1}^{m} \eta_i (z - K_i)_+\right\} \mathbb{1}\left(z \geq 0\right) \tag{5.13}$$

and optimize the distribution with numerical methods. Here we compare the performance of the MEFN algorithm with the method proposed in Buchen and Kelly [1996]. To enforce the positivity constraint we choose $g(z) = e^{az+b}$, where a and b are additional parameters.

We collect the closing price of European call options on Nov. 1 2016 for the stock AAPL (Apple inc.) that expires on $t_e =$ Jun. 16 2017. We use $m = 4$ of the options with highest trading volume as training data and the rest as testing data. On the left panel of figure 5.2, we show the fitted risk-neutral density of S_{t_e} by MEFN (red line) with that of the fitted Gibbs distribution result (blue line). We find that while the distributions share similar location and variability, the distribution inferred by MEFN is smoother and arguably more plausible. In the middle panel we show a Q-Q plot of the quantiles of the MEFN and Gibbs distributions. We can see that

the quantile pairs match the identity closely, which should happen if both methods recovered the exact same distribution. This highlights the effectiveness of MEFN. There does exist a small mismatch in the tails: the distribution inferred by MEFN has slightly heavier tails. This mismatch is difficult to interpret: given that both the Gibbs and MEFN distributions are fit with option price data (and given that one can observe at most one value from the distribution, namely the stock price at expiration), it is fundamentally unclear which distribution is superior, in the sense of better capturing the true ME distribution's tails. On the right panel we show the fitted option price for the two fitted distributions (for each strike price, we can recover the fitted option price by Equation 5.11). We noted that the fitted option price and strike price lines for both methods are very similar (they are mostly indiscernible on the right panel of figure 5.2). We also compare the fitted performance on the test data by computing the root mean square error for the fitted and test data. We observe that the predictive performances for both methods are comparable.

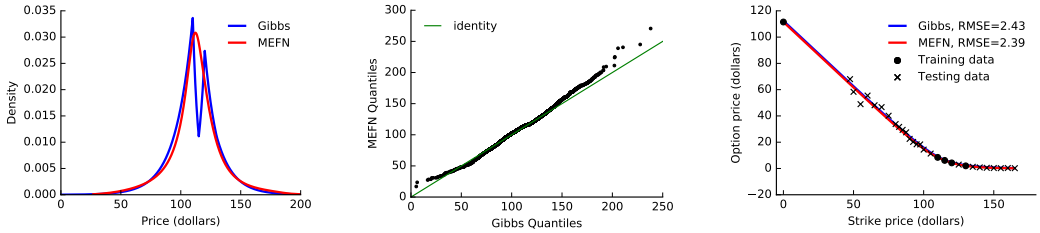

Figure 5.2: Constructing risk-neutral measure from observed option price. *Left panel*: fitted risk-neutral measure by Gibbs and MEFN method. *Middle panel*: Q-Q plot for the quantiles from the distributions on the left panel. *Right panel*: observed and fitted option price for different strikes.

We note that for this specific application, there are practical concerns such as the microstructure noise in the data and inefficiency in the market, etc. Applying a pre-processing procedure and incorporating prior assumptions can be helpful for getting a more full-fledged method. Here we mainly focus on illustrating the ability of the

MEFN method to approximate the ME distribution for non-typical distributions. Future work for this application includes fitting a risk-neutral distribution for multi-dimensional assets by incorporating dependence structure on assets.

5.4.3 Modeling images of textures

Constructing generative models to generate random images with certain texture structure is an important task in computer vision. A line of texture synthesis research proceeds by first extracting a set of features that characterizes the target texture and then generate images that match the features. The seminal work of Zhu *et al.* [1998] proposes constructing texture models under the ME framework, where features (or filters) of the given texture image are adaptively added in the model and a Gibbs distribution whose expected feature matches the target texture is learnt. One major difficulty with the method is that both model learning and image generation involve sampling from a complicated Gibbs distribution. More recent works exploit more complicated features [Portilla and Simoncelli, 2000; Gatys *et al.*, 2015; Ulyanov *et al.*, 2016]. Ulyanov *et al.* [2016] proposes the *texture net*, which uses a texture loss function by using the Gram matrices of the outputs of some convolutional layers of a pre-trained deep neural network for object recognition.

While the use of these complicated features does provide high-quality synthetic texture images, that work focuses exclusively on generating images that match these feature (moments). Importantly, this network focuses only on generating feature-matching images without using the ME framework to promote the diversity of the samples. Doing so can be deeply problematic: in Figure 5.4.1 (middle panel), we showed the lack of diversity resulting from only moment matching in that Dirichlet setting, and further we note that the extreme pathology would result in a point

mass on the training image – a global optimum for this objective, but obviously a terrible generative model for synthesizing textures. Ideally, the MEFN will match the moments *and* promote sample diversity.

We applied MEFN to texture synthesis with an RGB representation of the 224×224 pixel images , $\mathbf{z} \in \mathcal{Z} = [0,1]^d$, where $d = 224 \times 224 \times 3$. We follow Ulyanov *et al.* [2016] (we adapted `https://github.com/ProofByConstruction/texture-networks`) to create a texture loss measure $T(\mathbf{z}) : [0,1]^d \to \mathbb{R}$, and aim to sample a diverse set of images with small moment violation. For the transformation family $\mathcal{F}$ we use the real NVP network structure proposed in Dinh *et al.* [2016] (we adapted `https://github.com/taesung89/real-nvp`). We use 3 residual blocks with 32 feature maps for each coupling layer and downscale 3 times. For fair comparison, we use the same real NVP structure for both methods and implement both methods in TensorFlow [Abadi *et al.*, 2016]. Note that Ulyanov *et al.* [2016] used a quite different generative network structure for texture network, which is not invertible and is therefore infeasible for entropy evaluation. In our experiments we replace their generative network by the real NVP structure, which allows us to get an Monte Carlo estimate of the entropy for both generative models (by computing a sample average of log density) and ensures that the structure of the generative network does not affect the comparison.

As is shown in top row of figure 5.3, both methods generate visually pleasing images capturing the texture structure well. The bottom row of Figure 5.3 shows that texture cost (left panel) is similar for both methods, while MEFN generate figures with much larger entropy than the texture network formulation (middle panel), which is desirable (as previously discussed). The bottom right panel of figure 5.3 compares the marginal distribution of the RGB values sampled from the networks: we found that MEFN generates a more variable distribution of RGB values than the texture

net.

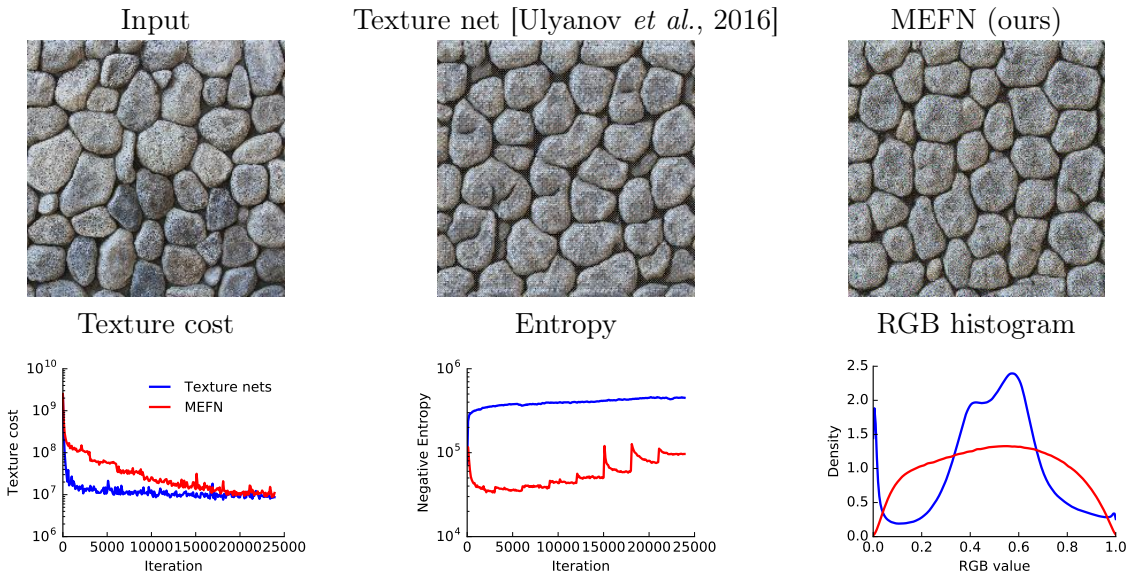

Figure 5.3: Analysis of texture synthesis experiment. See text for description.

We compute in Table 5.1 the average pairwise Euclidean distance between randomly sampled images ($d_{L^2} = \text{mean}_{i \neq j} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2$), and MEFN gives higher d_{L^2}, quantifying diversity across images. We also consider an ANOVA-style analysis to measure the diversity of the images, where we think of the RGB values for the same pixel across multiple images as a group, and compute the within and between group variance. Specifically, denoting $z_k^{(i)}$ as the pixel value for a specific pixel $k = 1, ..., d$ for an image $i = 1, ..., n$. We partition the total sum of square SST $= \sum_{i,k}(z_k^{(i)} - \bar{z})^2$ as the within group error SSW $= \sum_{i,k}(z_k^{(i)} - \bar{z}_k)^2$ and between group error SSB $= \sum_k n(\bar{z}_k - \bar{z})^2$, where $\bar{z}$ and $\bar{z}_k$ are the mean pixel values across all data and for a specific pixel k. Ideally we want the samples to exhibit large variability across images (large SSW, within a group/pixe) and no structure in the mean image (small SSB, across groups/pixels). Indeed, the MEFN has a larger SSW, implying higher variability around the mean image, a smaller SSB, implying the stationarity of the generated samples, and a larger SST, implying larger total variability also. The MEFN produces images that are con-

clusively more variable without sacrificing the quality of the texture, implicating the broad utility of ME.

Table 5.1: Quantitative measure of image diversity using 20 randomly sampled images

Method	d_{L^2}	SST	SSW	SSB
Texture net	11534	128680	109577	19103
MEFN	17014	175604	161639	13964

While the quantitative measures imply more diversity in MEFN result, visual examination for the samples generated by MEFN and texture network implies that both methods exhibit diverse samples. As is shown in Figure 5.4, the samples from both methods exhibit diversity and the mean images do not exhibit strong pattern.

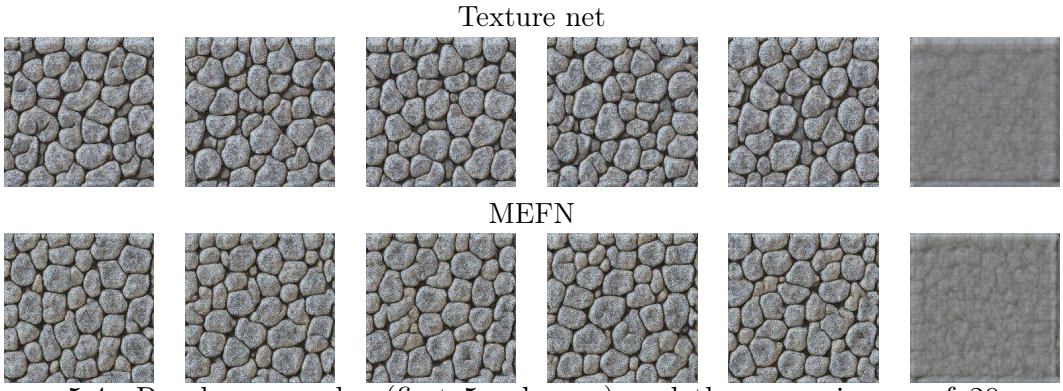

Figure 5.4: Random samples (first 5 columns) and the mean image of 20 random samples (last column) from texture net (upper row) and MEFN (bottom row) for the stone example.

While the texture net method [Ulyanov *et al.*, 2016] does exhibit a certain amount of sample diversity in the stone experiments and a few other cases that we tried, we expect that the performance to deteriorate when the image is more complicated and the generative network structure is more complex. To further understand the behavior, we tried another experiments with a brick texture. Here we used a more complicated generative real-nvp network structure with 8 residual blocks with 64

feature maps for each coupling layer and downscale 4 times. While we again observe higher entropy value for MEFN (not shown), from Figure 5.5 we find that that both texture net and MEFN with a large texture penalty (large initial c value) give non-diverse examples with mean image exhibiting strong patterns (last column of the second and third rows). We then set the initial coefficient for the quadratic penalty c_0 to be smaller and are able to get a much more diverse example (last row). We think that a small c_0 would initialize the network such that it explores the full image space, facilitating generating diverse images.

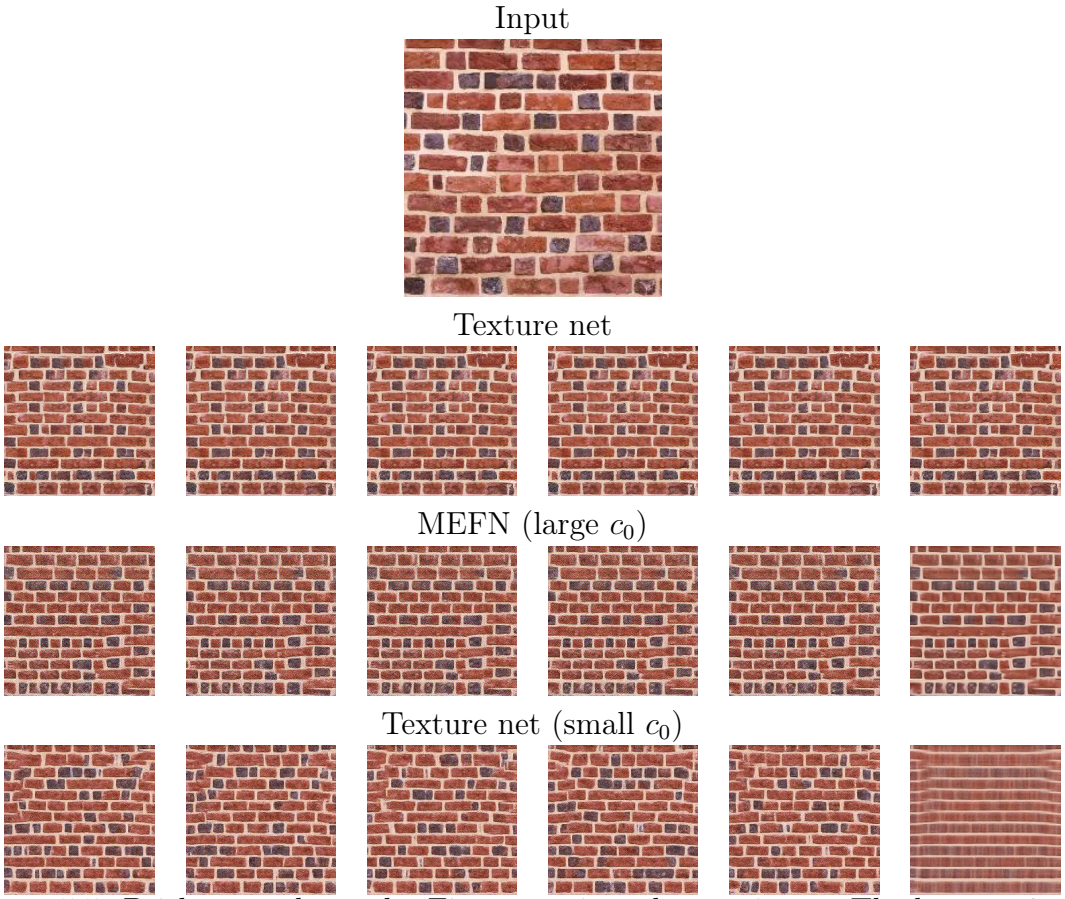

Figure 5.5: Brick example result. First row gives the raw input. The bottom 3 rows give 5 random samples (first 5 columns) and the mean image of 20 random samples (last column) from texture net (row 2) and MEFN with large initial texture cost penalty (row 3) and smaller initial texture cost penalty (bottom row) for the brick example.

We note that here large entropy does not necessarily imply more visual diversity. The image space is a very high-dimensional space and the distribution of images following a certain texture can be expected to lie close to a low-dimensional manifold, making entropy very small (any distribution on a subspace of the image space would have entropy to be negative infinity using our definition for continuous distribution here) and also unstable to estimate. For such complicated distribution we do not expect that our current method is able to sample from the real ME distribution but the hope is that our method can give more desirable method to algorithms that do not explicitly encourage sample diversity. Also our optimization methods may need to be fine tuned to get desirable performance.

5.5 Discussion

In this chapter we propose a general framework for fitting ME models. This approach is novel and has three key features. First, by learning a transformation of a simple distribution rather than the distribution itself, we are able to avoid explicitly computing an intractable normalizing constant for the ME distribution. Second, by combining stochastic optimization with the augmented Lagrangian method, we can fit the model efficiently, allowing us to evaluate the ME density of any point simply and accurately. Third, critically, this construction allows us to trivially sample iid from a ME distribution, extending the utility and efficiency of the ME framework more generally. Thus, accuracy equivalent to the classic Gibbs approach already would in itself be a contribution (owing to these other features).

The structure of the normalizing flow is crucial for the success of the algorithm. Ideally we want the network structure to be expressive while maintaining computational tractability and numerical stability. We would also hope that the normalizing

flow to be general and suitable for a wide range of applications. One possible direction is to consider linear transformation with structured matrix that allows fast matrix multiplication and Jacobian determinant computation (circulant matrix, for example), followed by point-wise nonlinearity.

While we have shown empirical outperformance in generating distribution close to maximum entropy distribution, it is easy to get trapped in local optimum when faced with complicated, multi-modal distribution, as is the case in texture modeling. How to improve our algorithm to get better distribution is an open question.

There are a few recent works encouraging sample diversity in the setting of texture modeling. Ulyanov *et al.* [2017] extended Ulyanov *et al.* [2016] by adding a penalty term using the Kozachenko-Leonenko estimator Kozachenko and Leonenko [1987] of entropy. Their generative network is an arbitrary deep neural network rather than a normalizing flow, which is more flexible but cannot give the probability density of each sample easily so as to compute an unbiased estimator of the entropy. Kozachenko-Leonenko is a biased estimator for entropy and requires a fairly large number of samples to get good performance in high-dimensional settings, hindering the scalability and accuracy of the method; indeed, our choice of normalizing flow networks was driven by these practical issues with Kozachenko-Leonenko. Lu *et al.* [2016] extended Zhu *et al.* [1998] by using a more flexible set of filters derived from a pre-trained deep neural networks, and using parallel MCMC chains to learn and sample from the Gibbs distribution. Running parallel MCMC chains results in diverse samples but can be computationally intensive for generating each new sample image. Our MEFN framework enables truly iid sampling with the ease of a feed forward network.

Chapter 6

Conclusion and discussion

Big data revolution in neuroscience has brought opportunities and challenges similar to those from many other areas. The ever complicated data requires advanced technology for effective and scalable data processing, more flexible and interpretable modeling, and more careful hypothesis checking and model validation. This chapter gives a summary and discussion for the outlook for each of these steps, and put the preceding chapters in context.

Extracting desirable signals from complicated and noisy observations is an important first step for neural data analysis which requires a thorough understanding and careful modeling of the generative process of data recording. As illustrated in chapter 4, exploiting the structure of the observation enables effective data processing signal extraction. The huge amount of data available calls for methods that scale to large datasets and can process data in an online fashion, while handling the complicated generative process of the observations. Another key challenge for the task is developing systematic methods for judging the effectiveness of the methods, especially in the case when ground truth data is unavailable (which is the case for many of the complicated observation technique).

Even with the clean signal, building statistical models to capture the structure of the large dataset is still challenging. While redundancy in neural data encourages simplified structure and high signal-to-noise ratio, the complex behavior of organisms implies that the brain activity is inherently complicated and highly non-stationary. While dimensionality reduction techniques indicates to that neural activities in certain brain areas tend to lie in low dimensional spaces for fairly simple behavior tasks, with capabilities of whole-brain recordings during complicated behavior calls for much more complicated model structures. Describing the complicated data imposes challenges for not only modeling, but also model fitting and interpreting. Chapter 2 and 3 provides two attempts to more faithfully capture the properties of the signal with flexible modeling. The auto-encoder variational inference discussed in chapter 3 set an example for model fitting for complicated models with scalable inference, which can be highly desirable in the big data regime. One extension of our models is to use a hierarchical model to adapt to the non-stationarity and variability of the data for complicated environments and tasks. It is also important to build more interpretable and biologically plausible dynamical models that represent the underlying biological processes.

While complicated models have shown promise in improving the fit and predictive performance of the neural data, it is less obvious how to use those models to draw scientific conclusions, due to several reasons. First, the growing complexity makes the models hard to interpret. For example, the fLDS discussed in chapter 3 parameterize the nonlinearity with a neural network, which helps with predictive performance but can be hard to interpret. Secondly, the high-dimensional model parameters and possibly non-convex optimization involved makes it hard to do exact inference and understand the statistical error of the inference, making it challenging even to get point estimation, letting alone getting confidence interval, hypothesis testing or model

checking. Therefore, exploiting advanced statistical and computational techniques for scientific pursuit still poses challenges. Chapter 5 is an attempt to facilitate scientific hypothesis testing by proposing a way that can be used to draw null distribution from complicated hypothesis. Future work involves more theoretical understanding of the statistical error of the models, as well as stronger connections between the complicated statistical models and the biological hypotheses.

Despite significant challenges, the exciting interaction of richer neural datasets, more advanced statistical modeling frameworks and computational capability provides promising new directions. And we expect and hope the proposed methods to be inspiring for related research.

Bibliography

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Ryan Prescott Adams, Iain Murray, and David JC MacKay. Tractable nonparametric bayesian inference in poisson processes with gaussian process intensities. In *Proceedings of the 26th International Conference on Machine Learning*, pages 9–16. ACM, 2009.

Misha B Ahrens and Philipp J Keller. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature Methods*, 2013.

Cande V Ananth and David G Kleinbaum. Regression models for ordinal responses: a review of methods and applications. *International Journal of Epidemiology*, 26(6):1323–1333, 1997.

Evan W Archer, Urs Koster, Jonathan W Pillow, and Jakob H Macke. Low-dimensional models of neural population activity in sensory cortical circuits. In *Advances in Neural Information Processing Systems*, pages 343–351, 2014.

Evan Archer, Il Memming Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX, 2010.

Thomas B Berrett, Richard J Samworth, and Ming Yuan. Efficient multivariate entropy estimation via k-nearest neighbour distances. *arXiv preprint arXiv:1606.00304*, 2016.

Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic Press, 2014.

David M. Blei, Michael I. Jordan, and John W. Paisley. Variational bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1367–1374. ACM, 2012.

Oleg Bondarenko. Estimation of risk-neutral densities using positive convolution approximation. *Journal of Econometrics*, 116(1):85–112, 2003.

Jonathan Borwein, Rustum Choksi, and Pierre Maréchal. Probability distributions of assets inferred from option prices via the principle of maximum entropy. *SIAM Journal on Optimization*, 14(2):464–478, 2003.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2009.

Emery N Brown, David P Nguyen, Loren M Frank, Matthew A Wilson, and Victor Solo. An analysis of neural receptive field plasticity by point process adaptive filtering. *Proceedings of the National Academy of Sciences*, 98(21):12261–12266, 2001.

Peter W Buchen and Michael Kelly. The maximum entropy distribution of an asset inferred from option prices. *Journal of Financial and Quantitative Analysis*, 31(01):143–159, 1996.

Lars Buesing, Jakob H Macke, and Maneesh Sahani. Learning stable, regularised latent models of neural population dynamics. *Network: Computation in Neural Systems*, 23(1-2):24–47, 2012.

Lars Buesing, Timothy A Machado, John P Cunningham, and Liam Paninski. Clustered factor analysis of multineuronal spike data. In *Advances in Neural Information Processing Systems*, pages 3500–3508, 2014.

Lars Buesing, Jakob H Macke, and Maneesh Sahani. Estimating state and parameters in state-space models of spike trains. In *Advanced State Space Methods for Neural and Clinical Data*. Cambridge Univ Press., 2015.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Mark M Churchland, John P Cunningham, Matthew T Kaufman, Stephen I Ryu, and Krishna V Shenoy. Cortical preparatory activity: representation of movement or first cog in a dynamical machine? *Neuron*, 2010.

Mark M Churchland, Byron M Yu, John P Cunningham, Leo P Sugrue, Marlene R Cohen, Greg S Corrado, William T Newsome, Andrew M Clark, Paymon Hosseini, Benjamin B Scott, et al. Stimulus onset quenches neural variability: a widespread cortical phenomenon. *Nature Neuroscience*, 13(3):369–378, 2010.

Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

Marlene R Cohen and Adam Kohn. Measuring and interpreting neuronal correlations. *Nature Neuroscience*, 14(7):811–819, 2011.

Michael Collins, Robert E Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1-3):253–285, 2002.

John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 17(71):1500–1509, 2014.

John P Cunningham, Byron M Yu, Krishna V Shenoy, and Sahani Maneesh. Inferring neural firing rates from spike trains using gaussian processes. In *Advances in Neural Information Processing Systems*, pages 329–336, 2007.

John N Darroch and Douglas Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, pages 1470–1480, 1972.

R de Boer and P Kuyper. Triggered correlation. *IEEE Transactions on Bio-medical Engineering*, 15(3):169–179, 1968.

Joan del Castillo and Marta Pérez-Casany. Overdispersed and underdispersed poisson generalizations. *Journal of Statistical Planning and Inference*, 134(2):486–500, 2005.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

Ferran Diego, Susanne Reichinnek, Martin Both, and Fred A Hamprecht. Automated identification of neuronal activity from calcium imaging by sparse dictionary learning. In *Biomedical Imaging (ISBI)*, pages 1058–1061. IEEE, 2013.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Miroslav Dudik, Steven J Phillips, and Robert E Schapire. Performance guarantees for regularized maximum entropy density estimation. In *International Conference on Computational Learning Theory*, pages 472–486. Springer, 2004.

Alexander S Ecker, Philipp Berens, R James Cotton, Manivannan Subramaniyan, George H Denfield, Cathryn R Cadwell, Stelios M Smirnakis, Matthias Bethge, and Andreas S Tolias. State dependence of noise correlations in macaque primary visual cortex. *Neuron*, 82(1):235–248, 2014.

Mohammad Emtiyaz Khan, Aleksandr Aravkin, Michael Friedlander, and Matthias Seeger. Fast dual variational inference for non-conjugate latent gaussian models. In *Proceedings of The 30th International Conference on Machine Learning*, pages 951–959, 2013.

Johannes Friedrich and Liam Paninski. Fast active set methods for online spike inference from calcium imaging. In *Advances In Neural Information Processing Systems*, pages 1984–1992, 2016.

Roger Frigola, Yutian Chen, and Carl Rasmussen. Variational gaussian process state-space models. In *Advances in Neural Information Processing Systems*, pages 3680–3688, 2014.

Peiran Gao and Surya Ganguli. On simplicity and complexity in the brave new world of large-scale neuroscience. *Current Opinion in Neurobiology*, 32:148–155, 2015.

Yuanjun Gao, Lars Busing, Krishna V Shenoy, and John P Cunningham. High-dimensional neural spike train analysis with generalized count linear dynamical systems. In *Advances in Neural Information Processing Systems*, pages 2035–2043, 2015.

Yuanjun Gao, Evan W Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. In *Advances in Neural Information Processing Systems*, pages 163–171, 2016.

Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

George L Gerstein and Nelson Y-S Kiang. An approach to the quantitative analysis of electrophysiological data from single neurons. *Biophysical Journal*, 1(1):15–28, 1960.

Vikash Gilja, Paul Nuyujukian, Cindy A Chestek, John P Cunningham, M Yu Byron, Joline M Fan, Mark M Churchland, Matthew T Kaufman, Jonathan C Kao, Stephen I Ryu, et al. A high-performance neural prosthesis enabled by control algorithm design. *Nature Neuroscience*, 15(12):1752–1757, 2012.

Irving J Good. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *The Annals of Mathematical Statistics*, pages 911–934, 1963.

Robbe LT Goris, J Anthony Movshon, and Eero P Simoncelli. Partitioning neuronal variability. *Nature Neuroscience*, 17(6):858–865, 2014.

Arnulf BA Graf, Adam Kohn, Mehrdad Jazayeri, and J Anthony Movshon. Decoding the activity of neuronal populations in macaque primary visual cortex. *Nature Neuroscience*, 14(2):239–245, 2011.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Edwin T Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620, 1957.

Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach. Structured sparse principal component analysis. *arXiv preprint arXiv:0909.1440*, 2009.

Jiantao Jiao, Kartik Venkat, Yanjun Han, and Tsachy Weissman. Minimax estimation of functionals of discrete distributions. *IEEE Transactions on Information Theory*, 61(5):2835–2885, 2015.

Jonathan C. Kao, Paul Nuyujukian, Stephen I. Ryu, Mark M. Churchland, John P. Cunningham, and Krishna V. Shenoy. Single-trial dynamics of motor cortex and their applications to brain-machine interfaces. *Nature Communications*, 6:7759+, July 2015.

Robert E Kass, Valérie Ventura, and Emery N Brown. Statistical issues in the analysis of neuronal data. *Journal of Neurophysiology*, 94(1):8–25, 2005.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

Shinsuke Koyama. On the spike train variability characterized by variance-to-mean power relationship. *Neural Computation*, 2015.

LF Kozachenko and Nikolai N Leonenko. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16, 1987.

Jayant E Kulkarni and Liam Paninski. Common-input models for multiple neural spike-train data. *Network: Computation in Neural Systems*, 18(4):375–407, 2007.

Diane Lambert. Zero-inflated poisson regression, with an application to defects in manufacturing. *Technometrics*, 34(1):1–14, 1992.

Vernon Lawhern, Wei Wu, Nicholas Hatsopoulos, and Liam Paninski. Population decoding of motor cortical activity using a generalized linear model with hidden states. *Journal of Neuroscience Methods*, 189(2):267–280, 2010.

Neil D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in Neural Information Processing Systems*, 16(3):329–336, 2004.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

Jeremy Lewi, David M Schneider, Sarah MN Woolley, and Liam Paninski. Automating the design of informative sequences of sensory stimuli. *Journal of computational neuroscience*, 30(1):181–200, 2011.

Michael S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53–R78, 1998.

Yingzhen Li and Richard E Turner. Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*, pages 1073–1081, 2016.

Scott W. Linderman, Ryan Adams, and Jonathan Pillow. Inferring structured connectivity from spike trains under negative-binomial generalized linear models. *COSYNE*, 2015.

Gabriel Loaiza-Ganem, Yuanjun Gao, and John P Cunningham. Maximum entropy flow networks. In *International Conference of Learning Representations (ICLR)*, 2017.

Yang Lu, Song-chun Zhu, and Ying Nian Wu. Learning FRAME models using cnn filters. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. In *Advances in Neural Information Processing Systems*, pages 1350–1358, 2011.

Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language learning*, volume 20, pages 1–7. Association for Computational Linguistics, 2002.

Ryuichi Maruyama, Kazuma Maeda, Hajime Moroda, Ichiro Kato, Masashi Inoue, Hiroyoshi Miyakawa, and Toru Aonishi. Detecting cells using non-negative matrix factorization on calcium imaging data. *Neural Networks*, 55:11–19, 2014.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Eran A Mukamel, Axel Nimmerjahn, and Mark J Schnitzer. Automated analysis of cellular signals from large-scale calcium imaging data. *Neuron*, 63(6):747–760, 2009.

I Nelken, Y Prut, E Vaadia, and M Abeles. In search of the best stimulus: an optimization procedure for finding efficient stimuli in the cat auditory cortex. *Hearing Research*, 72(1):237–253, 1994.

Michael Okun, Pierre Yger, Stephan L Marguet, Florian Gerard-Mercier, Andrea Benucci, Steffen Katzner, Laura Busse, Matteo Carandini, and Kenneth D Harris. Population rate dynamics and multineuron firing patterns in sensory cortex. *The Journal of Neuroscience*, 32(48):17108–17119, 2012.

Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

Marius Pachitariu, Adam M Packer, Noah Pettit, Henry Dalgleish, Michael Hausser, and Maneesh Sahani. Extracting regions of interest from biological images with convolutional sparse block coding. In *Advances in Neural Information Processing Systems*, pages 1745–1753, 2013.

Liam Paninski, Jonathan Pillow, and Jeremy Lewi. Statistical models for neural encoding, decoding, and optimal stimulus design. *Progress in Brain Research*, 165:493–507, 2007.

Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua Vogelstein, and Wei Wu. A new look at state-space models for neural data. *Journal of Computational Neuroscience*, 29(1-2):107–126, 2010.

Liam Paninski. Maximum likelihood estimation of cascade point-process neural encoding models. *Network: Computation in Neural Systems*, 15(4):243–262, 2004.

Biljana Petreska, Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Dynamical segmentation of single trials from population neural data. In *Advances in Neural Information Processing Systems*, pages 756–764, 2011.

David Pfau, Eftychios A Pnevmatikakis, and Liam Paninski. Robust learning of low-dimensional dynamics from large neural ensembles. In *Advances in Neural Information Processing Systems*, pages 2391–2399, 2013.

Steven J Phillips, Robert P Anderson, and Robert E Schapire. Maximum entropy modeling of species geographic distributions. *Ecological Modelling*, 190(3):231–259, 2006.

Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999, 2008.

Eftychios A Pnevmatikakis and Andrea Giovannucci. Normcorre: An online algorithm for piecewise rigid motion correction of calcium imaging data. *bioRxiv*, page 108514, 2017.

Eftychios A Pnevmatikakis and Liam Paninski. Sparse nonnegative deconvolution for compressive calcium imaging: algorithms and phase transitions. In *Advances in Neural Information Processing Systems*, pages 1250–1258, 2013.

Eftychios A Pnevmatikakis, Daniel Soudry, Yuanjun Gao, Timothy A Machado, Josh Merel, David Pfau, Thomas Reardon, Yu Mu, Clay Lacefield, Weijian Yang, et al. Simultaneous denoising, deconvolution, and demixing of calcium imaging data. *Neuron*, 89(2):285–299, 2016.

Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000.

Ruben Portugues, Claudia E Feierstein, Florian Engert, and Michael B Orger. Whole-brain activity maps reveal stereotyped, distributed networks for visuomotor behavior. *Neuron*, 81(6):1328–1343, 2014.

Robert Prevedel, Young-Gyu Yoon, Maximilian Hoffmann, Nikita Pak, Gordon Wetzstein, Saul Kato, Tina Schrödel, Ramesh Raskar, Manuel Zimmer, Edward S Boyden, et al. Simultaneous whole-animal 3d imaging of neuronal activity using light-field microscopy. *Nature Methods*, 11(7):727–730, 2014.

Rajesh Ranganath, Sean Gerrish, and David M Blei. Black box variational inference. *arXiv preprint arXiv:1401.0118*, 2013.

C Radhakrishna Rao. On discrete distributions arising out of methods of ascertainment. *Sankhya: The Indian Journal of Statistics, Series A*, pages 311–324, 1965.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

Jacob T Robinson, Marsela Jorgolli, Alex K Shalek, Myung-Han Yoon, Rona S Gertner, and Hongkun Park. Vertical nanowire electrode arrays as a scalable platform for intracellular interfacing to neuronal circuits. *Nature Nanotechnology*, 7(3):180–184, 2012.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

Patrick T Sadtler, Kristin M Quick, Matthew D Golub, Steven M Chase, Stephen I Ryu, Elizabeth C Tyler-Kabara, Byron M Yu, and Aaron P Batista. Neural constraints on learning. *Nature*, 512(7515):423–426, 2014.

Maneesh Sahani. *Latent variable models for neural data analysis*. PhD thesis, California Institute of Technology, 1999.

Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. On the convergence of bound optimization algorithms. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 509–516. Morgan Kaufmann Publishers Inc., 2002.

Elad Schneidman, Michael J Berry, Ronen Segev, and William Bialek. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087):1007–1012, 2006.

James Scott and Jonathan W Pillow. Fully bayesian inference for neural models with negative-binomial spiking. In *Advances in Neural Information Processing Systems*, pages 1898–1906, 2012.

Kimberly F Sellers and Galit Shmueli. A flexible regression model for count data. *The Annals of Applied Statistics*, pages 943–961, 2010.

Krishna V Shenoy, Matthew T Kaufman, Maneesh Sahani, and Mark M Churchland. A dynamical systems view of motor preparation: implications for neural prosthetic system design. *Progress in Brain Research*, 192:33, 2011.

Krishna V Shenoy, Maneesh Sahani, and Mark M Churchland. Cortical control of arm movements: a dynamical systems perspective. *Annual Review of Neuroscience*, 36:337–359, 2013.

Zhenming Shun and Peter McCullage. Laplace approximation of high dimensional integrals. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 749–760, 1995.

Jagbir Singh. A characterization of positive poisson distribution and its statistical application. *SIAM Journal on Applied Mathematics*, 34(3):545–548, 1978.

Spencer L Smith and Michael Häusser. Parallel processing of visual space by neighboring neurons in mouse visual cortex. *Nature Neuroscience*, 13(9):1144–1149, 2010.

Ian H Stevenson and Konrad P Kording. How advances in neural recording affect data analysis. *Nature Neuroscience*, 14(2):139–142, 2011.

Ian H Stevenson, James M Rebesco, Lee E Miller, and Konrad P Körding. Inferring functional connections between neurons. *Current opinion in neurobiology*, 18(6):582–588, 2008.

Aonan Tang, David Jackson, Jon Hobbs, Wei Chen, Jodi L Smith, Hema Patel, Anita Prieto, Dumitru Petrusca, Matthew I Grivich, Alexander Sher, et al. A maximum entropy model applied to spatial and temporal correlations from cortical networks in vitro. *Journal of Neuroscience*, 28(2):505–518, 2008.

Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Frédéric E Theunissen, Stephen V David, Nandini C Singh, Anne Hsu, William E Vinje, and Jack L Gallant. Estimating spatio-temporal receptive fields of auditory and visual neurons from their responses to natural stimuli. *Network: Computation in Neural Systems*, 12(3):289–316, 2001.

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1971–1979, 2014.

Wilson Truccolo, Uri T Eden, Matthew R Fellows, John P Donoghue, and Emery N Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology*, 93(2):1074–1089, 2005.

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *arXiv preprint arXiv:1603.03417*, 2016.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. *arXiv preprint arXiv:1701.02096*, 2017.

Paul Valiant and Gregory Valiant. Estimating the unseen: improved estimators for entropy and other properties. In *Advances in Neural Information Processing Systems*, pages 2157–2165, 2013.

Michael Vidne, Yashar Ahmadian, Jonathon Shlens, Jonathan W Pillow, Jayant Kulkarni, Alan M Litke, EJ Chichilnisky, Eero Simoncelli, and Liam Paninski. Modeling the impact of common noise inputs on the network activity of retinal ganglion cells. *Journal of Computational Neuroscience*, 33(1):97–121, 2012.

Joshua T Vogelstein, Adam M Packer, Tim A Machado, Tanya Sippy, Baktash Babadi, Rafael Yuste, and Liam Paninski. Fast non-negative deconvolution for spike train inference from population calcium imaging. *arXiv preprint arXiv:0912.1637*, 2009.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

Greg CG Wei and Martin A Tanner. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American statistical Association*, 85(411):699–704, 1990.

Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Journal of Neurophysiology*, 102(1):614–635, 2009.

Matthew D Zeiler. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Yuan Zhao and Il Memming Park. Variational latent gaussian process for recovering single-trial dynamics from population spike trains. *arXiv preprint arXiv:1604.03053*, 2016.

Pengcheng Zhou, Shanna L Resendez, Garret D Stuber, Robert E Kass, and Liam Paninski. Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data. *arXiv preprint arXiv:1605.07266*, 2016.

Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.