Black Box Variational Inference: Scalable, Generic Bayesian Computation and its Applications

Rajesh Ranganath

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: David M. Blei

November 2017

Abstract

Probabilistic generative models are robust to noise, uncover unseen patterns, and make predictions about the future. These models have been used successfully to solve problems in neuroscience, astrophysics, genetics, and medicine. The main computational challenge is computing the hidden structure given the data—posterior inference. For most models of interest, computing the posterior distribution requires approximations like variational inference. Variational inference transforms posterior inference into optimization. Classically, this optimization problem was feasible to deploy in only a small fraction of models.

This thesis develops *black box variational inference*. Black box variational inference is a variational inference algorithm that is easy to deploy on a broad class of models and has already found use in models for neuroscience and health care. It makes new kinds of models possible, ones that were too unruly for previous inference methods.

One set of models we develop is deep exponential families. Deep exponential families uncover new kinds of hidden pattens while being predictive of future data. Many existing models are deep exponential families. Black box variational inference makes it possible for us to quickly study a broad range of deep exponential families with minimal added effort for each new type of deep exponential family.

The ideas around black box variational inference also facilitate new kinds of variational methods. First, we develop hierarchical variational models. Hierarchical variational models improve the approximation quality of variational inference by building higher-fidelity approximations from coarser ones. We show that they help with inference in deep exponential families. Second, we introduce operator variational inference. Operator variational inference delves into the possible distance measures that can be used for the variational optimization problem. We show that this formulation categorizes various variational inference methods and enables variational approximations without tractable densities.

By developing black box variational inference, we have opened doors to new models, better posterior approximations, and new varieties of variational inference algorithms.

Acknowledgements

The work presented was supported by a wide array of individuals and organizations.

The research here was mainly supported by fellowships: the Princeton First Year Fellowship, the National Defense Science and Engineering Fellowship, the Siebel Scholarship, and finally the Porter Ogden Jacobus Fellowship.

My adviser David Blei has been a great mentor and collaborator. The complete freedom he provided to let me pursue my own interests made my PhD more fulfilling than it would have been otherwise. I am grateful for the time he spent guiding me.

I thank the members of my PhD committee: Sanjeev Arora, Barbara Engelhardt, and Peter Orbanz. Peter Orbanz's encouragement to pursue my interest in theory, though not related to my core research, was invaluable. Sanjeev Arora's "outsider's" perspective on machine learning has been fascinating.

I thank my collaborators in neuroscience: Ken Norman and Jeremy Manning. Their difficulty in computing with models of the brain led me towards developing easier to use inference. I am grateful for my collaborators in medicine, Noémie Elhadad and Adler Perotte, without whom I would not have been able to start pursuing my interest in healthcare. I thank Rimma Perotte and Sharon Gorman who both helped immensely with wrangling the medical data.

Princeton University and the Department of Computer Science have been extremely supportive of my endeavors, even though my physical time there has been limited after my adviser moved. The PICSciE staff were amazing at addressing my compute problems. The department administrators, especially Pamela DelOrefice, have been immensely helpful. Robert Schapire's clarity in explanation remains an inspiration. Erhan Cinlar's willingness to spend time explaining both the small and large things was unmatched.

I thank one of my undergraduate advisers Dan Jurafsky. Without his encouragement and the opportunities to do research he provided as an undergraduate, I would not be on this path.

My graduate experience would not have been the same without the many students and post-docs with whom I have interacted. These individuals include Jaan Altosaar, Alison Chaney, Laurent Charlin, Sean Gerrish, Prem Gopalan, Alp Kucukelbir, James Li, Scott Linderman, Jeremy Manning, James McInerney, David Mimno, Christian Naesseth, Reid Oda, Joan Ricart, Francisco Ruiz, Bharat Srikishan, Linpeng Tang, Dustin Tran, and Chong Wang.

Finally, I would like to thank my family who support me in all my endeavors, and to ajji whose unwavering backing was something I truly cherished.

Contents

Abstract iii

Acknowledgements v

List of Tables xii

List of Figures xiii

1 Introduction **1**

2 Probabilistic Models **5**

2.1 Goals of Modeling Data 5

2.2 Generative Probabilistic Models 6

2.2.1 Generative Probabilistic Models 7

2.3 Examples 9

2.3.1 Dynamical Systems 9

2.3.2 Sigmoid Belief Network 10

2.3.3 Mechanistic Models 12

2.3.4 Model Spectrum 13

2.4 Posterior Distribution 14

2.5 Exact Computation 15

2.6 Importance Sampling 15

2.7 Markov Chain Monte Carlo 16

2.7.1 Metropolis-Hastings 17

2.7.2 Gibbs Sampling . . . 18
2.7.3 Hamiltonian Monte Carlo . . . 19
2.8 Variational Inference . . . 20
2.9 Stochastic Variational Inference . . . 22
2.9.1 Model Class . . . 23
2.9.2 Approximating Family . . . 24
2.9.3 Coordinate Ascent Variational Inference . . . 25
2.9.4 Stochastic Variational Inference . . . 27
2.10 Conclusion . . . 29
2.11 Appendix . . . 30
2.11.1 Gradient of the evidence lower bound (ELBO) with respect to $\boldsymbol{\phi}_i$ . . . 30

3 Black Box Variational Inference **31**
3.1 A Case Study: Bayesian Logistic Regression . . . 32
3.2 Black Box Variational Inference . . . 34
3.3 Controlling the Variance . . . 37
3.3.1 Rao-Blackwellization . . . 38
3.3.2 Control Variates . . . 40
3.3.3 Reparameterization Gradients . . . 45
3.3.4 Conjugacy and Additional Information . . . 49
3.3.5 Data Subsampling . . . 51
3.4 Related Work . . . 53
3.5 Results . . . 53
3.5.1 Longitudinal Medical Data . . . 54
3.5.2 Evaluation and Hyperparameters . . . 54
3.5.3 Model . . . 54
3.5.4 Sampling Methods . . . 56
3.5.5 Exploring Models . . . 56

3.6 Conclusion 59

4 Deep Exponential Families **60**

4.1 Deep Exponential Families 62

4.2 Examples 66

4.2.1 Sparse Gamma DEF 66

4.2.2 Sigmoid Belief Network 68

4.2.3 Poisson DEF 69

4.3 Connections 70

4.4 Inference 77

4.5 Experiments 78

4.5.1 Text Modeling 79

4.5.2 Composing DEFs 81

4.6 Conclusion 87

5 Hierarchical Variational Models **88**

5.1 Hierarchical Variational Models 89

5.1.1 Hierarchical Variational Models 90

5.1.2 Specifying an HVM 92

5.2 Optimizing HVMs 95

5.2.1 Hierarchical Evidence Lower Bound 95

5.2.2 Stochastic Gradient of the Hierarchical ELBO 99

5.2.3 Choosing r 100

5.2.4 Optimizing the Hierarchical ELBO with Respect to $\boldsymbol{\phi}$ 103

5.2.5 Algorithm 103

5.2.6 Inference Networks 104

5.2.7 Related Work 104

5.3 Empirical Study 106

5.3.1 Correlated Discrete Latent Variables 106
5.3.2 Deep Exponential Families 107
5.4 Discussion 109
5.5 Appendix 110
5.5.1 Stochastic Gradient of the Hierarchical ELBO 110
5.5.2 Rao-Blackwellization and the Recursive Approximation r 112
5.5.3 Relationship to Empirical Bayes+Reinforcement Learning 113
5.5.4 Multi-level $q(\boldsymbol{\nu}; \boldsymbol{\theta})$ and Optimizing with Discrete Variables in the Variational Prior 114

6 Operator Variational Inference **115**
6.1 Operator Variational Objectives 117
6.1.1 Operator Variational Objectives 117
6.1.2 Understanding Operator Variational Objectives 118
6.1.3 Langevin-Stein Operator Variational Objective 119
6.1.4 The KL Divergence as an Operator Variational Objective 122
6.1.5 Control Variates 123
6.2 Operator Variational Inference 124
6.3 Characterizing Variational Methods with Operators 127
6.3.1 Data Subsampling and OPVI 128
6.3.2 Variational Programs 129
6.4 Empirical Study 130
6.4.1 Mixture of Gaussians 131
6.4.2 Logistic Factor Analysis 132
6.5 Summary 134
6.6 Appendix 135
6.6.1 Technical Conditions for Langevin-Stein Operators 135
6.6.2 Characterizing the Zeros of the Langevin-Stein Operators 136

7 Discussion **138**

Bibliography **141**

List of Tables

3.1 Model comparison with black box variational inference 58

4.1 DEF results on text 81

4.2 Double DEF results 82

4.3 Concordance on 25,000 patients for CHD 86

4.4 Comparing data types 87

5.1 Data models vs. variational models 89

5.2 Complexity of variational approximations 104

5.3 Comparing hierarchical variational model on *the New York Times* 107

5.4 Comparing hierarchical variational models on *Science* 108

6.1 Comparison of operator variational inference methods 133

List of Figures

2.1 Graphical model for the Kalman filter 10

2.2 Sigmoid belief network 11

2.3 Modeling spectrum 13

2.4 Variational inference 20

2.5 Graphical model for hierarchical Bayesian models 23

3.1 Recipe for variational inference 32

3.2 New recipe for variational inference 34

3.3 Gaussian mean score function 37

3.4 Variance comparison with and without leave-one-out control variates 44

3.5 Variance between reparameterization, score, and leave-one-out (16 samples) 47

3.6 Variance between reparameterization, score, and leave-one-out (8 samples) 48

3.7 Comparison between Metropolis-Hastings inside Gibbs and black box variational inference 57

4.1 Visualization of a DEF on *the New York Times* 61

4.2 Probability density of gamma distributions 67

4.3 Sparse gamma distribution vs. Poisson distribution 68

4.4 Transforming a uniform to gamma with a neural network 73

5.1 Graphical models for HVMS 90

5.2 A simple HVM 91

5.3 HVM for a discrete posterior 106

6.1 Variational inference revisited 116

6.2 Operator variational inference equivalences 131

6.3 Operator variational inference and Gaussian mixtures 131

Chapter 1

Introduction

Data are a staple of many disciplines. The goal of working with data is manyfold. First, we would like to be able to find hidden structure—structure that cannot be read off directly from the data. Second, we want to predict missing parts of our data or whole new instances of data. Third, we want to be able to build prior knowledge into our models, thereby reducing the data needed.

We can meet these goals with generative probabilistic models. Generative probabilistic models contain three parts: observed data, hidden structure, and a probability function that scores different combinations of data and hidden structure. The hidden structure can be used to represent concepts like "factors" or "groupings," while the rules of probability provide predictions. The scoring probability function encodes prior knowledge. Probabilistic generative models have been successfully deployed to find population structure in genetics (Pritchard et al., 2000), to infer brain connectivity in neuroscience (Manning et al., 2014), to make recommendations (Wang and Blei, 2011) in user behavior platforms, and in many other domains.

The core computational task in working with generative probabilistic models is computing the posterior distribution — the distribution of the hidden structure given the observations. This computation is known as posterior inference or Bayesian inference.[1] Computing the posterior requires computing a high dimensional integral or sum. For most models this computation is intractable.

To address this limitation, users of generative probabilistic models rely on approximate posterior inference methods. These fall into two classes: those that simulate from the posterior and those that try to match another distribution to the posterior via optimization. This latter process is called variational inference. Variational inference has been used to scale posterior inference to massive scientific data (Regier et al., 2015). The main challenge in using variational inference stems from the need to do analysis for every new model. This requirement limits the use of variational inference and forces users to build models for easy-to-derive inference rather than build models for their data. This thesis concerns itself with how to relax the constraints on variational inference methods and the implications of doing so.

The remainder of this thesis is organized as follows.

We begin by introducing probabilistic models in Chapter 2. This chapter sets the goals for modeling data and describes how generative probabilistic models meet these goals. We then follow with examples of several generative probabilistic models. After this, we introduce the central problem of working with these models: posterior inference. Computing the posterior is intractable, so we review posterior approximation algorithms with a focus on variational inference where it is easy to do, i.e., when models are conditionally conjugate. Along the way, we review stochastic optimization.

In Chapter 3, we address the core limitation of traditional variational inference methods — they require extensive mathematical derivations and model-specific analysis for each new

[1] The terms posterior inference and Bayesian inference can be exchanged in most places in this thesis.

use case. We develop black box variational inference (BBVI). This is based on Ranganath et al. (2014). BBVI makes use of stochastic optimization to step around the analytic difficulties in traditional variational inference. The variance of the stochastic gradients can be high, so we introduce several techniques to control this variance without requiring model-specific analysis. We use BBVI to explore a set of models for medical lab values. Black box variational inference requires minimal model-specific derivations and outperforms the corresponding Markov chain Monte Carlo technique.

Black box variational inference expands the scope of possible models. In Chapter 4, we introduce one such class of models, deep exponential families (DEFS). DEFS are a multilayer probabilistic model that build upon the structures that underlie deep neural networks. We show that several existing models fall into this class and develop new models based on cascades of Poisson variables and on cascades of sparse gamma variables. DEFS outperform competitors on problems in text and recommendation, while also yielding interpretable, explorable hidden structure. This chapter follows Ranganath et al. (2015b, 2016a).

Black box variational inference also makes better posterior approximations feasible. Variational inference approximates the posterior distribution with a distribution from a tractable approximating family. Traditionally, this approximating family was chosen to make variational inference easy to derive. With BBVI, making such a choice is no longer necessary. We introduce hierarchical variational models (HVMS) in Chapter 5. HVMS construct richer approximating families from simpler ones by placing priors over the simple families. This chapter takes the view that tools that enrich models of data can be used to enrich variational approximations. We demonstrate that HVMS improve inference for a wide range of deep exponential families. This work was presented in Ranganath et al. (2016c).

The most common form of variational inference relies on the Kullback-Leibler (KL) divergence. This reliance partly stemmed from the fact that the KL divergence led to nice updates for conditionally conjugate models. With BBVI, we are no longer limited to conditionally

conjugate models and are more free to explore alternative notions of distance between probability distributions. In Chapter 6, we develop operator variational inference (OPVI). This is based on Ranganath et al. (2016b). Operator variational inference constructs a broad class of objectives that can be used for variational inference. We use OPVI to characterize the kinds of variational objectives, and we show that certain kinds of operators allow the use of variational approximations without tractable densities, thus creating larger variational families to approximate against.

Lastly, in Chapter 7 we review the key themes and discuss directions for future work.

Chapter 2

Probabilistic Models

This chapter lays the foundations of probabilistic models, provides example uses, and presents the main computational challenges associated with these models.

2.1 Goals of Modeling Data

Data consists of collections of observations of the world. Building models of data stems from the desire to understand the process that generated the data using observations from that process. The goals of building models can be both to find unobserved structure in the data and to make predictions about the data. To aid in model construction, models should be able to incorporate prior knowledge about the data generating process. These considerations form three desiderata for a modeling framework: uncovering hidden structure, making predictions, and specifying prior knowledge. We go into these in more detail.

- *Uncovering Hidden Structure*: Hidden structures are relationships within data not explicitly observed. Examples include grouping of tumor cells into tumor types or population structure in genetic measurements of people. In this sense, models used to

find hidden structures are measurement devices, and the quality of the measurement depends on how well the model explains new data from the same process.

- *Making Predictions*: Predictions are completions of data. Examples include determining house prices from house age and size, survival time for cancer from genetic sequences, and graduation status given family income and location. Models use smoothness assumptions to complete data on new observations. The amount of faith practitioners should place in the hidden structure recovered by a model depends on how well the model predicts.

- *Expressing Prior Knowledge*: Prior knowledge is information about the data process independent of the collected data. Examples include the spatial structure of the basketball court or properties of a physical system (Allanach et al., 2007). All of the value in modeling comes from two sources: prior knowledge and observed data. Any prediction or inference must be supported by one of these two sources of information.

These criteria provide requirements for a modeling framework.

2.2 Generative Probabilistic Models

Generative probabilistic models tell stories about how the data were generated. While never capturing the entire detail, these stories encode the core features of the data. For example, when modeling news documents, it is common to assume that documents have no temporal structure. Though this assumption ignores information, these models still capture the core content of individual documents.

2.2.1 Generative Probabilistic Models

The ingredients to define a generative probabilistic model are the observed data $\boldsymbol{x}$, the hidden structure (or latent variables) $\boldsymbol{z}$, and a joint probability distribution p over the hidden structure and data. The model's joint distribution provides a score for all possible $\boldsymbol{x}, \boldsymbol{z}$ configurations. The model tells the story of how the data were generated through the factorization of the model's joint distribution.

As an example, consider a mixture model. In a mixture model we have n observed data points $\boldsymbol{x}_{1:n}$. The mixture model narrates that each data point comes from one of K classes. These classes represent the hidden structure. To explicitly encode this structure, we assign to each data point a latent variable z_i that takes one of K possible values. The last thing we need to specify this model is the probability distribution that connects the variables:

$$p(\boldsymbol{x}_{1:n}, \boldsymbol{z}_{1:n}) = \prod_{i=1}^{n} p(z_i)\, p(\boldsymbol{x}_i \mid z_i).$$

Here $p(z_i)$ is the prior probability of selecting each of the K classes, and $p(\boldsymbol{x}_i \mid z_i)$ is the likelihood of the ith observation when it is assigned to the class z_i. If $\boldsymbol{\beta}_k$ denotes parameters that identify each of the classes, the likelihood is

$$p(\boldsymbol{x}_i \mid z_i) = p(\boldsymbol{x}_i; \boldsymbol{\beta}_{z_i}). \tag{2.1}$$

Note that there is one parameter for each class. The joint distribution encodes the story that the mixture model tells about the observations. First for each point choose a class (draw from $p(z_i)$), then given that class draw from the likelihood (Equation (2.1)).

To find the unobserved classes for each data point, we need to compute the distribution of the hidden structure given the observations, *the posterior distribution*:

$$p(z_i \mid \boldsymbol{x}_i) = \frac{p(\boldsymbol{x}_i \mid z_i)p(z_i)}{p(\boldsymbol{x}_i)}.$$

In terms of distributions specified in the model, we get

$$p(z_i = k \mid \boldsymbol{x}_i) = \frac{p(\boldsymbol{x}_i \mid z_i = k)p(z_i = k)}{\sum_{j=1}^{K} p(\boldsymbol{x}_i \mid z_i = j)p(z_i = j)}.$$

In general the normalization constant $p(\boldsymbol{x}_i)$ and thus the posterior will be computationally intractable. This challenge is a central theme of subsequent chapters.

Randomness. The randomness in generative probabilistic models comes from multiple sources. First, the randomness could be intrinsic—our measurements may be noisy simply due to the tolerance of the measurement device. Second, the randomness could be due to missing values—the system is partially observed, so the entire state is uncertain. Finally, the randomness (or noise) accounts for model mismatch—if the model explains only a fraction of the data, the rest must be explained by noise. The randomness in generative probabilistic models stems from a combination of these sources.

Did we meet our desiderata? To start this chapter, we laid out criteria for modeling data. Here we will address whether generative probabilistic models meet our criteria.

- *Uncovering Hidden Structure*: The latent variables $\boldsymbol{z}$ represent the posited hidden structure in generative probabilistic models. To find the hidden structure from observed data $\boldsymbol{x}$, we compute $p(\boldsymbol{z} \mid \boldsymbol{x})$, the posterior distribution.

- *Making Predictions*: Predictions of new data $\boldsymbol{x}^*$ can be made by conditioning on the observed data and averaging over all hidden structure configurations. Each config-

uration is weighted by the probability of that configuration of the hidden structure, given the observations:

$$p(\boldsymbol{x}^* \mid \boldsymbol{x}) = \int p(\boldsymbol{x}^* \mid \boldsymbol{z}, \boldsymbol{x}) p(\boldsymbol{z} \mid \boldsymbol{x}) d\boldsymbol{z}.$$

Again, the central requirement for making predictions is the posterior distribution.

- *Expressing Prior Knowledge*: The types of hidden structure, the factorization of the model's joint probability distribution, and the form of each individual factor can be used to encode prior knowledge. A simple example would be to use a distribution with positive support when building a predictive model of test scores.

In summary, generative probabilistic models meet our desiderata for a framework to model data. Given data, the posterior distribution of generative probabilistic models encodes the salient structure.

2.3 Examples

We now describe several classes of generative probabilistic models.

2.3.1 Dynamical Systems

Dynamical systems model observations over time. The canonical example is the linear dynamical system or Kalman filter (Kalman, 1960). The linear dynamical system models a sequence of observed values $\boldsymbol{x}_{1:T}$ with a sequence of hidden states $\boldsymbol{z}_{1:T}$. The hidden states $\boldsymbol{z}_t$ evolve via a Gaussian transition with parameters A, B: $\boldsymbol{z}_t \sim \mathcal{N}(A\boldsymbol{z}_{t-1}, B)$. The observation $\boldsymbol{x}_t$ at time t is conditionally independent of all the other time steps given $\boldsymbol{z}_t$. In this sense, $\boldsymbol{z}_t$ captures the state of the system, and it is what evolves over time.

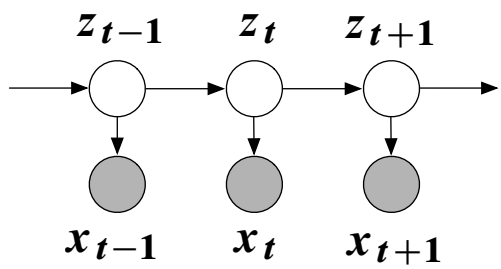


Figure 2.1: The graphical model for the Kalman filter. Shaded nodes are observed data. Unshaded nodes are latent variables. The observations are independent given the latent variables.

Figure 2.1 depicts the graphical model for the Kalman filter. Graphical models describe the factorization of the model's joint distribution. Simulation from a node needs only the arrows pointing into to it. The graphical model implies a set of conditional independences. These conditional independences can be read off using the Bayes-ball algorithm or d-separation (Bishop, 2006). Graphical models do not specify the form of each conditional distribution, thus Figure 2.1 also depicts more general dynamical systems, ones that may be non-Gaussian. Dynamical systems form the backbone of many models in signal processing, statistics, and machine learning. They have been used to study neural spike trains (Linderman et al., 2017), language (Bowman et al., 2016), health records (Ranganath et al., 2015a), and video (Weng et al., 2006).

2.3.2 Sigmoid Belief Network

The sigmoid belief network (Neal, 1990) models a collection of data points $\boldsymbol{x}_{1:n}$ as independent samples from a series of transformed binary variables. This sequential structure is depicted in Figure 2.2. Let $\boldsymbol{z}_\ell$ be the ℓth layer of binary variables for a single observation, and W_ℓ be parameters for the ℓth layer. Then conditional on the layer above, the sigmoid

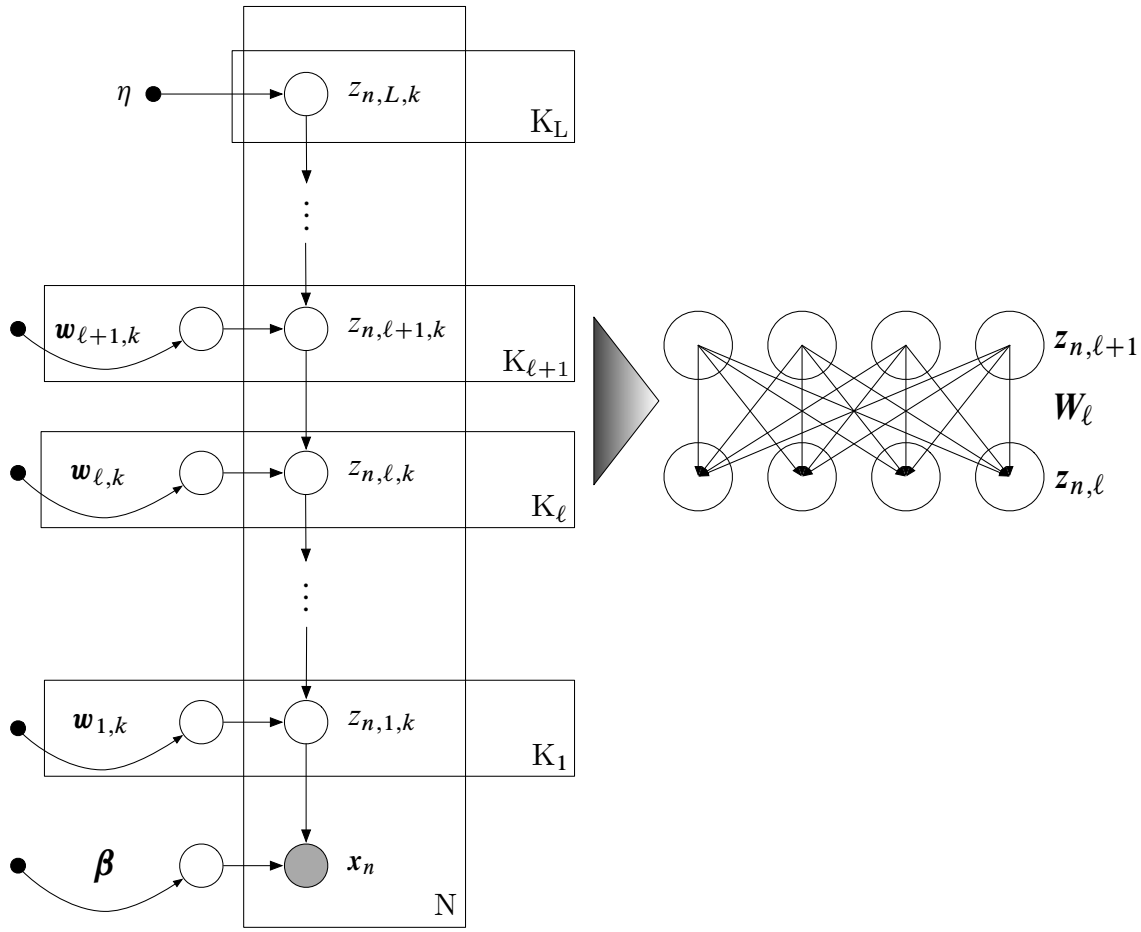


Figure 2.2: The graphical model for the sigmoid belief network. Each layer contains Bernoulli variables that depend on all of the previous layer's variables. The dependency is controlled by the weights.

belief network says

$$z_\ell \sim \text{Bernoulli}(\sigma(W_\ell z_{\ell+1})),$$

where σ denotes the elementwise sigmoid function $\sigma(a) = (1+\exp(-a))^{-1}$.[1] To complete the specification of the sigmoid belief network, we need to specify a prior on the highest level and specify how each data point is generated given the hidden structure. If there are

[1]This is the multivariate vector of independent Bernoullis.

L layers in the sigmoid belief network, the prior variable k is

$$\boldsymbol{z}_{L,k} \sim \text{Bernoulli}(\sigma(\eta)).$$

Lastly, the data come from a parametric likelihood,

$$\boldsymbol{x}_i \sim p(\boldsymbol{x}_i \mid \boldsymbol{z}_1).$$

The main computational task for this model is to estimate the distributions of the unobserved hidden state $\boldsymbol{z}$ given the data $\boldsymbol{x}$.

The sigmoid belief network encodes very little prior knowledge. These models were constructed to learn arbitrary distributions (over binary vectors), with the speed at which they learned as a main consideration (Neal, 1990). In this sense, it is an *empirical model*. Empirical models use a large set of parameters to mimic the data generating process with minimal data and computation requirements.

2.3.3 Mechanistic Models

In contrast to empirical models, *mechanistic models* describe a data generating process using knowledge of the world rather than mimicry of observed data. Mechanistic models can be defined as models that on known laws of nature. Being based on the laws of nature, such models tend encode causal structure. The canonical example of a mechanistic model would be to use Newton's laws of motion to describe an undamped spring system. This model can characterize observations from a spring system given the spring constant. For another example, predator-prey systems can be modeled using the Lotka-Volterra coupled differential equations (Wilkinson, 2011). If x_1 is the prey population, x_2 is the predator

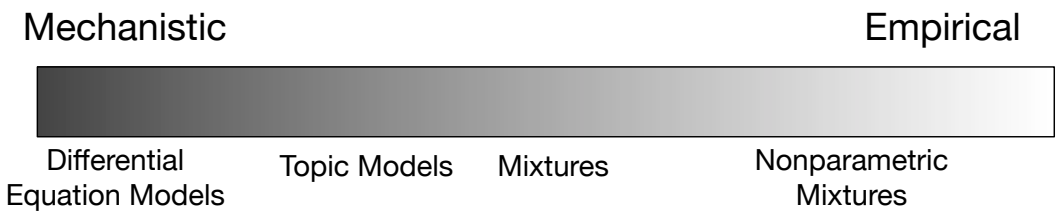


Figure 2.3: The spectrum of models ranges from mechanistic models that encode a great deal of information into the generative process to empirical ones that derive information mainly from observed data.

population, and $\boldsymbol{\beta}$ are latent parameters, this model is

$$\frac{dx_1}{dt} = \beta_1 x_1 - \beta_2 x_1 x_2 + \epsilon_1, \qquad \epsilon_1 \sim \text{Normal}(0, 10),$$
$$\frac{dx_2}{dt} = -\beta_2 x_2 + \beta_3 x_1 x_2 + \epsilon_2, \quad \epsilon_2 \sim \text{Normal}(0, 10),$$

where Gaussian noises ϵ_1, ϵ_2 are added at each full time step. The differential equations encode that more prey yield more predators, and more predators yield less prey. The computational task to work with this model is to compute the posterior distribution of $\boldsymbol{\beta}$.[2]

The Lotka-Volterra model, like many other physical systems, is a dynamical system. Other examples include models of bloodstream glucose (Sedigh-Sarvestani et al., 2012) and simulators of the world (Feynman, 1982) measured with noisy observations.

2.3.4 Model Spectrum

We have discussed an empirical model, the sigmoid belief network, and a mechanistic model, the Lotka-Volterra predator-prey model. In terms of where they get their information, these models represent the extremes. Empirical models mostly rely on the data, while mechanistic models rely more on prior knowledge baked into the structure. All models lie

[2] For brevity, we ignore the task of specifying the prior on $\boldsymbol{\beta}$.

on a spectrum from mechanistic models to empirical models. The most extreme empirical models are density estimators and the most extreme mechanistic models are noise-free physical systems that detail the full mechanism for data generation.

Measurement models are models with prescribed hidden structure, such as grouping in the mixture model or topics in a topic model (Blei et al., 2003). They fall somewhere in the middle of this spectrum. While not mechanistic, these models impose prior constraints, which are a type of structure. However, this structure is meaningless without data to give it context.

Figure 2.3 displays this spectrum with common models on it. To work with these models, we need the posterior distribution. This is the main computational challenge.

2.4 Posterior Distribution

The posterior distribution $p(\boldsymbol{z} \mid \boldsymbol{x})$ is the distribution of the hidden structure $\boldsymbol{z}$ given the data $\boldsymbol{x}$ — it is the central quantity in Bayesian inference. Generative probabilistic models specify the joint distribution of hidden structure and observed data $p(\boldsymbol{x}, \boldsymbol{z})$. Given the joint distribution, we can write the posterior as

$$p(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}, \boldsymbol{z})}{\int p(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}}. \tag{2.2}$$

The challenge in computing the posterior lies in the intractability of the integral for the marginal likelihood $p(\boldsymbol{x})$. For example, in the sigmoid belief network with L layers and K variables in each layer, this integral corresponds to a sum over 2^{L*K} terms. Computing this integral is infeasible for all but the smallest models. The intractability of the posterior in the sigmoid belief network is not special. The majority of the models of interest have intractable posteriors.

Many techniques can be used to compute the posterior distribution. We present an overview of several, including exact computation, importance sampling, Markov chain Monte Carlo, variational inference, and stochastic variational inference.

2.5 Exact Computation

Exact computation uses numerical integration (Davis and Rabinowitz, 2007) to compute the posterior Equation (2.2). For discrete variables this integration is a summation, like in the sigmoid belief network. The difficulty of computing this integral can be exponential in the dimension of the latent space, so exact computation only works in problems with low dimensional latent spaces.

2.6 Importance Sampling

Importance sampling (Owen, 2013) is a key tool for computing expectations. The main idea behind importance sampling is that samples from one distribution can be used to compute expectations with respect to another. Let r, q be distributions and f be a function, then importance sampling can be summarized by

$$\mathbb{E}_r f(z) = \int f(z) r(z) dz = \int \frac{f(z) r(z)}{q(z)} q(z) dz = \mathbb{E}_q \left[\frac{f(z) r(z)}{q(z)} \right]. \tag{2.3}$$

This expectation can be estimated using Monte Carlo, Draw S samples from q and average

$$\mathbb{E}_r f(z) \approx \frac{1}{S} \sum_{i=1}^{S} \frac{f(z_i) r(z_i)}{q(z_i)} \qquad z_i \sim q. \tag{2.4}$$

Importance sampling from Equation (2.3) requires knowing r, but the posterior is only known up to a constant $p(\boldsymbol{x})$. Importance sampling for the posterior requires self-normalized importance sampling.

Self-normalized importance sampling works as follows. First take a distribution $q(\boldsymbol{z})$ known up to a multiplicative constant and which has a known simulation procedure. Next, to estimate the expectation of a function $f(\boldsymbol{z})$ with respect to the posterior distribution, draw S samples $z_{1:S}$ from q, define $w_i = \frac{p(\boldsymbol{x}_i, \boldsymbol{z}_i)}{q(\boldsymbol{z}_i)}$, then compute

$$\mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x})} f(\boldsymbol{z}) \approx \frac{\sum_{i=1}^{S} f(\boldsymbol{z}_i) w_i}{\sum_{i=1}^{S} w_i}.$$

Note that the joint distribution is the posterior distribution up to a constant. The unknown normalization constants for the posterior and q cancel out in the numerator and denominator, thus making the self-normalized estimate correct.

At face value, importance sampling works; there is nothing that is computationally intractable. The challenge lies in the variance of the self-normalized importance sampling estimate. The variance of the estimator depends on the divergence of q_i, i.e., how close q is to $p(\boldsymbol{z} \mid \boldsymbol{x})$. This becomes very problematic in high dimensions (Tokdar and Kass, 2010).

2.7 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods are a staple in computing with generative probabilistic models. The main building block for MCMC algorithms is a transition probability kernel $\mathcal{T}$ that stochastically maps a position in latent space $\boldsymbol{z}_t$ to a new position $\boldsymbol{z}_{t+1}$. Applying $\mathcal{T}$ repeatedly defines a Markov chain $(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots)$. The stationary distribution π of a

Markov distribution is one where

$$\int \mathcal{T}(z_t, z_{t-1})\pi(z_{t-1})dz_{t-1} = \pi(z_t).$$

Put in words, applying the transition probability kernel does not change the probability distribution from the stationary distribution. For a Markov chain to generate samples from the posterior, the posterior needs to be the Markov chain's stationary distribution. In addition, the Markov chain must converge to the stationary distribution regardless of initialization (ergodic) to guarantee that running the chain generates samples from the posterior. There are many ways to design transition probability kernels that meet these criteria, and developing new MCMC methods remains an active area of research. We detail a few below.

2.7.1 Metropolis-Hastings

The earliest example of a MCMC algorithm is the random walk Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970). The Metropolis-Hastings transition kernel requires a proposal distribution q, say the standard Gaussian. Let r be the target; the distribution we want to sample. A Metropolis-Hastings transition produces samples as

$$\begin{aligned} z^* &\sim q(z^* \mid z_t) \\ a &= \min\left(1, \frac{r(z^*)q(z_t \mid z^*)}{r(z_t)q(z_t^* \mid z)}\right) \\ c &\sim \text{Bernoulli}(a) \\ z_{t+1} &= z^* c + z_t(1-c). \end{aligned}$$

The existence of a stationary distribution can be verified by checking the detailed balance condition (Bishop, 2006). The density r need only be known up to a constant. The constant cancels out in the computation of the acceptance ratio a. This means Metropolis-Hastings

can be used for posterior inference. The challenge with Metropolis-Hastings relates to the challenge in importance sampling. Intuitively, the speed at which Metropolis-Hastings mixes depends on the acceptance ratio. The acceptance ratio is high when the likelihood ratio is high (q is close to r). This is a challenge to achieve in high dimensions.

2.7.2 Gibbs Sampling

Metropolis-Hastings can be slow in high dimensions. An alternative way to build an MCMC transition probability kernel is to compose simpler ones. Consider splitting the latent space $\boldsymbol{z}$ into two parts $\boldsymbol{z}^1$ and $\boldsymbol{z}^2$. Next, define two transition probability kernels, one for $\boldsymbol{z}^1$ given $\boldsymbol{z}^2$ and one for $\boldsymbol{z}^2$ given $\boldsymbol{z}^1$, by their conditional distributions in π.

Each of these transitions preserve stationarity:

$$\begin{aligned}
\int \mathcal{T}(\boldsymbol{z}_t, \boldsymbol{z}_{t-1})\pi(\boldsymbol{z}_{t-1})d\boldsymbol{z}_{t-1} &= \int \pi(\boldsymbol{z}_t^2 \mid \boldsymbol{z}_{t-1}^1)\delta_{\boldsymbol{z}_{t-1}^1}(\boldsymbol{z}_t^1)\pi(\boldsymbol{z}_{t-1}^1, \boldsymbol{z}_{t-1}^2)d\boldsymbol{z}_{t-1}^2 d\boldsymbol{z}_{t-1}^1 \\
&= \int \pi(\boldsymbol{z}_t^2 \mid \boldsymbol{z}_{t-1}^1)\delta_{\boldsymbol{z}_{t-1}^1}(\boldsymbol{z}_t^1) \int \pi(\boldsymbol{z}_{t-1}^1, \boldsymbol{z}_{t-1}^2)d\boldsymbol{z}_{t-1}^2 d\boldsymbol{z}_{t-1}^1 \\
&= \int \pi(\boldsymbol{z}_t^2 \mid \boldsymbol{z}_{t-1}^1)\delta_{\boldsymbol{z}_{t-1}^1}(\boldsymbol{z}_t^1)\pi(\boldsymbol{z}_{t-1}^1)d\boldsymbol{z}_{t-1}^1 \\
&= \pi(\boldsymbol{z}_t^2 \mid \boldsymbol{z}_t^1)\pi(\boldsymbol{z}_t^1) = \pi(\boldsymbol{z}_t^1, \boldsymbol{z}_t^2).
\end{aligned}$$

This derivation shows that sampling from the conditional distributions preserves stationarity. Alternating between two variable sets gives a posterior sampling algorithm called the Gibbs sampler (Geman and Geman, 1984). For efficient sampling, the Gibbs sampler requires easy-to-sample *complete conditionals*—the distribution of one latent variable $\boldsymbol{z}_1$ given the rest of the latent variables $\boldsymbol{z}_2$ and data $\boldsymbol{x}$. Without easy-to-sample conditionals, the Metropolis-Hastings algorithm can be used to sample each conditional at a cost of lower acceptance rates. Finally, while mitigating the acceptance ratio problem in Metropolis-

Hastings, the Gibbs sampler can be prone to getting stuck when variables with high correlation fall into different groups.

2.7.3 Hamiltonian Monte Carlo

The proposal in the traditional Metropolis-Hastings algorithm does not take into account information from the unnormalized density. One possible kind of information is the gradient of the logarithm of the unnormalized density. We now discuss the intuitions behind an MCMC algorithm that makes use of gradient information: Hamiltonian Monte Carlo (Duane et al., 1987). Hamiltonian Monte Carlo defines a proposal by simulating a physical system. Tie the energy of a physical system to the logarithm of the unnormalized target density r. Then by conservation of energy, the proposal created by simulating the physical system will be accepted with probability one (the energies, thus the probabilities are the same). In Hamiltonian Monte Carlo, the potential energy of the system is set to be $U = -\log r(\boldsymbol{z})$. In the simplest case, the kinetic energy of the system can be defined via an auxiliary variable $\boldsymbol{m}$ with form

$$K = \frac{1}{2}\boldsymbol{m}^\top \boldsymbol{m},$$

meaning that $\boldsymbol{m}$ is a standard Gaussian. Together, these define the Hamiltonian $H = U + K$. Next define a distribution $h \propto \exp(-H)$. The marginal density of h on $\boldsymbol{z}$ is the target r. This means sampling the joint space of $(\boldsymbol{z}, \boldsymbol{m})$ according to h, then tossing out the dimension corresponding to $\boldsymbol{m}$ produces samples from r. To generate proposals, Hamiltonian Monte Carlo simulates the laws of motion, by following a discretized form of Hamilton's equations using gradients, from an initial $(\boldsymbol{z}, \boldsymbol{m})$ pair. This produces another $(\boldsymbol{z}^*, \boldsymbol{m}^*)$ pair with the same Hamiltonian. This is a proposal will be accepted with probability one. To ensure sampling of all possible states, the auxiliary variable should be resampled from the standard

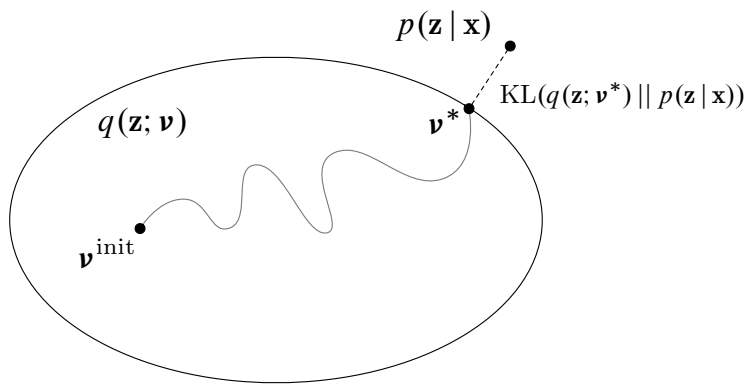


Figure 2.4: A pictorial description of variational inference. The family of variational approximations is the oval, $\boldsymbol{\nu}^{\text{init}}$ denotes the initial approximation, and $\boldsymbol{\nu}^*$ the approximation after running variational inference. The posterior lives outside the approximating family. The distance between distributions in the picture maps to the KL divergence.

Gaussian. Finally, simulating discretized Hamilton's equations incurs error, so the energy may not be exactly conserved. This means an accept-reject step is still required. Hamiltonian Monte Carlo can be far more efficient than the classic Metropolis-Hastings approach as it can use gradients to quickly find regions of high probability.

2.8 Variational Inference

Variational inference takes a different approach than the previous posterior approximation algorithms. The previous algorithms, like the Gibbs sampler, rely on stochastic processes that yield samples from the posterior.[3] Instead, variational inference transforms posterior inference into optimization (Jordan, 1999; Bishop, 2006; Wainwright and Jordan, 2008).

[3]Importance sampling can be viewed as a type of posterior sampling procedure when resampling from the weighted particle set.

Variational inference posits a class of distributions q over the latent space with parameters $\boldsymbol{\nu}$ and tries to find the closest distribution in KL divergence to the posterior. Figure 2.4 depicts this. The KL divergence arises without explicitly being instantiated in many statistical problems. For example, maximum likelihood minimizes the KL divergence from the empirical distribution on the observed data to the model, and posteriors concentrate to the point closest in KL divergence to the true data generating distribution (Kleijn and van der Vaart, 2006). We will use semicolons to denote parameters of probability distributions; these parameters are not random variables.

The direct KL divergence optimization problem requires the posterior distribution:

$$\mathrm{KL}(q(\boldsymbol{z};\boldsymbol{\nu})||p(\boldsymbol{z}\mid\boldsymbol{x})) = \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\nu})}\left[\log\frac{q(\boldsymbol{z};\boldsymbol{\nu})}{p(\boldsymbol{z}\mid\boldsymbol{x})}\right].$$

We can make this objective tractable by pulling out $\log p(\boldsymbol{x})$

$$\begin{aligned}\mathrm{KL}(q(\boldsymbol{z};\boldsymbol{\nu})||p(\boldsymbol{z}\mid\boldsymbol{x})) &= \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\nu})}\left[\log\frac{q(\boldsymbol{z};\boldsymbol{\nu})}{p(\boldsymbol{z}\mid\boldsymbol{x})}\right]\\ &= \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\nu})}\left[\log\frac{q(\boldsymbol{z};\boldsymbol{\nu})p(\boldsymbol{x})}{p(\boldsymbol{x},\boldsymbol{z})}\right]\\ &= \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\nu})}\left[\log\frac{q(\boldsymbol{z};\boldsymbol{\nu})}{p(\boldsymbol{x},\boldsymbol{z})}\right] + \log p(\boldsymbol{x}).\end{aligned} \tag{2.5}$$

This means we can minimize the KL up to a constant.

Minimizing the KL divergence is equivalent to maximizing the evidence lower bound (ELBO) denoted by $\mathcal{L}$,

$$\mathcal{L}(\boldsymbol{\nu}) = \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\nu})}\left[\log p(\boldsymbol{x},\boldsymbol{z}) - \log q(\boldsymbol{z};\boldsymbol{\nu})\right]. \tag{2.6}$$

The ELBO gets its name from being a lower bound on $\log p(\boldsymbol{x})$, the "evidence." This property follows from Equation (2.5) and the non-negativity of the Kullback-Leibler divergence.

The choice of variational family $q(\boldsymbol{z}; \boldsymbol{\nu})$ trades off the fidelity of the posterior approximation with the difficulty of optimizing over the variational parameters. The classic choice of variational family is the mean field family, which factorizes across groups of hidden variables. Assume that there are d groups. T he mean field family is

$$q(\boldsymbol{z}; \boldsymbol{\nu}) = \prod_{i=1}^{d} q(\boldsymbol{z}_i; \boldsymbol{\nu}_i).$$

We can compute the optimal value of the mean field distribution for $\boldsymbol{z}_i$ while keeping the rest, $\boldsymbol{z}_{-i}$, fixed. This gives

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\nu}_i) &= \mathbb{E}_{q(\boldsymbol{z})}\left[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}; \boldsymbol{\nu})\right] \\
&= \mathbb{E}_{q(\boldsymbol{z}_i)}\left[\mathbb{E}_{q(\boldsymbol{z}_{-i})}[\log p(\boldsymbol{x}, \boldsymbol{z})] - \log q(\boldsymbol{z}_i; \boldsymbol{\nu}_i)\right] + C \\
&= -\mathrm{KL}\left(q(\boldsymbol{z}_i; \boldsymbol{\nu}_i) || \frac{1}{Z} \exp(\mathbb{E}_{q(\boldsymbol{z}_{-i})}[\log p(\boldsymbol{x}, \boldsymbol{z})])\right) + C,
\end{aligned}$$

where C is an arbitrary constant and Z is introduced to normalize $\exp(\mathbb{E}_{q(\boldsymbol{z}_{-i})}[\log p(\boldsymbol{x}, \boldsymbol{z})])$ to make it a proper distribution. This equation implies that the optimal variational family for $\boldsymbol{z}_i$ keeping the rest $\boldsymbol{z}_{-i}$ fixed minimizes the KL divergence to $\exp(\mathbb{E}_{q(\boldsymbol{z}_{-i})}[\log p(\boldsymbol{x}, \boldsymbol{z})])$. Thus the optimal variational approximation across all distributions is $\frac{1}{Z}\exp(\mathbb{E}_{q(\boldsymbol{z}_{-i})}[\log p(\boldsymbol{x}, \boldsymbol{z})])$.

In general, the optimization problem Equation (2.6) is non-concave. This means that optimization may not converge to the distribution with minimum KL divergence to the posterior. Rather, it will converge to a local optimum.

2.9 Stochastic Variational Inference

Dealing with data sets is an ever-growing challenge in modern data analysis. Stochastic variational inference (SVI) scales variational inference to large datasets. SVI works over a subset

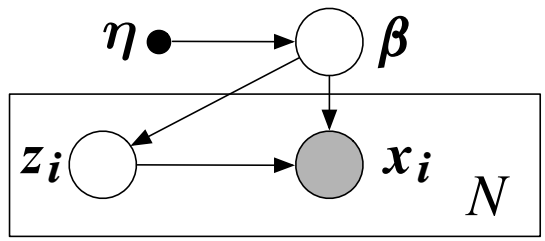


Figure 2.5: The graphical model for hierarchical Bayesian models. The latent variable $\boldsymbol{\beta}$ is shared across data $\boldsymbol{x}_i$ and data-specific latent variables $\boldsymbol{z}_i$. The local variables, data, and data-specific latent variables are independent given the global variable.

of models that have *conditional conjugacy*. We describe this class of models, then derive the traditional coordinate ascent algorithm (Ghahramani and Beal, 2001) and SVI.

2.9.1 Model Class

We depict the class of models SVI supports in Figure 2.5. There are three types of variables: the data are $\boldsymbol{x}_{1:n}$, the latent variables associated with each observation are $\boldsymbol{z}_{1:n}$, and the latent variables shared across data are $\boldsymbol{\beta}$. The model assumes that given the global hidden structure $\boldsymbol{\beta}$, the observations and their corresponding local hidden variables are conditionally independent. Let $\boldsymbol{\eta}$ be a hyperparameter. The model factorizes as

$$p(\boldsymbol{\beta}, \boldsymbol{x}_{1:n}, \boldsymbol{z}_{1:n}; \boldsymbol{\eta}) = p(\boldsymbol{\beta}; \boldsymbol{\eta}) \prod_{i=1}^{n} p(\boldsymbol{z}_i, \boldsymbol{x}_i \mid \boldsymbol{\beta}).$$

This expression describes a hierarchical model. In SVI, each distribution in this factorization is chosen to be in the exponential family (Brown, 1986),

$$p(\boldsymbol{z}_i, \boldsymbol{x}_i \mid \boldsymbol{\beta}) = h(\boldsymbol{z}_i, \boldsymbol{x}_i) \exp(\boldsymbol{\beta}^\top t(\boldsymbol{z}_i, \boldsymbol{x}_i) - a(\boldsymbol{\beta}))$$
$$p(\boldsymbol{\beta}; \boldsymbol{\eta}) = h(\boldsymbol{\beta}) \exp(\boldsymbol{\eta}^\top t(\boldsymbol{\beta}) - a(\boldsymbol{\eta})),$$

where h is the base measure, t the sufficient statistics, and a the log-normalizers.

The exponential families are chosen such that the model satisfies *conditional conjugacy*—prior $p(\boldsymbol{\beta}; \boldsymbol{\eta})$ is conjugate to $p(\boldsymbol{z}_i, \boldsymbol{x}_i \mid \boldsymbol{\beta})$. Conditional conjugacy means the distribution of the complete conditional $p(\boldsymbol{\beta} \mid \boldsymbol{z}_{1:n}, \boldsymbol{x}_{1:n})$ is in the same family as the prior $p(\boldsymbol{\beta}; \boldsymbol{\eta})$. Unlike traditional Bayesian conjugacy, conditional conjugacy requires the presence of the hidden variables $\boldsymbol{z}$. In this model class, the conditional distribution given only the observations need not be in the same family as the prior.

Though this model class imposes many restrictions, it includes many popular models such as Bayesian Gaussian mixtures, latent Dirichlet allocation (Blei et al., 2003), probabilistic matrix factorization (Salakhutdinov and Mnih, 2008), hierachical linear regression (Gelman and Hill, 2007), and several hierarchical Bayesian nonparametric models (Hjort et al., 2010).

2.9.2 Approximating Family

The evidence lower bound (ELBO) for the SVI model family is

$$\mathcal{L} = \mathbb{E}_q[\log p(\boldsymbol{x}_{1:n}, \boldsymbol{z}_{1:n}, \boldsymbol{\beta}) - \log q(\boldsymbol{\beta}, \boldsymbol{z}_{1:n}; \boldsymbol{\nu})].$$

SVI specifies the variational distribution to be the mean field family. Let $\boldsymbol{\lambda}$ be the variational parameters for $\boldsymbol{\beta}$, and $\boldsymbol{\phi}_i$ be the parameters for $\boldsymbol{z}_i$, then

$$q(\boldsymbol{\beta}, \boldsymbol{z}_{1:n}; \boldsymbol{\lambda}, \boldsymbol{\phi}_{1:n}) = q(\boldsymbol{\beta}; \boldsymbol{\lambda}) \prod_{i=1}^{n} q(\boldsymbol{z}_i; \boldsymbol{\phi}_i).$$

SVI assumes that each variational distribution comes from the same family as the conditional. Using the mean field family, the ELBO is

$$\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\phi}_{1:n}) = \mathbb{E}_q \left[\log p(\boldsymbol{\beta}; \boldsymbol{\eta}) - \log q(\boldsymbol{\beta}; \boldsymbol{\lambda}) + \sum_{i=1}^{n} \log p(\boldsymbol{x}_i, \boldsymbol{z}_i \mid \boldsymbol{\beta}) - \log q(\boldsymbol{z}_i; \boldsymbol{\phi}_i) \right]. \tag{2.7}$$

We review the coordinate ascent variational inference algorithm for this model before covering stochastic variational inference.

2.9.3 Coordinate Ascent Variational Inference

Coordinate ascent variational inference has been the workhorse for deploying variational inference in a variety of applications (Blei et al., 2003; Nallapati et al., 2008; Airoldi et al., 2008; Sudderth and Jordan, 2009).

SVI writes the ELBO as only a function of the global variational approximation by setting each of the $\boldsymbol{\phi}$ to their optimal value given a fixed value of $\boldsymbol{\lambda}$,

$$\mathcal{L}(\boldsymbol{\lambda}) = \max_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\phi}_{1:n}).$$

This form is called the $\boldsymbol{\lambda}$-ELBO.

Define the optimal local variational parameters $\boldsymbol{\phi}(\boldsymbol{\lambda}) = \arg\max_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\phi})$. By Equation (2.7), the optimal $\boldsymbol{\phi}_i$ can be computed separately for each data point. This computation has a closed form solution due to the conditional conjugacy assumption. Let $\boldsymbol{\zeta}_i$ be the parameter of the complete conditional of $\boldsymbol{z}_i$. The gradient is

$$\nabla_{\boldsymbol{\phi}_i} \mathcal{L} = \nabla^2_{\boldsymbol{\phi}_i} a(\boldsymbol{\phi}_i)(\mathbb{E}_{q(\boldsymbol{\beta};\boldsymbol{\lambda})}[\boldsymbol{\zeta}_i] - \boldsymbol{\phi}_i).$$

We derive this in the appendix of this chapter. Setting the gradient to zero gives the closed form update

$$\boldsymbol{\phi}_i = \mathbb{E}_{q(\boldsymbol{\beta};\boldsymbol{\lambda})}[\boldsymbol{\zeta}_i].$$

In words, the optimal variational parameter equals the expected value of the complete conditional's natural parameter.

Returning back to the $\boldsymbol{\lambda}$-ELBO, by a similar argument

$$\mathcal{L}(\boldsymbol{\lambda}) = a(\boldsymbol{\lambda}) + \nabla_{\boldsymbol{\lambda}} a(\boldsymbol{\lambda})^\top \left(-\boldsymbol{\lambda} + \boldsymbol{\eta} + \sum_{i=1}^{n} \mathbb{E}_{q(\boldsymbol{z}_i;\boldsymbol{\phi}_i(\boldsymbol{\lambda}))}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] \right) + C.$$

Differentiating this with respect to $\boldsymbol{\lambda}$ yields

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) = \nabla^2_{\boldsymbol{\lambda}} a(\boldsymbol{\lambda}) \left(-\boldsymbol{\lambda} + \boldsymbol{\eta} + \sum_{i=1}^{n} \mathbb{E}_{q(\boldsymbol{z}_i;\boldsymbol{\phi}_i(\boldsymbol{\lambda}))}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] \right). \tag{2.8}$$

The chain rule term for $\boldsymbol{\phi}_i(\boldsymbol{\lambda})$ disappears since $\boldsymbol{\phi}_i(\boldsymbol{\lambda})$ is an optimum.

Setting the gradient equal to zero gives a fixed-point update for the global variational parameters. Let $\boldsymbol{\lambda}_t$ be the current iterate of the global variational parameters. Define

$$\boldsymbol{\lambda}_t^* = \boldsymbol{\eta} + \sum_{i=1}^{n} \mathbb{E}_{q(\boldsymbol{z}_i;\boldsymbol{\phi}_i(\boldsymbol{\lambda}))}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)]. \tag{2.9}$$

This equation is the optimal value for $\boldsymbol{\lambda}$ given the current value of $\boldsymbol{\lambda}_t$. Coordinate ascent variational inference alternates between computing the optimal local variational parameters given the global variational parameters and the optimal global parameters given the local ones. Iteration of this fixed-point update increases the ELBO. Coordinate ascent for hierarchical Bayesian models can be generalized beyond conditionally conjugate hierarchical models to all conditionally conjugate models.

2.9.4 Stochastic Variational Inference

Hoffman et al. (2013) point out that computing the optimal variational parameter in Equation (2.9) requires computing a function of the current global variational parameters for each data point. This is inefficient; particularly so at initialization, when the global variational parameters are random. They propose a solution: use stochastic optimization.

Stochastic optimization. Stochastic optimization (Robbins and Monro, 1951) maximizes a function by following noisy, yet unbiased gradients. Let f be a function to be maximized, H be a random variable such that $\mathbb{E}H(a) = \nabla_a f(a)$, and $h(a)$ be a draw from $H(a)$. Finally, let ρ_t be nonnegative. Then stochastic optimization updates the current parameter a_t as

$$a_{t+1} = a_t + \rho_t h(a_t).$$

This converges to a maximum of f when the ρ_t satisfy the Robbins-Monro conditions,

$$\sum_{i=1}^{\infty} \rho_t = \infty, \qquad \sum_{i=1}^{\infty} \rho_t^2 < \infty.$$

The first condition is needed to ensure convergence from any starting position, while the second is needed to control the variance caused by using noisy gradients.

Stochastic variational inference. The gradient of $\boldsymbol{\lambda}$ contains a sum over the data points. One can construct a noisy unbiased variant of the gradient by sampling a single data point

and scaling by the number of data points,

$$
\begin{aligned}
i &\sim \text{Unif}(1 \ldots n) \\
\hat{\nabla}_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) &= \nabla^2_{\boldsymbol{\lambda}} a(\boldsymbol{\lambda}) \left(-\boldsymbol{\lambda} + \boldsymbol{\eta} + n \mathbb{E}_{q(\boldsymbol{z}_i; \boldsymbol{\phi}_i(\boldsymbol{\lambda}))}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] \right).
\end{aligned}
$$

The expected value of $\hat{\nabla}_{\boldsymbol{\lambda}}$ with respect to the uniform sample is the gradient of the ELBO. Thus, it is a noisy and unbiased gradient suitable for use in stochastic optimization.

There remains a problem. The gradient in Equation (2.8) and thus the stochastic gradient requires multiplication with a matrix $\nabla^2_{\boldsymbol{\lambda}} a(\boldsymbol{\lambda})$ that has the same dimension as the number of global variational parameters.

Hoffman et al. (2013) step around this by scaling the gradient by the inverse of $\nabla^2_{\boldsymbol{\lambda}} a(\boldsymbol{\lambda})$, which is also known as the Fisher information matrix. By properties of the exponential family, the matrix $\nabla^2_{\boldsymbol{\lambda}} a(\boldsymbol{\lambda})$ is the covariance matrix of the sufficient statistics $t(\boldsymbol{\beta})$. Thus, this rescaling does not change the optima of the problem as the Fisher information matrix is positive definite (Sunehag et al., 2009). This rescaled gradient is the natural gradient (Amari, 1998). Natural gradients accelerate gradient methods over probability distributions by accounting for their non-Euclidean nature.

Putting this all together gives stochastic variational inference. At iteration t:

1. Sample a data point $i \sim \text{Unif}(1 \ldots n)$

2. Compute the optimal local parameter $\boldsymbol{\phi}_i(\boldsymbol{\lambda}_t)$ given the current global $\boldsymbol{\lambda}_t$

3. Compute intermediate global parameters as the coordinate update of $\boldsymbol{\lambda}$ for data point i:

$$
\hat{\boldsymbol{\lambda}}_t = \eta + n \mathbb{E}_{q(\boldsymbol{z}_i; \boldsymbol{\phi}_i(\boldsymbol{\lambda}))}[t(\boldsymbol{x}_i, \boldsymbol{z}_i)] \tag{2.10}
$$

4. Following the noisy gradient

$$\boldsymbol{\lambda}_{t+1} = (1 - \rho_t)\boldsymbol{\lambda}_t + \rho_t \hat{\boldsymbol{\lambda}}_t. \tag{2.11}$$

This process is much more efficient than coordinate ascent inference. Rather than analyzing the whole data set before updating the global parameters, SVI needs only analyze a single sampled data point. Stochastic variational inference has scaled posterior inference to larger datasets, but remains tied to conditioned conjugate models.

2.10 Conclusion

We laid out desiderata for models of data, showed that generative probabilistic models satisfy the desiderata, and described example generative probabilistic models. Working with probabilistic generative models requires computing the posterior distribution. However, as models grow in complexity and data grow in size, computing the posterior distribution quickly becomes intractable. The subsequent chapters wrestle with this question with a focus on posterior approximation methods that are easy-to-use for non machine learning/statistics experts—scientists with data who build models to understand their data. The next chapter develops a variant of variational inference that requires minimal work. This variant opens the door to new model classes, better variational approximations, and new notions of distance we study the subsequent chapters.

2.11 Appendix

2.11.1 Gradient of the ELBO with respect to $\boldsymbol{\phi}_i$

We differentiate the ELBO and use properties of the exponential family:

$$
\begin{aligned}
\nabla_{\boldsymbol{\phi}_i} \mathcal{L} &= \nabla_{\boldsymbol{\phi}_i} \mathbb{E}_q[\log p(\boldsymbol{x}_i, \boldsymbol{z}_i \mid \boldsymbol{\beta}) - \log q(\boldsymbol{z}_i; \boldsymbol{\phi}_i)] \\
&= \nabla_{\boldsymbol{\phi}_i} \mathbb{E}_q[\log p(\boldsymbol{z}_i \mid \boldsymbol{x}_i, \boldsymbol{\beta}) - \log q(\boldsymbol{z}_i; \boldsymbol{\phi}_i)] \\
&= \nabla_{\boldsymbol{\phi}_i} \mathbb{E}_q[\log h(\boldsymbol{z}_i) + \boldsymbol{\zeta}_i^\top t(\boldsymbol{z}_i) - a(\boldsymbol{\zeta}_i) - \log h(\boldsymbol{z}_i) - \boldsymbol{\phi}_i^\top t(\boldsymbol{z}_i) + a(\boldsymbol{\phi}_i)] \\
&= \nabla_{\boldsymbol{\phi}_i} \mathbb{E}_{q(\boldsymbol{z}_i;\boldsymbol{\phi}_i)} \left[\mathbb{E}_{q(\boldsymbol{\beta};\boldsymbol{\lambda})}[\boldsymbol{\zeta}_i]^\top t(\boldsymbol{z}_i) - a(\boldsymbol{\zeta}_i) - \boldsymbol{\phi}_i^\top t(\boldsymbol{z}_i) + a(\boldsymbol{\phi}_i)\right].
\end{aligned}
$$

Using the fact that the expected value of the sufficient statistics is the gradient of the log-normalizer ($\mathbb{E}[t(\boldsymbol{z}_i)] = \nabla_{\boldsymbol{\phi}_i} a(\boldsymbol{\phi}_i)$), we get

$$
\begin{aligned}
\nabla_{\boldsymbol{\phi}_i} \mathcal{L} &= \nabla_{\boldsymbol{\phi}_i} \left[(\mathbb{E}_{q(\boldsymbol{\beta};\boldsymbol{\lambda})}[\boldsymbol{\zeta}_i] - \boldsymbol{\phi}_i)^\top \nabla_{\boldsymbol{\phi}_i} a(\boldsymbol{\phi}_i)) - a(\boldsymbol{\zeta}_i) + a(\boldsymbol{\phi}_i)\right] \\
&= \nabla^2_{\boldsymbol{\phi}_i} a(\boldsymbol{\phi}_i)(\mathbb{E}_{q(\boldsymbol{\beta};\boldsymbol{\lambda})}[\boldsymbol{\zeta}_i] - \boldsymbol{\phi}_i).
\end{aligned}
$$

This is the gradient used in the main text.

Chapter 3

Black Box Variational Inference

In the previous chapter, we reviewed both coordinate ascent variational inference and stochastic variational inference (SVI). SVI provides a recipe for a broad class of conditionally conjugate models, yet the use of variational inference in general models remains difficult. The models where these algorithms apply are those where the conditionals have convenient form (conditionally conjugate) and the variational approximations have been matched appropriately to the conditionals. Outside of this setting, practitioners have resorted to model-specific algorithms (Jaakkola and Jordan, 1997; Blei and Lafferty, 2006a; Braun and McAuliffe, 2010) or generic algorithms that require model-specific computations (Knowles and Minka, 2011; Wang and Blei, 2013; Paisley et al., 2012).

Deriving variational inference for non-conjugate models on a model-by-model basis is tedious work. Rather than choosing models based on the needs of the data, models get chosen based on the ease of deriving inference. The need to derive inference creates sunk costs that hinder the rapid exploration of models. Moreover, the model-specific analysis required for variational inference may be impractical for users of generative probabilistic models in the sciences and social sciences.

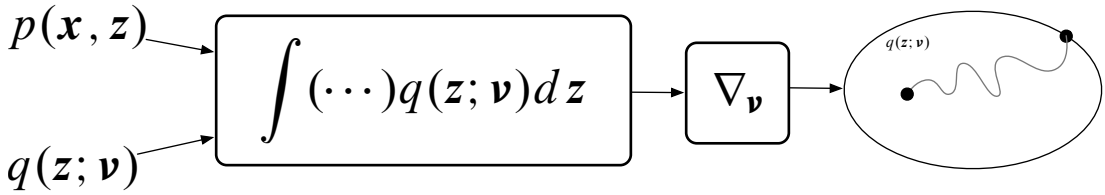


Figure 3.1: The recipe for variational inference. First, choose an approximating family. Next, compute the ELBO by integration. Then, differentiate the ELBO. Maximize the ELBO either by following the derivatives or setting them to zero for coordinate ascent.

In this chapter, we develop a *black box* variational inference algorithm. BBVI is a variational inference method that can quickly be applied to almost any model and approximating family with minimal effort. BBVI allows practitioners to design, apply, and revise models of their data, without painstaking derivations each time they adjust the model. This moves the generative probabilistic modeling process more towards building models for data rather than models for inference.

3.1 A Case Study: Bayesian Logistic Regression

In the previous chapter, we showed how coordinate ascent variational inference and SVI give methods for variational inference in conditionally conjugate models. Both methods were derived via a recipe: 1) compute the expectations in the ELBO; 2) differentiate the ELBO with respect to its parameters; 3) use the derivative to maximize the ELBO. Figure 3.1 summarizes this recipe. Here we explore what happens for a simple non-conjugate model, Bayesian logistic regression (Cox, 1958).

Bayesian logistic regression is a prediction model for binary outcomes. The data come in covariate, binary outcome $(\boldsymbol{x}_i, y_i)$ pairs. The model conditions on the covariates $\boldsymbol{x}_i$ to

generate the response

$$y_i \sim \text{Bernoulli}(\sigma(\boldsymbol{x}_i^\top \boldsymbol{w})),$$

and the prior on the regression coefficients is normal:

$$\boldsymbol{w} \sim \mathcal{N}(0, 1).$$

For simplicity, we assume we have only a single data point (x, y), and that the covariates are scalars. In this case, the commonly used variational approximation is the Normal distribution with parameters μ, σ^2:

$$q(w; \mu, \sigma^2) = \mathcal{N}(\mu, \sigma^2).$$

We now follow the recipe. We write down the ELBO and compute expectations with respect to the variational approximations:

$$\begin{aligned}
\mathcal{L}(\mu, \sigma^2) &= \mathbb{E}_q[\log p(w) - \log q(w; \mu, \sigma^2) + \log p(y \mid x, w)] \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log(\sigma^2) + \mathbb{E}_q[\log p(y \mid x, w)] + C \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log(\sigma^2) + \mathbb{E}_q[yxw - \log(1 + \exp(xw))] + C \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log(\sigma^2) + yx\mu - \mathbb{E}_q[\log(1 + \exp(xw))] + C.
\end{aligned}$$

At this point, we can proceed no further in following the recipe. The expected value of the last term does not have a nice closed form in terms of the variational parameters. Possible solutions include further model-specific lower bounds (Jaakkola and Jordan, 1997) or more general techniques that require model-specific analysis (Knowles and Minka, 2011; Wang and Blei, 2013). The model-specific nature of these approaches coupled with the additional approximations limits their applicability. Instead, we need an approach that does not require

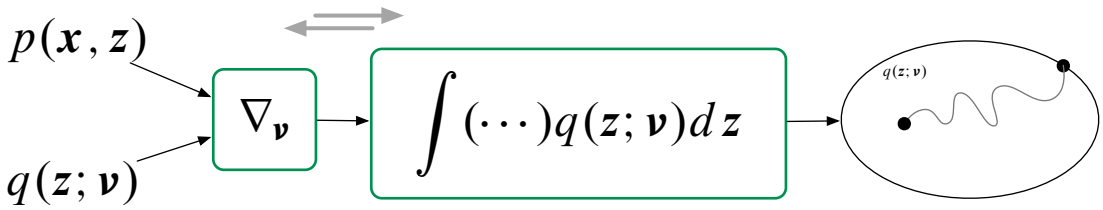


Figure 3.2: The new recipe for variational inference. First, choose an approximating family. Next, write the derivative of the ELBO as an expectation. Then, use Monte Carlo with samples from q to compute noisy gradients. This approach avoids intractable integrals. Finally, use stochastic optimization to maximize the ELBO.

additional approximations nor manual computation of expectations. Black box variational inference provides such an approach.

3.2 Black Box Variational Inference

Black box variational inference (BBVI) is a new recipe for variational inference. The problem in the old recipe was the that expectations of certain functions were analytically intractable. We step around this problem by writing the derivative as an expectation and using Monte Carlo estimates of this derivative to get noisy unbiased gradients. Figure 3.2 summarizes the procedure.

Consider a probabilistic model $p(\boldsymbol{x}, \boldsymbol{z})$ with data $\boldsymbol{x}$ and latent variables $\boldsymbol{z}$, and let $q(\boldsymbol{z}; \boldsymbol{\nu})$ be the variational approximation. The ELBO is

$$\mathcal{L}(\boldsymbol{\nu}) = \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\nu})}[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}; \boldsymbol{\nu})].$$

Define $g(\boldsymbol{z}, \boldsymbol{\nu}) = \log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}; \boldsymbol{\nu})$ to be the instantaneous ELBO. Then the derivative of the ELBO is

$$\begin{aligned}
\nabla_{\boldsymbol{\nu}} \mathcal{L} &= \nabla_{\boldsymbol{\nu}} \int q(\boldsymbol{z}; \boldsymbol{\nu}) g(\boldsymbol{z}, \boldsymbol{\nu}) d\boldsymbol{z} \\
&= \int \nabla_{\boldsymbol{\nu}} q(\boldsymbol{z}; \boldsymbol{\nu}) g(\boldsymbol{z}, \boldsymbol{\nu}) + q(\boldsymbol{z}; \boldsymbol{\nu}) \nabla_{\boldsymbol{\nu}} g(\boldsymbol{z}, \boldsymbol{\nu}) d\boldsymbol{z} \\
&= \int q(\boldsymbol{z}; \boldsymbol{\nu}) \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu}) g(\boldsymbol{z}, \boldsymbol{\nu}) + q(\boldsymbol{z}; \boldsymbol{u}) \nabla_{\boldsymbol{\nu}} g(\boldsymbol{z}, \boldsymbol{\nu}) d\boldsymbol{z} \\
&= \mathbb{E}_{q(\boldsymbol{z}; \boldsymbol{\nu})} \left[\nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu}) g(\boldsymbol{z}, \boldsymbol{\nu}) + \nabla_{\boldsymbol{\nu}} g(\boldsymbol{z}, \boldsymbol{\nu}) \right], \qquad (3.1)
\end{aligned}$$

where we use $\nabla_{\boldsymbol{\nu}} q(\boldsymbol{z}; \boldsymbol{\nu}) = q(\boldsymbol{z}; \boldsymbol{\nu}) \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu})$ between lines two and three. Interchanging derivatives and integrals requires technical conditions to be met. Derivatives are limits, so these conditions are given by the theorems for exchanging limits and integrals (the dominated convergence theorem or the monotone convergence theorem (Cinlar, 2011)).

We have written the derivative of the ELBO as an expectation with respect to the variational approximation. The derivative Equation (3.1) can be simplified using properties of the instantaneous ELBO. Namely,

$$\begin{aligned}
\mathbb{E}_{q(\boldsymbol{z}; \boldsymbol{u})} [\nabla_{\boldsymbol{\nu}} g(\boldsymbol{z}, \boldsymbol{\nu})] &= \mathbb{E}_{q(\boldsymbol{z}; \boldsymbol{u})} [\nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu})] \\
&= \int q(\boldsymbol{z}; \boldsymbol{\nu}) \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu}) d\boldsymbol{z} \\
&= \int \nabla_{\boldsymbol{\nu}} q(\boldsymbol{z}; \boldsymbol{\nu}) d\boldsymbol{z} \\
&= \nabla_{\boldsymbol{\nu}} \int q(\boldsymbol{z}; \boldsymbol{\nu}) d\boldsymbol{z} = \nabla_{\boldsymbol{\nu}} 1 = 0. \qquad (3.2)
\end{aligned}$$

The expected value of the score function $\nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu})$ of a distribution equals zero. Substituting this back into the derivative Equation (3.1) gives

$$\nabla_{\boldsymbol{\nu}} \mathcal{L} = \mathbb{E}_{q(\boldsymbol{z}; \boldsymbol{u})} [\nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu}) (\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}; \boldsymbol{\nu}))]. \qquad (3.3)$$

Algorithm 1: Black box variational inference (BBVI)

Input: Data $\boldsymbol{x}$, model $p(\boldsymbol{x}, \boldsymbol{z})$, approximating family $q(\boldsymbol{z}; \boldsymbol{\nu})$.
Initialize $\boldsymbol{\nu}$ randomly.
Initialize iteration counter i to zero.

while *change in* ELBO *is above some threshold* **do**

Draw S samples $\boldsymbol{z}_s \sim$ from $q(\boldsymbol{z}; \boldsymbol{\nu})$ the variational approximation.

Compute noisy gradient $\hat{\nabla}_{\boldsymbol{\nu}} \mathcal{L}$ using samples (Equation (3.4)).

Calculate step-size $\boldsymbol{\rho}^{(i)}$ (e.g., Robbins-Monro).

Update $\boldsymbol{\nu} \longleftarrow \boldsymbol{\nu} + \boldsymbol{\rho}^{(i)} \hat{\nabla}_{\boldsymbol{\nu}} \mathcal{L}$.

Increment iteration counter.

end
Return $\boldsymbol{\nu}$.

With this representation of the gradient of the ELBO as an expectation, we compute noisy unbiased gradients using Monte Carlo

$$\hat{\nabla} \mathcal{L} = \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}_s; \boldsymbol{\nu}) (\log p(\boldsymbol{x}, \boldsymbol{z}_s) - \log q(\boldsymbol{z}_s; \boldsymbol{\nu})),$$

$$\text{where } \boldsymbol{z}_s \sim q(\boldsymbol{z}_s; \boldsymbol{\nu}). \tag{3.4}$$

With Equation (3.4), we can use stochastic optimization to maximize the ELBO. This style of gradient has appeared in reinforcement learning (Williams, 1992) and in general Monte Carlo gradient estimation (Glynn, 1990).

Algorithm 1 summarizes this procedure. The requirements for this procedure are as follows:

- We need the score function of the variational approximation: $\nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu})$.
- We need to be able to sample from the variational approximation q.
- We need to be able to evaluate $\log p(\boldsymbol{x}, \boldsymbol{z})$ and evaluate $\log q(\boldsymbol{z}; \boldsymbol{\nu})$.

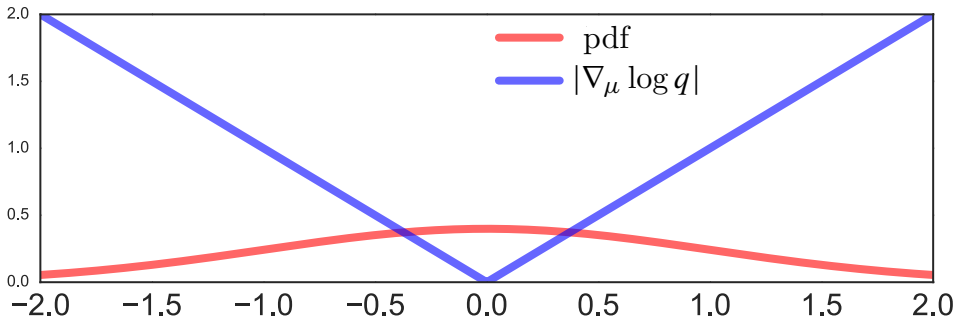


Figure 3.3: The score function for the mean of the normal distribution. It takes large values when the density is small. This leads to high variance. (Y-axis is labeled small as precise values are unimportant.)

The sampling algorithms and score functions depend only on the variational approximation, not the underlying model. Thus, we can build a collection of the functions needed, and reuse them across a broad class of models. We call these requirements the *black box criteria*. We did not make any assumptions about the form of the model, only that the practitioner can compute the logarithm of the joint distribution of the model. BBVI significantly reduces the effort needed to implement variational inference.

3.3 Controlling the Variance

We can use Equation (3.4) to maximize the ELBO, but the variance of the Monte Carlo gradients can be too large to be useful. Figure 3.3 plots the absolute score function for the mean of the standard Normal along with its density to understand the source of this variance. The score function for the mean takes on large values when the density is small. This leads to high variance.

High variance gradients require small step sizes; this slows down convergence. We develop two variance reduction methods: one based on Rao-Blackwellization (Casella and Robert, 1996) and the other on control variates (Ross, 2002). Both of these variance reductions take advantage of the structure of the ELBO and require no model-specific computations. This preserves the goal of black box variational inference.

3.3.1 Rao-Blackwellization

Rao-Blackwellization (Casella and Robert, 1996) reduces the variance of a random variable by replacing it with its conditional expectation with respect to a subset of the variables. The conditional expectation is a random variable with respect to the conditioning set.

In the simplest setting, Rao-Blackwellization replaces a function of two variables with its conditional expectation with respect to one of them. Consider two variables x and y, and let $\xi(x, y)$ be a function whose expectation with respect to the joint distribution of x, y that we seek.

Define $\tilde{\xi}(x) = \mathbb{E}[\xi(x, y) \mid x]$, then $\mathbb{E}[\tilde{\xi}(x)] = \mathbb{E}[\xi(x, y)]$. Therefore $\tilde{\xi}(x)$ can be used in place of $\xi(x, y)$ in Monte Carlo approximations of $\mathbb{E}[\xi(x, y)]$. The variance of $\tilde{\xi}(x)$ is

$$\mathrm{Var}(\tilde{\xi}(x)) = \mathrm{Var}(\xi(x, y)) - \mathbb{E}[(\xi(x, y) - \tilde{\xi}(x))^2].$$

The second term on the right is nonnegative, so $\mathrm{Var}(\tilde{\xi}(x))$ has lower variance than $\xi(x, y)$.

Now return to the problem of estimating the gradient of $\mathcal{L}$. Suppose there are d latent variables groups $\boldsymbol{z}_i$, and we are using the mean field variational family. Recall that each

random variable z_i is governed by its own variational distribution,

$$q(z; \nu) = \prod_{i=1}^{d} q(z_i; \nu_i).$$

Consider the gradient with respect to ν_i. Let $q_{(i)}$ be the distribution of z_i, its parents, its children, and its children's parents, i.e., the Markov blanket in p; let $p_{(i)}(x, z_{(i)})$ be the terms in the joint that contain $z_{(i)}$ and $p_{(-i)}(x, z_{(-i)})$ and $z_{(-i)}$ denote the respective complements. Then we can write the gradient with respect to ν_i as

$$\begin{aligned}
\nabla_{\nu_i} \mathcal{L} &= \mathbb{E}_q\big[\nabla_{\nu_i} \log q(z_i; \nu_i)(\log p_{(i)}(x, z_{(i)}) \\
&\quad + \log p_{(-i)}(x, z_{(-i)}) - \sum_{k=1}^{d} \log q(z_k; \nu_k))\big] \\
&= \mathbb{E}_{q(z_{(i)})}\big[\nabla_{\nu_i} \log q(z_i; \nu_i)\big(\log p_{(i)}(x, z_{(i)}) - \log q(z_i; \nu_i) \\
&\quad + \mathbb{E}_{q(z_{(-i)})}\big[\log p_{(-i)}(x, z_{(-i)}) - \sum_{k \neq i}^{d} \log q(z_k; \nu_k))\big]\big)\big]. \\
&= \mathbb{E}_{q(z_i)}\big[\nabla_{\nu_i} \log q(z_i; \nu_i)\big(\log p_{(i)}(x, z_{(i)}) - \log q(z_i; \nu_i)\big)\big] \\
&\quad + \mathbb{E}_{q(z_{(i)})}\big[\nabla_{\nu_i} \log q(z_i; \nu_i)\big]\mathbb{E}_{q(z_{(-i)})}\big[\log p_{(-i)}(x, z_{(-i)}) - \sum_{k \neq i}^{d} \log q(z_k; \nu_k))\big]. \\
&= \mathbb{E}_{q(z_{(i)})}[\nabla_{\nu_i} \log q(z_i; \nu_i)(\log p_{(i)}(x, z_{(i)}) - \log q(z_i; \nu_i))].
\end{aligned}$$

This follows directly from the expected value of the score function (Equation (3.2)). This equation says we can Rao-Blackwellize each component of the gradient with respect to the variables outside its Markov blanket, without having to construct model-specific conditional expectations.

We build a Monte Carlo estimate for the gradient of $\boldsymbol{\nu}_i$ by sampling the variational approximation

$$\hat{\nabla}\mathcal{L} = \frac{1}{S}\sum_{s=1}^{S} \nabla_{\boldsymbol{\nu}_i} \log q(\boldsymbol{z}_s; \boldsymbol{\nu})(\log p_{(i)}(\boldsymbol{x}, \boldsymbol{z}_s) - \log q(\boldsymbol{z}_s; \boldsymbol{\nu}_i)),$$

$$\text{where } \boldsymbol{z}_s \sim q_{(i)}(\boldsymbol{z}_s; \boldsymbol{\nu}).$$

We can use these gradients in place of Equation (3.4) to maximize the ELBO. (Note that the sample can be shared across each factor.)

Here we Rao-Blackwellized $\boldsymbol{z}_{(-i)}$, the complement of the variable z_i and its Markov blanket. Titsias and Lázaro-Gredilla (2015) expand the Rao-Blackwellization procedure to include $\boldsymbol{z}_i$. They rely on numeric integration to compute the expectation with respect to $\boldsymbol{z}_i$. This is feasible when $\boldsymbol{z}_i$ has low dimension or small discrete support.

3.3.2 Control Variates

Variance reduction methods work by replacing the function whose expectation is being estimated by another function that has the same expectation, but smaller variance. Formally, to estimate $\mathbb{E}_q[f]$ via Monte Carlo, we instead estimate $\mathbb{E}_q[\tilde{f}]$ where $\tilde{f}$ is chosen so $\mathbb{E}_q[f] = \mathbb{E}_q[\tilde{f}]$ and $\mathrm{Var}_q[f] > \mathrm{Var}_q[\tilde{f}]$.

Control variates (Ross, 2002) provide a way to construct a family of functions with equivalent expectation using functions with known expectation. Control variates have been used in a model-specific way for variational inference (Paisley et al., 2012) and have been used to control the variance in more general machine learning optimization problems (Wang et al., 2013).

Formally, for control variates, consider a function h which has known finite expectation, and let a be a scalar. Define $\tilde{f}$ to be

$$\tilde{f} := f - a\left(h - \mathbb{E}\left[h\right]\right). \tag{3.5}$$

This family of functions, indexed by a, has the same expectation as f, as required. Given a particular function h, we can choose a to minimize the variance of $\tilde{f}$. The variance of $\tilde{f}$ is

$$\mathrm{Var}(\tilde{f}) = \mathrm{Var}(f) + a^2\mathrm{Var}(h) - 2a\mathrm{Cov}(f, h).$$

This expression for the variance of $\tilde{f}$ shows that good control variates have high correlation with the function whose expectation is being estimated. Minimizing the variance gives the optimal value of a:

$$a^* = \frac{\mathrm{Cov}(f, h)}{\mathrm{Var}(h)}. \tag{3.6}$$

We apply this method to BBVI. The main requirement to apply control variates to BBVI is a function h that has known expectation without requiring model-specific computation. To meet this criterion, we choose h to be the score function $\nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu})$, which as shown above has expectation zero. This control variate relates to baseline estimation in reinforcement learning (Williams, 1992).

To apply our control variate to the Rao-Blackwellized gradient, we choose

$$\begin{aligned} f_j &= \nabla_{\boldsymbol{\nu}^j} \log q(\boldsymbol{z}; \boldsymbol{\nu}_i)(\log p_{(i)}(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}_i; \boldsymbol{\nu}_i)) \\ h_j &= \nabla_{\boldsymbol{\nu}^j} \log q(\boldsymbol{z}; \boldsymbol{\nu}_i), \end{aligned}$$

where the jth entry belongs to the ith factor.

We can estimate the variance and covariance for the optimal scaling in Equation (3.6). For the expectation of $\tilde{f}$ to remain unchanged, this estimate should be independent of the samples used in the Monte Carlo estimate of $\mathbb{E}[\tilde{f}]$. This can be done by either using a small number of held-out samples or in a round robin fashion where all but one sample are used to estimate $\hat{a}_j$, and the held-out sample is used to compute the Monte Carlo gradient.

Formally for the round robin procedure, given S samples, let s be the current held-out sample. Let $\hat{\mathrm{Cov}}(b, c; \mathcal{S})$ denote the empirical covariance between b and c evaluated on the sample collection $\mathcal{S}$, $\hat{\mathrm{Var}}$ be the empirical variance, and let $\boldsymbol{z}_{-s}$ be the collection of samples excluding the sth one. Then the estimate of the optimal scaling $\hat{a}_j^s$ not including sample s is

$$\hat{a}_j^s = \frac{\hat{\mathrm{Cov}}(f_j, h_j; \boldsymbol{z}_{-s})}{\hat{\mathrm{Var}}(h_j; \boldsymbol{z}_{-s})}.$$

Using this expression, the sth gradient estimate is

$$\hat{\nabla}^s_{\boldsymbol{\nu}^j} \mathcal{L} = \nabla_{\boldsymbol{\nu}^j} \log q(\boldsymbol{z}_s; \boldsymbol{\nu}_i)(\log p_{(i)}(\boldsymbol{x}, \boldsymbol{z}_s) - \log q(\boldsymbol{z}_s; \boldsymbol{\nu}_i) - \hat{a}_j^s). \tag{3.7}$$

The estimated $\hat{a}_j^s$ and the sample $\boldsymbol{z}_s$ are independent. The expectation matches that of the true gradient. Finally, we average over all S gradient estimates

$$\hat{\nabla}_{\boldsymbol{\nu}^j} \mathcal{L} = \frac{1}{S} \sum_{s=1}^{S} \hat{\nabla}^s_{\boldsymbol{\nu}^j} \mathcal{L}. \tag{3.8}$$

We call these leave-one-out control variates. Though this estimator appears to take quadratic time in the number of samples, it can be computed in linear time. This is done by first computing the required variance and covariance using a one pass algorithm and then removing each sample one-by-one. A similar process for baseline estimation in alternative lower bounds was developed in Mnih and Rezende (2016). The leave-one-out estimator combines both Rao-Blackwellization and control variates. Algorithm 2 summarizes this procedure. It

only requires samples from the variational distribution, computations about the variational distribution, and easy computations about the model.

We compare the leave-one-out estimator with the standard score function estimator for estimating the gradient of the following problem

$$\nabla_\mu \mathbb{E}_{q(z)}[\sigma(xz)], \quad \text{where } z \sim \text{Normal}(\mu, 1). \tag{3.9}$$

The score function estimator for this problem is

$$\nabla_\mu = \mathbb{E}_{q(z)}[(z-\mu)\sigma(xz)], \quad \text{where } z \sim \text{Normal}(\mu, 1).$$

To study the score function and leave-one-out estimators' tolerance to additional noise, we corrupt with gamma noise with varying shape and scale α,

$$\begin{aligned}\nabla_\mu &= \mathbb{E}_{q(z)}[(z-\mu)(\sigma(xz)+a)], \\ &\quad \text{where } z \sim \text{Normal}(\mu, 1), \text{ and } a \sim \text{Gamma}(\alpha, \alpha^{-1}).\end{aligned}$$

This corruption does not alter the gradient since $\mathbb{E}_q[\nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu})] = 0$.

Figure 3.4 displays the result. We see that the leave-one-out estimator handles the added noise much better than the score function estimator. Notice that the variance of the leave-one-out estimator also increases. The control variate can remove the non-random portion of a, $\mathbb{E}[a]$, but the variance it injects remains. This is why combining Rao-Blackwellization and the score function control variates is a good idea. Rao-Blackwellization removes both the random and non-random parts from the independent terms, while the control variate removes additional correlated noise. Finally, the estimate $\hat{a}$ is computed by Monte Carlo and is thus susceptible to the same variance issues. Rao-Blackwellization helps control the variance of this estimate.

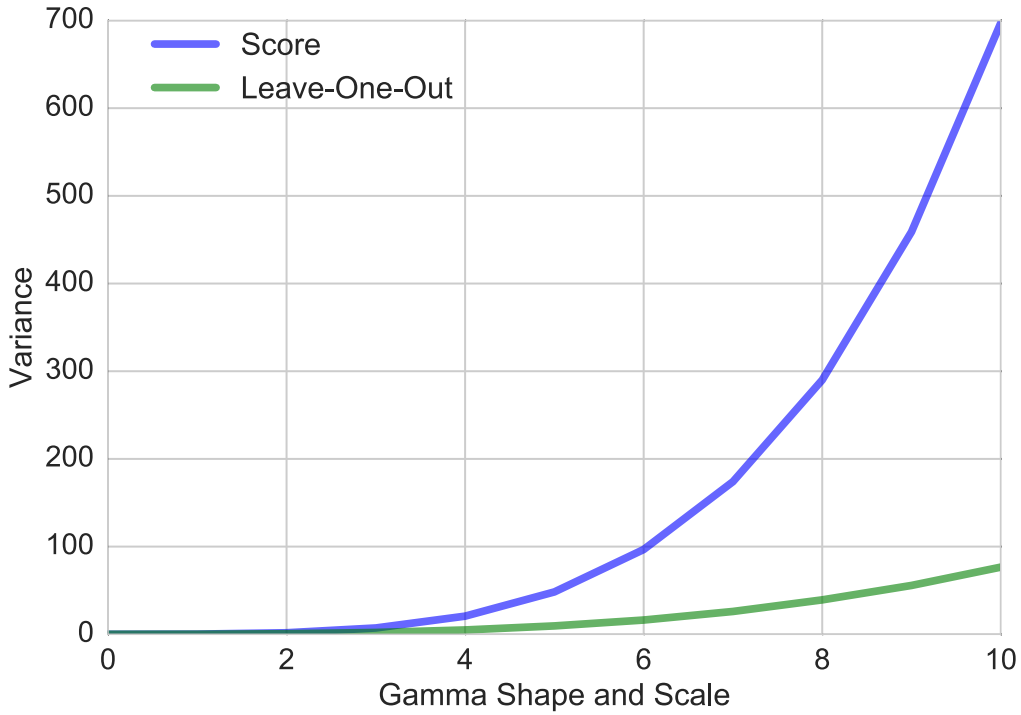


Figure 3.4: Variance comparison with and without leave-one-out control variates. The x-axis denotes the shape and scale of the added Gamma noise. The score function's variance grows rapidly. The leave-one-out control variate drastically reduces the variance. The leave-one-out control variate removes the constant offset due to the gamma noise (its expectation). However, its variance still grows due to the randomness of the gamma distribution.

What to do with additional facts about q? Sometimes we have additional knowledge about q such as its entropy, KL divergence, or mean. For optimal variance reduction, these facts should be used as control variates. For example, many variational implementations with analytic entropies (KL divergences) can have higher variance than stochastic entropies, depending on the correlation of the the entropy (KL divergence) with the joint (likelihood). The optimal use of this information would be to use them to define h in a control variate.

Algorithm 2: Black box variational inference (BBVI)

Input: Data $\boldsymbol{x}$, model $p(\boldsymbol{x}, \boldsymbol{z})$, approximating family $q(\boldsymbol{z}; \boldsymbol{\nu})$.
Initialize $\boldsymbol{\nu}$ randomly.
Initialize iteration counter i to zero.

while *change in* ELBO *is above some threshold* **do**
  Draw S samples $\boldsymbol{z}_s \sim$ from $q(\boldsymbol{z}; \boldsymbol{\nu})$ the variational approximation.
  for *each dimension* d *in* $1 : D$ **do**
    Compute covariance and variance for a^*.
    for *each sample* s *in* $1 : S$ **do**
      Compute $\hat{a}_d^s$ by removing sample s from the precomputed variance and covariance.
      Compute $\hat{\nabla}_{\boldsymbol{\nu}_d}^s \mathcal{L}$ using Equation (3.7).
    end
    Compute $\hat{\nabla}_{\boldsymbol{\nu}_d} \mathcal{L}$ using Equation (3.8).
  end
  Calculate step-size $\boldsymbol{\rho}^{(i)}$ (e.g., Robbins-Monro).
  Update $\boldsymbol{\nu} \longleftarrow \boldsymbol{\nu} + \boldsymbol{\rho}^{(i)} \hat{\nabla}_{\boldsymbol{\nu}} \mathcal{L}$.
  Increment iteration counter.
end
Return $\boldsymbol{\nu}$.

3.3.3 Reparameterization Gradients

An alternative way to reduce variance is to make more assumptions about the model $\log p(\boldsymbol{x}, \boldsymbol{z})$. One such assumption is to assume we have access to gradients of the model with respect to the latent variables: $\nabla_{\boldsymbol{z}} \log p(\boldsymbol{x}, \boldsymbol{z})$. This assumption rules out models with discrete random variables, but can lead to an estimator with lower variance for smooth models. This approach was developed by many (Kingma and Welling, 2014; Rezende et al., 2014; Titsias and Lázaro-Gredilla, 2014).

Recall the form of the original gradient of the ELBO from Equation (3.1),

$$\nabla_{\boldsymbol{\nu}} \mathcal{L} = \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{u})} \left[\nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{u}) g(\boldsymbol{z}, \boldsymbol{\nu}) + \nabla_{\boldsymbol{\nu}} g(\boldsymbol{z}, \boldsymbol{\nu}) \right].$$

If we make the reparmeterization assumption, i.e., that $z \sim q$ can be written as a deterministic transformation of parameter free noise source s,

$$\boldsymbol{\epsilon} \sim s, \;\; \boldsymbol{z} = t(\boldsymbol{\epsilon}, \boldsymbol{\nu}) \quad \rightarrow \quad \boldsymbol{z} \sim q(\boldsymbol{z}; \boldsymbol{\nu}),$$

we have that

$$\nabla_{\boldsymbol{\nu}} \mathcal{L} = \mathbb{E}_{s(\boldsymbol{\epsilon})} \left[\nabla_{\boldsymbol{\nu}} s(\boldsymbol{\epsilon}) g(t(\boldsymbol{\epsilon}, \boldsymbol{\nu}), \boldsymbol{\nu}) + \nabla_{\boldsymbol{\nu}} g(t(\boldsymbol{\epsilon}, \boldsymbol{\nu}), \boldsymbol{\nu}) \right].$$

Using the fact that $\nabla_{\boldsymbol{\nu}} s(\boldsymbol{\epsilon}) = 0$, we can simplify this with the multivariate chain rule to

$$\begin{aligned}
\nabla_{\boldsymbol{\nu}} \mathcal{L} &= \mathbb{E}_{s(\boldsymbol{\epsilon})} \left[\nabla_{\boldsymbol{\nu}} g(t(\boldsymbol{\epsilon}, \boldsymbol{\nu}), \boldsymbol{\nu}) \right] \\
&= \mathbb{E}_{s(\boldsymbol{\epsilon})} \left[\nabla_{\boldsymbol{z}} \left[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}; \boldsymbol{\nu}) \right] \nabla_{\boldsymbol{\nu}} t(\boldsymbol{\epsilon}, \boldsymbol{\nu}) - \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu}) \right] \\
&= \mathbb{E}_{s(\boldsymbol{\epsilon})} \left[\nabla_{\boldsymbol{z}} \left[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}; \boldsymbol{\nu}) \right] \nabla_{\boldsymbol{\nu}} t(\boldsymbol{\epsilon}, \boldsymbol{\nu}) \right].
\end{aligned}$$

This is the pathwise gradient (Fu, 2006; Glasserman, 2013) known as the reparameterization gradient in machine learning.

Figure 3.5 plots the variances for the reparameterization gradient, score function gradient, and score function gradient with control variates versus the covariate x for the toy regression problem from Equation (3.9). We iterate each estimator 100,000 times to compute the variance. We see that as the covariate increases in magnitude, the score function gradient become more favorable. This makes intuitive sense. The logistic function is bounded, but its gradient grows with x. The final curve shows the leave-one-out control variate. It performs best across the covariates.

Figure 3.5 compares variance estimates with 16 samples. For the reparameterization and score function gradients, changing the number of samples does not change the shape of the curves; each sample independently contributes to the estimate. However, the leave-one-out

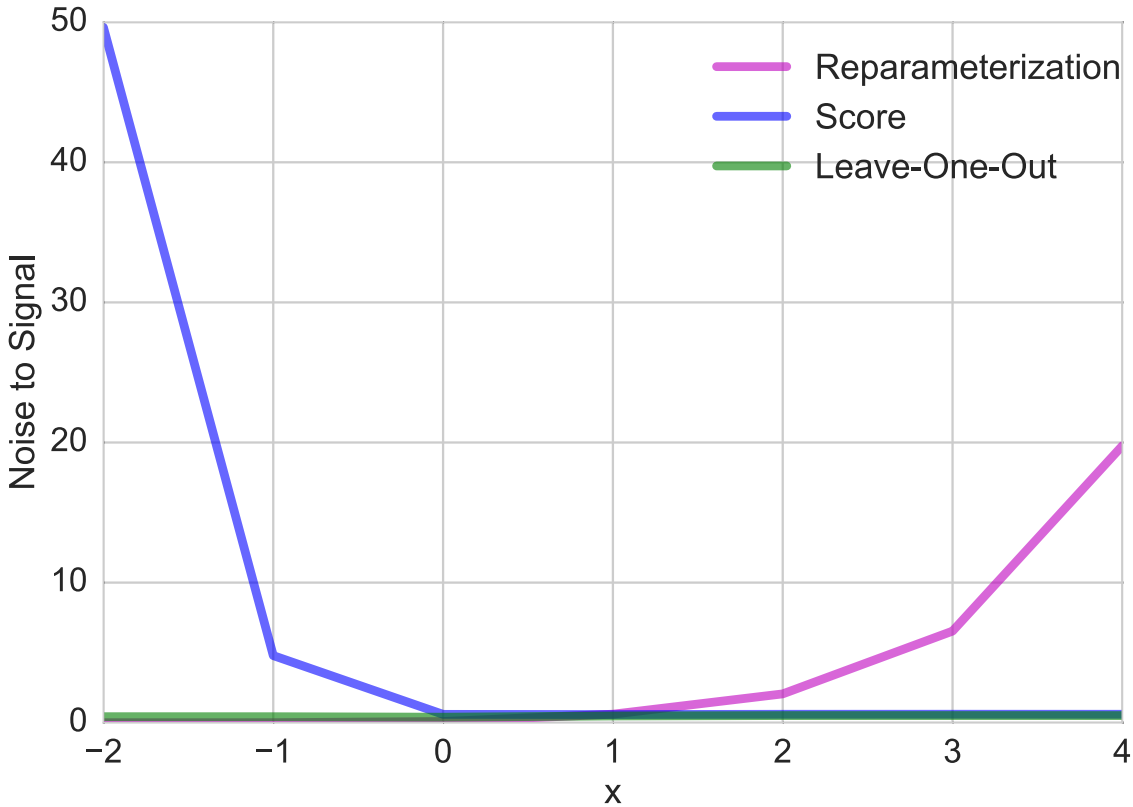


Figure 3.5: Noise-to-signal of the reparameterization, score function, and leave-one-out estimators as the function of a covariate x (see Equation (3.9)) on a logarithmic scale. Reparameterization gradients have lower variances for small covariate values, while score function gradients have lower variances for large covariate values. The intuition is that $\sigma(zx)$ is bounded, while its gradient grows with x. The leave-one-out estimator performs best.

estimator requires multiple samples. Figure 3.6 plots the same variance comparison with only 8 samples. Here the leave-one-out estimator's variance varies wildly. In general, the leave-one-out estimator controls the variance of the gradient well, but only with a sufficient number of samples.

What if we can use both score function and reparameterization gradients? The previous observation leads to the question of when do we use the score function gradient and when do we use the reparameterization gradient. This question can be answered by Monte Carlo variance analysis. Consider two functions f, g that have equivalent expectation. Then we can construct a family of functions h with equivalent expectation from convex combi-

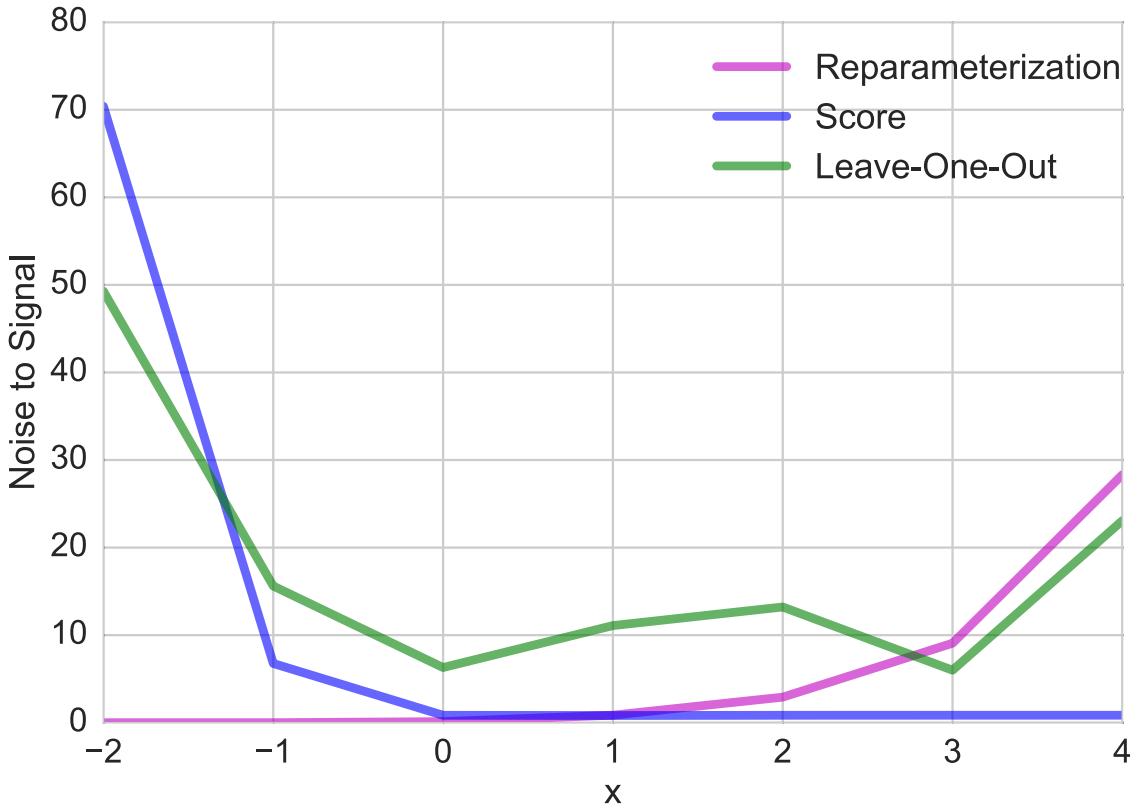


Figure 3.6: Same figure as Figure 3.5 except with 8 samples instead of 16. The character of the score function and reparameterization gradients do not change. They are an average of independent estimates. The leave-one-out estimate requires multiple samples. Its variances varies wildly with only 8 samples.

nations of f and g:

$$h = (1-\alpha)f + \alpha g.$$

To verify, $\mathbb{E}h = (1-\alpha)\mathbb{E}f + \alpha\mathbb{E}g = \mathbb{E}f = \mathbb{E}g$. We can now compute the variance of h and ask what value of α minimizes the variance. The variance of h is

$$\mathrm{Var}(h) = (1-\alpha)^2\mathrm{Var}(f) + 2\alpha(1-\alpha)\mathrm{Cov}(f,g) + \alpha^2\mathrm{Var}(g).$$

Optimizing this with respect to α gives

$$\alpha^* = \frac{\mathrm{Var}(f) - \mathrm{Cov(f, g)}}{\mathrm{Var}(f) - 2\mathrm{Cov}(f, g) + \mathrm{Var}(g)} = \frac{\mathrm{Var}(f) - \mathrm{Cov(f, g)}}{\mathrm{Var}(f - g)}. \tag{3.10}$$

The Monte Carlo estimate of h with α^* is guaranteed to have smaller variances than either the Monte Carlo estimate of f or the Monte Carlo estimate of g. If independent samples are used to estimate f and g, then α^* gives variance-weighted results:

$$\alpha^* = \frac{\mathrm{Var}(g)}{\mathrm{Var}(g) + \mathrm{Var}(f)} f + \frac{\mathrm{Var}(f)}{\mathrm{Var}(g) + \mathrm{Var}(f)} g.$$

The parameter α^* can be estimated by Monte Carlo. When considering variance, Equation (3.10) implies that when we have access to both score function and reparameterization gradients, we should use both in a weighted manner.

3.3.4 Conjugacy and Additional Information

Rao-Blackwellization and control variates match conditional conjugacy. If the model is conditionally conjugate, we can use Rao-Blackwellization to write gradients using complete conditionals. For the ith variable, this gives

$$\nabla_{\nu_i} \mathcal{L} = \mathbb{E}_{q(\boldsymbol{z})}[\nabla_{\boldsymbol{\nu}_i} \log q(\boldsymbol{z}_i; \boldsymbol{\nu}_i)(\log p(\boldsymbol{z}_i \mid \boldsymbol{z}_{-i}, \boldsymbol{x}) - \log q(\boldsymbol{z}_i; \boldsymbol{\nu}_i))].$$

When performing closed form variational inference in conditionally conjugate models, the mean field factors are chosen to be in the same exponential family as the complete conditionals. This yields

$$\nabla_{\nu_i} \mathcal{L} = \mathbb{E}_{q(\boldsymbol{z})}[(t(\boldsymbol{z}_i) - \nabla a(\boldsymbol{\nu}_i))(t(\boldsymbol{z}_i)t(\boldsymbol{z}_{-i})t(\boldsymbol{x}) - t(\boldsymbol{z}_i)^\top \boldsymbol{\nu}_i - a(\boldsymbol{\nu}_i)].$$

All quantities inside have known expectation (given by derivatives of the log-normalizers), and can thus be used as control variates. If these quantities are added as control variates to BBVI, we get zero-variance gradients (up to the estimation of the control variate scalings). Thus, BBVI with Rao-Blackwellization and sufficient statistic control variates match the gradients used by the classical conditional conjugacy algorithm. In this sense, these are not separate algorithms. We can handle conditionally conjugate models effectively within BBVI.

Distance metrics and variance. Gradients point in the direction of steepest ascent. This direction is the solution to

$$\arg\max_{d\boldsymbol{\nu}} \mathcal{L}(\nu + d\boldsymbol{\nu}), \quad \text{where} \quad ||d\boldsymbol{\nu}||^2 < \boldsymbol{\epsilon}^2,$$

for sufficiently small $\boldsymbol{\epsilon}$. Expanding this in the case of the ELBO gives

$$\begin{aligned} &\arg\max_{d\boldsymbol{\nu}} \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\nu}+d\boldsymbol{\nu})}[g(\boldsymbol{z};\boldsymbol{\nu}+d\boldsymbol{\nu})], \quad \text{where} \quad ||d\boldsymbol{\nu}||^2 < \boldsymbol{\epsilon}^2, \\ &= \arg\max_{d\boldsymbol{\nu}} \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\nu})}\left[\frac{q(\boldsymbol{z};\boldsymbol{\nu}+d\boldsymbol{\nu})}{q(\boldsymbol{z};\boldsymbol{\nu})} g(\boldsymbol{z};\boldsymbol{\nu}+d\boldsymbol{\nu})\right], \quad \text{where} \quad ||d\boldsymbol{\nu}||^2 < \boldsymbol{\epsilon}^2. \end{aligned}$$

The score function gradient is the limit of importance sampling (Glasserman, 2013). The variance of a Monte Carlo estimate of this estimator scales with how well small Euclidean perturbations of the distribution work with importance sampling. This can be poor. As an example, consider a normal distribution with variance σ^2 and mean μ. Consider the gradient with respect to μ:

$$\nabla_\mu \mathbb{E}_{q(z;\theta)}[f(z)] = \frac{1}{\sigma^2}\mathbb{E}[(z-\mu)f(z)].$$

The variance of this estimator scales with the inverse of the current variance σ^2. Contrast this with an equivalent parameterization of the normal where the mean is replaced by its

natural parameters $\eta = \frac{\mu}{\sigma^2}$

$$\nabla_\eta \mathbb{E}_{q(z;\theta)}[f(z)] = \mathbb{E}[(z-\mu)f(z)];$$

The variance of this estimator (up to first order) is now independent of the current variance σ^2. This can be more stable to optimize.

The problem in the original setup is that for small variances, a slight Euclidean perturbation in the mean can lead to a massive difference in the distributions. This is because the Euclidean distance is not a probability distance. Instead of finding the ascent direction via Euclidean perturbations, it can be beneficial to consider symmetric-KL perturbations or other distances like the χ-divergence (appropriate for variances) that include a notion of probability.

3.3.5 Data Subsampling

Though stochastic variational inference was developed for models that satisfy conditional conjugacy, the principle of data subsampling still applies. This follows from the decomposition of the log likelihood in hierarchical Bayesian models.

Recall that in a hierarchical Bayesian model we have a hyperparameter $\boldsymbol{\eta}$, global latent variables $\boldsymbol{\beta}$, local latent variables $\boldsymbol{z}_{1:n}$, and data $\boldsymbol{x}_{1:n}$. The log joint distribution of a hierarchical Bayesian model is

$$\log p(\boldsymbol{x}_{1:n}, \boldsymbol{z}_{1:n}, \boldsymbol{\beta}) = \log p(\boldsymbol{\beta}; \boldsymbol{\eta}) + \sum_{i=1}^{n} \log p(\boldsymbol{x}_i, \boldsymbol{z}_i \mid \boldsymbol{\beta}).$$

This is the same model class as Hoffman et al. (2013), but Hoffman et al. (2013) place restrictions on the forms of the distributions and the complete conditionals. Let us assume the variational approximation, with parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\phi}_{1:n}$, follows the true factorization of

the posterior

$$q(\boldsymbol{\beta}, \boldsymbol{z}_{1:n}) = q(\boldsymbol{\beta}; \boldsymbol{\lambda}) \prod_{i=1}^{n} q(\boldsymbol{z}_i \mid \boldsymbol{\beta}; \boldsymbol{\phi}_i).$$

From Equation (3.4), we see that to compute the gradient of the ELBO we need the log probability of p, the log probability of q, and the score function of q. If we instead use a noisy, unbiased estimate of the log-probability of p and q, the gradient remains unbiased. This estimate can be constructed by subsampling data points

$$\log p(\boldsymbol{\beta}; \boldsymbol{\eta}) - \log q(\boldsymbol{\beta}; \boldsymbol{\lambda}) + n(\log p(\boldsymbol{x}_i, \boldsymbol{z}_i \mid \boldsymbol{\beta}) - \log q(\boldsymbol{z}_i \mid \boldsymbol{\beta}; \boldsymbol{\phi}_i)).$$

Putting this together for a uniformly sampled data index i and samples $\boldsymbol{z}_{i,s}$ and $\boldsymbol{\beta}_s$ from the variational approximation, we have

$$\begin{aligned}\hat{\nabla}_{\boldsymbol{\lambda}} \mathcal{L} =& \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\beta}_s; \boldsymbol{\lambda})(\log p(\boldsymbol{\beta}_s; \boldsymbol{\eta}) - \log q(\boldsymbol{\beta}_s; \boldsymbol{\lambda}) \\ &+ n(\log p(\boldsymbol{x}_i, \boldsymbol{z}_{i,s} \mid \boldsymbol{\beta}_s) - \log q(\boldsymbol{z}_{i,s} \mid \boldsymbol{\beta}_s; \boldsymbol{\phi}_i))),\end{aligned}$$

and

$$\hat{\nabla}_{\boldsymbol{\phi}_i} \mathcal{L} = \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{z}_{i,s} \mid \boldsymbol{\beta}_s; \boldsymbol{\phi}_i)(n(\log p(\boldsymbol{x}_i, \boldsymbol{z}_{i,s} \mid \boldsymbol{\beta}_s) - \log q(\boldsymbol{z}_{i,s} \mid \boldsymbol{\beta}_s; \boldsymbol{\phi}_i))),$$

where the gradients for $\boldsymbol{\phi}_j$ for $j \neq i$ are zero by Rao-Blackwellization. It is not the distributional properties of the model that determine if it is amenable to subsampling; rather, it is the conditional independence structure that matters.

3.4 Related Work

Variational inference has undergone a boom in the last few years. In addition to the works referenced throughout the rest of the chapter, we highlight a few pieces of work here. Wingate and Weber (2013) consider a similar procedure to ours for probabilistic programs. Salimans et al. (2013) provide a framework based on stochastic linear regression. Their approach can be more simply described as one which uses the vector of score function control variates (each dimension uses all other dimensions as a control variate) and the biased control variate estimation procedure from the Monte Carlo literature (Owen, 2013). Carbonetto et al. (2009) present a stochastic optimization scheme for moment estimation based on the specific form of the variational objective when both the model and the approximating family are in the same exponential family. This differs from our general modeling setting where latent variables may be outside of the exponential family. Finally, Paisley et al. (2012) use Monte Carlo gradients for difficult terms in the variational objective and also use control variates to reduce variance. However, theirs is not a black box method. Both the objective function and control variates they propose require model-specific derivations.

3.5 Results

We use black box variational inference to quickly construct and evaluate several models on longitudinal medical data. We compare the speed and predictive likelihood of BBVI to sampling based methods. We demonstrate the ease with which several models can be explored.

3.5.1 Longitudinal Medical Data

Our data consists of longitudinal data from 976 patients (803 train + 173 test) from an ambulatory clinic at the Columbia University Medical Center (CUMC). The data are obtained from CUMC under IRB protocol. These patients visited the clinic a total of 33K times. During each visit a subset of labs (determined by the doctor) were collected. The labs measure the amount of a particular quantity (such as sodium) in the blood.

3.5.2 Evaluation and Hyperparameters

We evaluate models using predictive likelihood. To compute predictive likelihoods, we need an approximate posterior on latent variables shared across patients and on latent variables within patients. We use the training patients to get the approximate posterior for the variables shared across patients and use 75% of each test patient to compute the approximate posterior for latent variables within patients. We then calculate the predictive likelihood on the remaining 25%.

We initialize the approximations randomly and choose the mean field variational family. We use AdaGrad (Duchi et al., 2011) for learning rates. We use a large number of samples (1K) to estimate the gradient and use 100 samples to estimate the control variate scaling. Finally, we subsample data in batches of 25 for all experiments.

3.5.3 Model

Our goal is to come up with a low dimensional summary of each patient's labs that helps complete the sparse observations. Further, we want the summaries to be indicators of health. From this, we aim to summarize each visit with positive random variables.

To meet our modeling desiderata, we construct a Gamma-Normal time series (Gamma-Normal-TS) model. Let $\boldsymbol{x}_{p,v}$ be the lab values for patient p at visit v (some of the entries may be missing). Further, let $\boldsymbol{W}$ be a matrix of shared latent variables, $\boldsymbol{z}_{p,v}$ be latent variables for patient p at visit v, and let $\boldsymbol{o}_p$ be an intercept term for a particular patient. The generative model with hyperparameters σ is

$$\begin{aligned}
\boldsymbol{W} &\sim \text{Normal}(0, \sigma_w), \text{ an } L \times K \text{ matrix} \\
\boldsymbol{o}_p &\sim \text{Normal}(0, \sigma_o), \text{ a vector of length } L \\
\boldsymbol{z}_{p,1} &\sim \text{GammaE}(\sigma_z, \sigma_z), \text{ a vector of length } K \\
\boldsymbol{z}_{p,v} &\sim \text{GammaE}(\boldsymbol{z}_{p,v-1}, \sigma_z), \text{ a vector of length } K \\
\boldsymbol{x}_{p,v} &\sim \text{Normal}(\boldsymbol{W}\boldsymbol{z}_{p,v} + o_p, \sigma_x), \text{ a vector of length } L.
\end{aligned} \tag{3.11}$$

We set σ_w, σ_o, σ_z to be 1.0 and σ_x to be 0.01. GammaE is the expectation/variance parameterization of the (L-dimensional) gamma distribution. The mapping between this parameterization and the more standard shape α, rate β parameterization is

$$\mathbb{E} = \frac{\alpha}{\beta}, \qquad \text{Var} = \frac{\alpha}{\beta^2}.$$

Black box variational inference allows us to make use of non-standard parameterization for distributions that are easier to reason about. This is an important observation—the standard set of model families usable with classic variational inference methods is limited. In this model, the expectation parameterization of the gamma distribution allows the previous visit factors to define the expectation of the current visit factors. Finally, we emphasize that coordinate ascent variational inference and Gibbs sampling are not available for this model because the required conditional distributions do not have closed form.

3.5.4 Sampling Methods

We compare BBVI to a standard sampling based technique that only requires the joint distribution, Metropolis-Hastings (Bishop, 2006).[1]

Recall that Metropolis-Hastings works by sampling from a proposal distribution and accepting or rejecting the samples based on the likelihood. Standard Metropolis-Hastings works poorly in the Gamma-Normal-TS model due to the dimensionality. Instead, we compare to a Gibbs sampling method that uses Metropolis-Hastings to sample from the complete conditional of each variable. For proposal distributions, we use model distributions with mean equal to the value of the current parameter.

We compare BBVI to Metropolis-Hastings inside Gibbs. We used a fixed computational budget of 20 hours. Figure 3.7 plots the predictive likelihood versus time on a held-out set for both methods. We found similar results with different random initializations of both methods. Black box variational inference gives better predictive likelihoods and gets them in a shorter time.

3.5.5 Exploring Models

Black box variational inference enables us to quickly explore and fit many new models to a data set. We demonstrate this by considering three variations of our model for this data: Gamma-Normal, Gamma, and Gamma-TS.

[1] Methods that require more of the model such as Hamiltonian Monte Carlo (Duane et al., 1987) could work in this setting. However, as black box variational inference requires only the joint distribution and could benefit from the extra information used in complex methods, we compare only to Metropolis-Hastings within Gibbs.

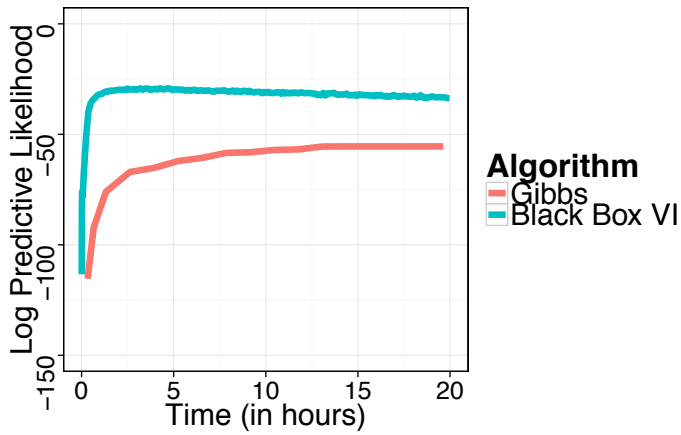


Figure 3.7: Comparison between Metropolis-Hastings inside Gibbs and BBVI. Black box variational inference yields better predictive results in a shorter time.

Gamma-Normal. This variant of the model in Equation (3.11) removes the temporal structure across visits.

$$
\begin{aligned}
\boldsymbol{W} &\sim \text{Normal}(0, \sigma_w), \text{ an } L \times K \text{ matrix} \\
\boldsymbol{o}_p &\sim \text{Normal}(0, \sigma_o), \text{ a vector of length } L \\
\boldsymbol{z}_{p,v} &\sim \text{GammaE}(\alpha_z, \beta_z), \text{ a vector of length } K \\
\boldsymbol{x}_{p,v} &\sim \text{Normal}(\boldsymbol{W}\boldsymbol{z}_{p,v} + o_p, \sigma_x), \text{ a vector of length } L.
\end{aligned}
$$

This is a factor model that represents each visit with a low dimensional latent variable, $\boldsymbol{z}_{p,v}$.

Gamma. This model requires the factors to be nonnegative. We replace all of the normally distributed variables with gamma distributed ones. The gamma model is a positive valued factor model, where all of the factors, weights, and observations have positive values. This limits the covariance between observed labs to be positive.

Models	Log-Predictive Likelihood
Gamma-Normal	-31.2
Gamma-Normal-TS	-30.0
Gamma-Gamma	-146
Gamma-Gamma-TS	-132

Table 3.1: A comparison between several models. Models that consider time and that allow for negative correlations between observations perform best. Without BBVI, fitting each of these models would have been a large undertaking.

Gamma-TS. We can link the factors through time using the expectation parameterization of the gamma distribution. (Note that this is harder with the usual natural parameterization of the gamma.) This changes $\boldsymbol{z}_{p,v}$ to be distributed as GammaE($\boldsymbol{W}\boldsymbol{z}_{p,v-1} + \boldsymbol{o}_p, \sigma_o$). In this model, the expected value of the factors at the next visit is the same as the value at the current visit. This allows us to propagate patient states through time. It is the nonegative analogue of the Gamma-Normal-TS model.

Model comparisons. We determined convergence using the change in the log-predictive likelihood on a validation set. We use a smaller learning rate for the Gamma models as optimization was less stable.

Table 3.1 summarizes our models along with their predictive likelihoods. We find that the Gamma models perform poorly. This is likely because they cannot capture the negative correlations that exist between different medical labs. By using black box variational inference, we were able to quickly fit and explore a set of complicated non-conjugate models. Without our generic algorithm, approximating the posterior of any of these models is a project in itself.

3.6 Conclusion

We developed black box variational inference, a new variational inference algorithm that drastically reduces the analytic burden on practitioners. This approach uses samples from the variational approximation to construct stochastic gradients. Essential to making it practical are model-free variance reductions. Black box variational inference is easy-to-use and simple for nonexperts to deploy on new models. With neuroscientists, Manning et al. (2014) used BBVI to build spatial factor models for functional MRI measurements of blood flow in the brain.

Black box variational inference opens the door to many directions. First, models that were impractical before now become possible to study. Second, variational approximations no longer need to be matched to the model based on conjugacy considerations. This means we can build new families of variational approximations that better approximate the posterior distribution. Finally, the choice of closeness between distributions, the KL divergence, was originally chosen for the computationally tractable algorithms that it yields. With black box variational inference, we can study new distances. We explore all of these directions in the subsequent chapters.

Chapter 4

Deep Exponential Families

New computational methods lead to new models. For example, the sigmoid belief network (Neal, 1990) hinged on the concurrent development of variational inference; the Gibbs sampler (Geman and Geman, 1984; Gelfand and Smith, 1990) ushered in the work on conditionally conjugate models, and MCMC methods for the Dirchlet process (Neal, 2000) fertilized the growth of Bayesian nonparametric methods. In a similar vein, black box variational inference (BBVI) reduces the barrier to exploring new kinds of models. We explore one such family of models here, deep exponential families (DEFS). DEFS capture a hierarchy of dependencies between latent variables. They build upon the intuitions behind deep unsupervised feature learning. Through the flexibility of exponential families, DEFS generalize to many settings.

As an example, consider modeling documents. We can represent a document as a vector of term counts modeled by Poisson random variables. In one type of DEF, the rates of these Poisson random variables come from a layer of latent variables specific to a document. These variables are scaled by weights that are shared across data. The document specific latent layers can be thought of as factor activations. Normally, in conditionally conjugate models the modeling process would stop here. This happens because conjugate priors of

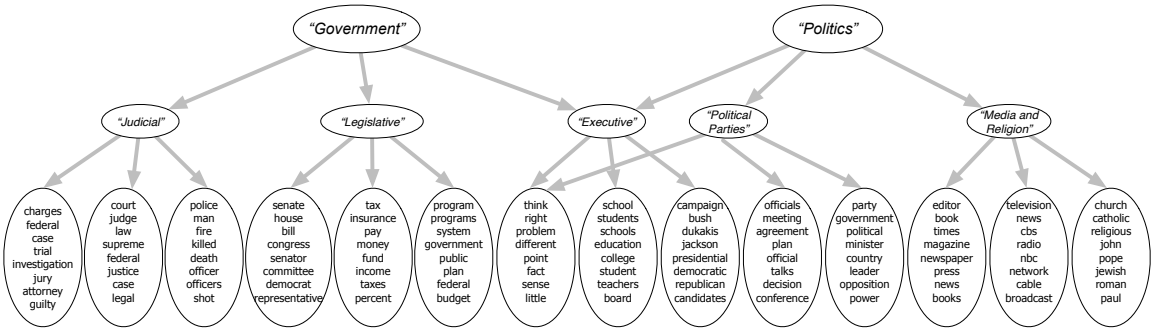


Figure 4.1: A fraction of the three layer factor hierarchy on 166K *the New York Times* articles. The top words are shown for each factor. The arrows represent hierarchical groupings.

conjugate priors quickly become intractable. As an example, consider the conjugate prior of the Dirichlet distribution. However, with BBVI we can proceed. One way to build a DEF is to model the factor activations in a similar way. That is, the factor activations come from a higher level of activations, scaled by weights shared across data. Just as the factors group related words, the higher level factor activations group related factors.

Figure 4.1 displays an example of a three level DEF with nonnegative factors uncovered from a large set of articles in *the New York Times.* (This style of model, though with different details, has been previously studied in the topic modeling literature (Li and McCallum, 2006).) Conditional on the observed word counts, the DEF defines a posterior distribution of the per-document latent variables and latent variables shared across data. This figure visualizes two of the third-layer factors, which correspond to the themes of "Government" and "Politics". We focus on "Government" and notice at the second level that the the model has discovered the three branches of government: judiciary (left), legislative (center), and executive (right),

In DEFS, the latent variables can be from any exponential family. For example, Bernoulli latent variables recover the classical sigmoid belief network (Neal, 1990); gamma latent variables give deep variations of nonnegative matrix factorization (Lee and Seung, 1999); and Gaussian random variables lead to the deep latent Gaussian models (Rezende et al.,

2014). All of these models fall broadly into the class of stochastic feedforward networks (Neal, 1990). We will develop several examples in this chapter.

DEFS can be used as drop-in replacements for distributions in traditional models from statistics and machine learning, such as for grouped data (Gelman and Hill, 2007), sequential data (Blei and Lafferty, 2006b), or pairwise data (Salakhutdinov and Mnih, 2008). As a concrete example, we develop and study the *double DEF*. The double DEF models a matrix of pairwise observations such as users rating items. It has one DEF for the users and one for the items. The observed matrix entries depend on the DEFS for both the users and the items.

In this chapter we develop DEFS, describe posterior inference for them using BBVI, and show how they can be used as building blocks in more complex models.

4.1 Deep Exponential Families

Deep exponential families build upon exponential families. Recall that exponential families have the following form

$$p(\boldsymbol{z}; \boldsymbol{\eta}) = h(\boldsymbol{z}) \exp(\boldsymbol{\eta}^\top T(\boldsymbol{z}) - a(\boldsymbol{\eta})),$$

where h is the base measure, $\boldsymbol{\eta}$ are the natural parameters, T are the sufficient statistics, and a is the log-normalizer. The expectation of statistics T provides an alternative parameterization for exponential families via the relation $\mathbb{E}[T(x)] = \nabla_\eta a(\eta)$ (Brown, 1986). The sufficient statistics and base measure[1] characterize an exponential family. For example, if the base measure is $h = \sqrt{2\pi}$ and the sufficient statistics are $T(z) = [z, z^2]$, we get the Nor-

[1]Technically, this can be absorbed into the dominating measure for the family. In this case the dominating measure and sufficient statistics characterize the exponential family.

mal distribution. If the base measure is one on the unit interval, and the sufficient statistics are $T(z) = [\log z, \log 1 - z]$, we get the Beta distribution.

Deep exponential families. Deep exponential families chain together exponential families to form a hierarchy, where the draw from one layer controls the parameters of the next.

For each data point $\boldsymbol{x}_n$, the model has L layers of latent variables $\boldsymbol{z}_{n,1}, ..., \boldsymbol{z}_{n,L}$, where each $\boldsymbol{z}_{n,\ell} = \{z_{n,\ell,1}, ..., z_{n,\ell,K_\ell}\}$. We assume that $z_{n,\ell,k}$ is a scalar, but the model generalizes beyond this. Shared across data, the model has $L - 1$ hidden parameters $\boldsymbol{W}_1, ..., \boldsymbol{W}_{L-1}$. These hidden parameters have a prior distribution $p(\boldsymbol{W}_\ell)$.

For simplicity, we omit the data index n and describe the distribution of a single data point $\boldsymbol{x}$. Given a hyperparameter $\boldsymbol{\eta}$, the top layer of latent variables follows

$$\boldsymbol{z}_L \sim \textsc{expfam}(\boldsymbol{\eta}),$$

where $\textsc{expfam}(\boldsymbol{\eta})$ denotes an exponential family with natural parameter $\boldsymbol{\eta}$.

In lower layers, each latent variable is drawn conditional on the previous layer:

$$p(\boldsymbol{z}_\ell \mid \boldsymbol{z}_{\ell+1}, \boldsymbol{W}_\ell) = \textsc{expfam}_\ell\left(g_\ell(\boldsymbol{z}_{\ell+1}, \boldsymbol{W}_\ell)\right).$$

The subscript ℓ on $\textsc{expfam}$ denotes that the family can change at each layer. The function g_ℓ defines a mapping from the higher level to the natural parameters of the current level. This mapping is controlled by hidden parameters $\boldsymbol{W}$. We call the function g_ℓ the link function. As an example, consider the inner product link function. Let g_ℓ be a parameter-free function such as z^2 or $\log(z)$. Then the inner product link sets the layer conditional distribution

to

$$p(z_{\ell,k} \mid \boldsymbol{z}_{\ell+1}, \boldsymbol{w}_{\ell,k}) = \textsc{expfam}\left(g_\ell(\boldsymbol{z}_{\ell+1}^\top \boldsymbol{w}_{\ell,k})\right). \tag{4.1}$$

Here $\boldsymbol{W}_\ell$ is a matrix and $\boldsymbol{w}_{\ell,k}$ is a vector of dimension K_ℓ. DEFS of this form are random effects models (Gelman et al., 2003) where each layer of variables are controlled by the product of a weight vector and a set of latent variables. Other example link functions include splines and multilayer neural networks. We will focus on the inner product family of links for the rest of this chapter.

Likelihood. The likelihood defines the generative process for the data conditional on the hidden structure. In the simplest case for a DEF, the likelihood depends on the lowest layer of the DEF, $p(\boldsymbol{x} \mid \boldsymbol{z}_1)$. Separating out the likelihood allows for simpler compositions and embeddings of DEFS in other models.

As a concrete example, we will assume the nth observation $\boldsymbol{x}_n$ is a vector of counts modeled with a Poisson distribution. The Poisson distribution with mean (rate) ζ is

$$p(\boldsymbol{x}_{n,i} = x) = e^{-\zeta} \frac{\zeta^x}{x!}.$$

If we let $x_{n,i}$ be the count of type i, then the likelihood is

$$p(\boldsymbol{x}_{n,i} \mid \boldsymbol{z}_{n,1}, \boldsymbol{W}_0) = \text{Poisson}(g_0(\boldsymbol{z}_{n,1}^\top \boldsymbol{w}_{0,i})).$$

The observation weights have prior $p(\boldsymbol{W}_0)$, and the choice of function g_0 ensures the rate parameter's positivity constraint is satisfied. (This likelihood can be generalized to $g_0(\boldsymbol{z}_{n,1}, \boldsymbol{w}_{0,i})$.) For example, in the case where $\boldsymbol{z}$ is nonnegative, $\boldsymbol{W}_0$ can have a nonnegative prior and g_0 can be the identity function.

Returning to the example of modeling documents, $\boldsymbol{x}_n$ represents term counts. This means that the first layer of the DEF groups similar terms, the second layer groups the groups into higher level groups, and so on. Figure 4.1 depicts the compositionality and sharing semantics of deep exponential families.

Nonlinearities. The nonlinearities in DEFS arise from two sources. The first source is explicit; it is the link function. The second source is implicit and comes from the properties of exponential families.

Using properties of exponential families we can determine how the link function alters the distribution of the ℓth layer. The moments of the sufficient statistics of an exponential family completely characterize the distribution. They are given by the gradient of the log-normalizer:

$$\mathbb{E}[T(z_{\ell,k})] = \nabla a(g_\ell(\boldsymbol{z}_{\ell+1}^\top \boldsymbol{w}_{\ell,k})). \tag{4.2}$$

Thus, in DEFS the mean of the next layer is controlled by both the link function g_ℓ and the gradient of the log-normalizer. In the case of the identity link function, the expectation of latent variables in deep exponential families gets transformed by the log-normalizer at each level.

Here is a diagram of the transformation process.

$$\begin{tikzcd}
\mathbf{w}_{\ell,k} \arrow[dr] & & \\
& g \arrow[r, "\nabla a"] & \mathrm{E}[T(z_{n,\ell,k})] \\
\mathbf{z}_{n,\ell+1} \arrow[ur] & &
\end{tikzcd}$$

For example, in the sigmoid belief network (Neal, 1990), we will see that the identity link recovers the sigmoid transformation used in in neural networks.

4.2 Examples

To demonstrate the possibilities with deep exponential families, we present three examples: the sparse gamma DEF, the sigmoid belief network, and Poisson DEFS.

4.2.1 Sparse Gamma DEF

The sparse gamma DEF builds upon the gamma distribution. The gamma distribution has support in the positive reals and is a member of the exponential family. It has two parameters, the shape α and the rate β. In its exponential family form the gamma density is

$$p(z) = \frac{1}{z} \exp(\alpha \log(z) - \beta z - \log \Gamma(\alpha) + \alpha \log(\beta)),$$

where Γ is the gamma function. The mean of the gamma distribution is $\mathbb{E}[z] = \alpha\beta^{-1}$. Figure 4.2 plots the gamma density for various shape and rates. We can see that both parameters are well named. Changing the shape parameter alters the look of the density function, while changing the rate only rescales it.

Gamma distributions with shape smaller than one exhibit a peculiar property — they asymptote at zero. This is a kind of soft sparsity, where the majority of the probability mass remains close to zero. This type of distribution is akin to a soft spike-and-slab prior (Ishwaran and Rao, 2005). Spike-and-slab priors perform well on feature selection and unsupervised feature discovery (Goodfellow et al., 2012; Hernández-Lobato et al., 2013). We will use this observation to build sparse gamma DEFS.

For sparse gamma DEFS, we let the components in the higher level control the expected value of the next hidden layer, while keeping the shape fixed to be less than one to get soft sparsity. Let α_ℓ be the shape at layer ℓ, then the link function for the sparse gamma DEF

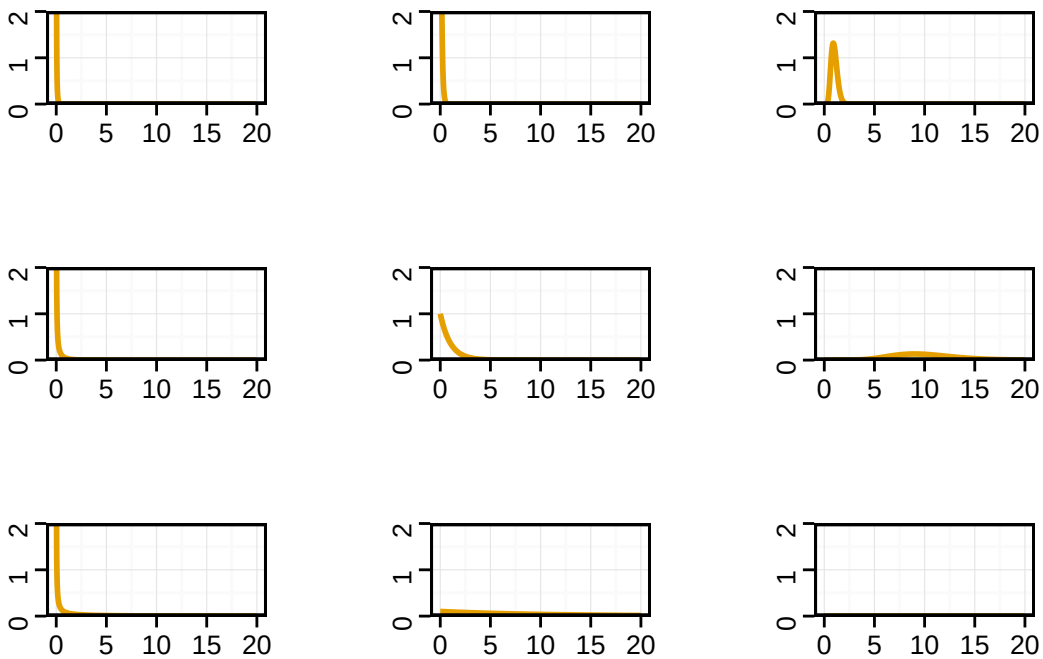


Figure 4.2: Gamma distribution at various shapes and scales. Moving to the right increases the shapes by an order of magnitude, while moving down decreases the rates by an order of magnitude. When the shapes are small, most of the mass is near zero.

is

$$g_\alpha = \alpha_\ell, \qquad g_\beta = \frac{\alpha_\ell}{\boldsymbol{z}_{\ell+1}^\top \boldsymbol{w}_{\ell,k}}.$$

The rate of the gamma distribution has positive constraints, so we place a gamma prior on the weights too.

The sparse gamma distribution differs from distributions such as the Poisson and the normal in how probability mass moves when the mean changes. For example, when the expected value is high, draws from the Poisson are likely to be high. However, in the sparse gamma distribution draws will either be close to zero or very large. Figure 4.3 demonstrates this. Changing the mean stretches the spike-and-slab. Hence, the sparse gamma DEF is a mul-

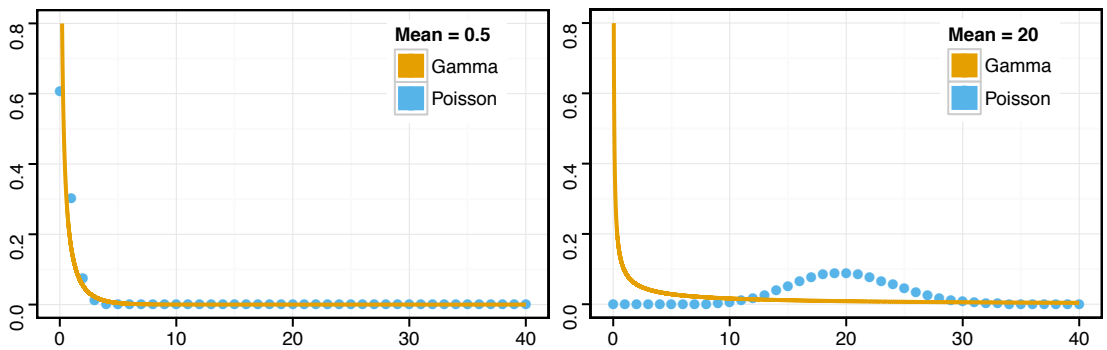


Figure 4.3: Sparse gamma distribution versus the Poisson at two different means. Increasing the mean moves the peak of the Poisson, but not the sparse gamma.

tilevel soft spike-and-slab model. The hierarchy shown in Figure 4.1 comes from a sparse gamma deep exponential families.

4.2.2 Sigmoid Belief Network

Recall the sigmoid belief network (Neal, 1990; Mnih and Gregor, 2014) discussed in Chapter 2. It consists of latent Bernoulli layers that represent the on-or-off state of a feature. The distribution of a feature at layer ℓ is controlled by the previous layer $\ell + 1$ scaled by a weight vector. Its mean is the sigmoid transformation of the inner product of the two.

Consider, an example of a DEF with Bernoulli layers and identity link. The form of the conditional Equation (4.1) in this setup is

$$p(z_{\ell,k} \mid \boldsymbol{z}_{\ell+1}, \boldsymbol{w}_{\ell,k}) = \exp(\boldsymbol{z}_{\ell+1}^\top \boldsymbol{w}_{\ell,k} z_{\ell,k} - \log(1 + \exp(\boldsymbol{z}_{\ell+1}^\top \boldsymbol{w}_{\ell,k}))).$$

From Equation (4.2), we have that the expected value is the derivative of the log-normalizer:

$$\mathbb{E}[z_{\ell,k} \mid \boldsymbol{z}_{\ell+1}, \boldsymbol{w}_{\ell,k}] = \nabla \log(1 + \exp(\cdot))|_{\boldsymbol{z}_{\ell+1}^\top \boldsymbol{w}_{\ell,k}} = \sigma(\boldsymbol{z}_{\ell+1}^\top \boldsymbol{w}_{\ell,k}).$$

The conditional expectation of a hidden variable given the layer above and parameters is the sigmoid transformation of the inner product of the previous layer and weights. A DEF with Bernoulli exponential families and identity link function is equivalent to the sigmoid belief network. The weights in the sigmoid belief network are real valued, so we place a normal prior over the weights shared across data. We allow for each feature to have an intercept term. This controls the baseline activation of each feature independent of the other layers.

4.2.3 Poisson DEF

The sigmoid belief network models each observation with a cascade of binary switches that represent whether a feature is expressed in that observation. This can be limiting. Observations may exhibit multiple instances of a feature, for example in an image that contains multiple spoons. Here we develop the multi-feature generalization of the sigmoid belief network in which observations can exhibit a feature a count-valued number of times.

The Poisson distribution models counts. Its exponential family form with parameter η is

$$p(z) = \frac{1}{z!} \exp(\eta z - \exp(\eta)).$$

Its mean is e^{η}.

Replacing the binary-valued Bernoulli distribution in the sigmoid belief network with the count-valued Poisson distribution gives the multi-feature generalization of the sigmoid belief network. Consider the Poisson DEF with identity link function

$$p(z_{\ell,k} \mid \boldsymbol{z}_{\ell+1}, \boldsymbol{w}_{\ell,k}) = \frac{1}{z_{\ell,k}} \exp(\boldsymbol{z}_{\ell+1}^{\top} \boldsymbol{w}_{\ell,k} z_{\ell,k} - \exp(\boldsymbol{z}_{\ell+1}^{\top} \boldsymbol{w}_{\ell,k})).$$

The counts of the Poisson $z_{\ell+1,k}$ represent how many times the feature $\boldsymbol{w}_{\ell,k}$ occurs in an observation. Using the link function property of DEFS, the mean activation of each Poisson is

$$\mathbb{E}[z_{\ell,k}] = \exp(\boldsymbol{z}_{\ell+1}^{\top}\boldsymbol{w}_{\ell,k}).$$

In practice, we found that the exponential function leads to instability in optimization for large natural parameters (because the derivative of the expected value gets exponentially large, unlike in the sigmoid belief network). Instead we consider the log-softplus link function, $g(\cdot) = \log\log(1+\exp(\cdot))$. The log-softplus link function implies a derivative close to one for the expected value at large inputs. Similar to the sigmoid belief network, we allow each hidden variable to have an intercept term.

Poisson DEFS can also be adapted for nonnegative matrix factorization. We do this by using the log-link. This implies that the expected value of one of the hidden variables $z_{\ell,k}$ is $\boldsymbol{z}_{\ell+1}^{\top}\boldsymbol{w}_{\ell,k}$. Poisson random variables and their expectations are nonnegative. With a prior over nonnegative values on the weights, we get a variant of deep nonnegative matrix factorization in which the factor expressions are discrete. This contrasts the identity and log-softplus links that allow both positive and negative weights.

4.3 Connections

Via its relationship to exponential families, hierarchical models, and deep models, deep exponential families connect to many areas. Here we discuss several of them.

Generalized linear models. Generalized linear models (McCullagh and Nelder, 1989) provide a mechanism to build regression models from exponential families, such as, count-

valued regression using the Poisson distribution. If $\boldsymbol{x}_i$ are features or covariates (like house size) and $\boldsymbol{y}_i$ are responses (like house cost), a generalized linear model is

$$\boldsymbol{y}_i \sim \textsc{expfam}(g(\boldsymbol{x}_i^\top \boldsymbol{w})),$$

where $\boldsymbol{w}$ are the regression coefficients. Note that this is a slight redefinition of the link function from the classic generalized linear models literature. Generalized linear models make predictions via conditional expectations. Given a covariate $\boldsymbol{x}$, the prediction is $\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{x}]$. Compare generalized linear models to the DEF layer Equation (4.1). We see that, conditional on the higher level $\boldsymbol{z}_{\ell+1}$, each hidden DEF unit is a generalized linear model with covariates $\boldsymbol{z}_{\ell+1}$ and regression coefficients $\boldsymbol{W}_{\ell,k}$. DEFS are nested generalized linear models, conditional on the top layer.

Randomness vs. determinism. The most general conditional layer distribution can be represented as a function f of a variable $\boldsymbol{\epsilon}_\ell$ from noise source s and the layer above $\boldsymbol{z}_{\ell+1}$,

$$\boldsymbol{z}_\ell = f(\boldsymbol{z}_{\ell+1}, \boldsymbol{\epsilon}_\ell), \quad \boldsymbol{\epsilon}_\ell \sim s, \tag{4.3}$$

with noise dimension K_ℓ. This describes stochastic feedforward networks.

For deep exponential families, if the noise is uniform, the transformation f is the inverse cumulative distribution of an exponential family with parameters $\boldsymbol{W}_\ell \boldsymbol{z}_{\ell+1}$. The restriction to a normalized exponential family in DEFS guarantees the existence of a joint likelihood function. The existence of a likelihood function simplifies inference.

Compare the general layer construction with deterministic feedforward neural networks, where $\boldsymbol{z}_{\ell+1}$ is the input and $\boldsymbol{z}_\ell$ is the output,

$$\boldsymbol{z}_\ell = f(\boldsymbol{z}_{\ell+1}). \tag{4.4}$$

The difference between this transformation and the one in Equation (4.3) is the presence of noise. We can write the deterministic feedforward neural network transformation (Equation (4.4)) as a transformation of delta mass noise (noise that takes a fixed value, say zero, with probability one):

$$\boldsymbol{z}_\ell = f(\boldsymbol{z}_{\ell+1}, \boldsymbol{\epsilon}_\ell), \quad \boldsymbol{\epsilon}_\ell \sim \delta_0.$$

This equation unifies the deterministic feedforward neural network with general conditional layer distributions. It reveals two things. First, deterministic feedforward neural networks are a subset of the general conditional ones. Second, inference with the conditional transforms is harder. In the stochastic case, inference has to adjudicate on what the noise $\boldsymbol{\epsilon}_\ell$ explains and what the signal $\boldsymbol{z}_{\ell+1}$ explains (This is the explaining away problem.). In the deterministic setting, the posterior on the noise is fixed and known, thus only the signal $\boldsymbol{z}_{\ell+1}$ can explain the data.

The distinction between the signal $\boldsymbol{z}_{\ell+1}$ and noise $\boldsymbol{\epsilon}_\ell$ is made by the practitioner, who uses $\boldsymbol{z}_{\ell+1}$ to encode structure. This structure is then modeled with parameters shared across data points, while the noise remains unmodeled.

The representation in Equation (4.3) implies that any latent variable can be represented as a deterministic transformation of a uniform random variable. Why then do we need distributions beyond a fixed one like the uniform or the normal distribution? The problem lies in the complexity of the transformation. As a concrete example, consider mapping the uniform distribution to a gamma distribution with shape α and rate β. The true mapping f

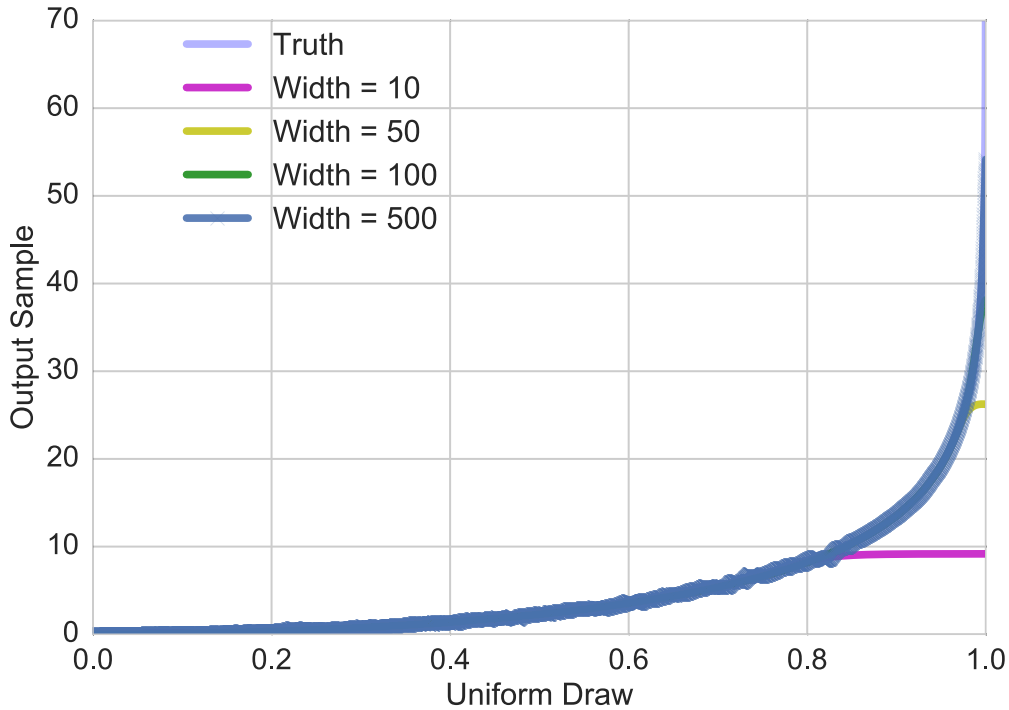


Figure 4.4: Here we use various width neural networks with five layers to learn the transformation from a uniform distribution to a gamma distribution. It takes a width of 100 to get a close approximation. The roughness comes from the width-500 network. The width-100 neural network has more than 25,000 parameters. A simple change in distribution can greatly reduce the number of parameters needed.

is

$$f(\cdot) = \text{CDF-Gamma}^{-1}(\cdot).$$

We plot the approximations to this function for shape 0.5 and rate 0.1 using five layer neural networks with varying widths in Figure 4.4. We compute the approximation by minimizing the absolute distance to the true function on a grid of points in the uniform interval. The widths need to get sufficiently large (≥ 100) for the neural networks to get close. The width-100 neural network has more than 25,000 parameters.[2] The use of alternative distributions

[2]The network used hyperbolic tangent nonlinearities. We tried rectifier activations, but we found that they output only values close to zero.

is a form of model structure that can drastically reduce the amount of data and parameters needed.

Do we need latent variables? Models that transform randomness $\boldsymbol{\epsilon}$ can produce arbitrary densities. This leads to the question: whether we need latent variables (holding computational considerations aside)?

As an example, consider the case where true data is generated as follows. Let $\boldsymbol{z}_i$ be a K dimensional vector, and let $\boldsymbol{W}$ be a matrix of size $L \times K$, where $L > K$. Then the data is generated as

$$\begin{aligned} \boldsymbol{z}_i &\sim p \\ \boldsymbol{\epsilon}_i &\sim s \\ \boldsymbol{x}_i &= \boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{\epsilon}_i, \end{aligned} \tag{4.5}$$

where $\boldsymbol{\epsilon}$ is L dimensional. We will now consider a pair of models to fit the data from the true model.

Let f be an arbitrary function, the first model is

$$\begin{aligned} \boldsymbol{\epsilon}_i &\sim s \\ x_i &= f(\boldsymbol{\epsilon}_i). \end{aligned}$$

Since f is arbitrary, this model can mimic the model in Equation (4.5). However, this model cannot separate out the K dimensional signal $\boldsymbol{z}_i$ that appears in the true model. Latent variables add value in that they can be used to posit and recover hidden structure.

For a second model, suppose we know *a priori* that the dimensionality of noise is L dimensional and the dimensionality of the signal is K dimensional, like in the previous model

(Equation (4.5), where $\boldsymbol{z}_i$ is K dimensional and $\boldsymbol{\epsilon}_i$ is L dimensional). Then we could propose the following model that posits both signal and noise

$$\begin{aligned}
\boldsymbol{z}_i &\sim p \\
\boldsymbol{\epsilon}_i &\sim s \\
\boldsymbol{x}_i &= f(\boldsymbol{z}_i, \boldsymbol{\epsilon}_i).
\end{aligned}$$

To understand the problem with this approach, we can compute the posterior distribution on $\boldsymbol{z}_i$ by writing down the ELBO in terms of the true model M,

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}_{\boldsymbol{x}\sim M}[\mathbb{E}_q[\log p(\boldsymbol{x} \mid \boldsymbol{z}) + \log p(\boldsymbol{z}) - \log q(\boldsymbol{z})]] \\
\mathcal{L} &= \mathbb{E}_M[\mathbb{E}_q[\log p(\boldsymbol{x} \mid \boldsymbol{z})]] - \mathrm{KL}(q||p),
\end{aligned}$$

where the noise $\boldsymbol{\epsilon}_i$ is hidden in the likelihood ($p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) = \int_{\boldsymbol{\epsilon}: f(\boldsymbol{\epsilon}, \boldsymbol{z}_i) = \boldsymbol{x}_i} s(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}$). The first term is maximized by matching p to M. This is possible while also making $\boldsymbol{x}$ independent of $\boldsymbol{z}$ because $\boldsymbol{\epsilon}$ has the same dimension as the data and f is arbitrary. The second term is maximized by setting q to the prior p. This means the posterior is the prior. This model also does not recover the hidden structure. The use of unrestricted transformations of noise $\boldsymbol{\epsilon}_i$ to produce the data makes the hidden structure independent of the data.

To get models with meaningful posterior distributions (to recover hidden structure), the dimension of randomness should exceed the data dimension and the noise's role in generation must be limited. Note that a similar argument applies to justify the use of multiple levels of latent variables.

Many models are DEFs. Many latent variable models in the literature are deep exponential families. For example, a Gaussian mixture model is a one layer DEF with a single

multinomial layer,

$$\boldsymbol{z}_i \sim \text{Multinomial}(\pi)$$
$$\boldsymbol{x}_i \sim \text{Gaussian}(W\boldsymbol{z}_i, \sigma).$$

Similarly, Poisson factorization (Canny, 2004; Gopalan et al., 2015), factorial mixture models (Ghahramani, 1995), and probabilistic matrix factorization (Salakhutdinov and Mnih, 2008) are all one layer DEFS with varying priors on $\boldsymbol{z}_i$. This class includes Bayesian factor analysis methods such as exponential family principle component analysis Mohamed et al. (2008) and multinomial principle component analysis (Buntine and Jakulin, 2004). The independent development of these related models partially stemmed from the need for model-specific inference.

For multilayer examples, the correlated topic model (Blei and Lafferty, 2006a) is a multilayer DEF, where the highest layer is a multivariate normal and the subsequent layer is multinomial. Deep latent Gaussian models (Rezende et al., 2014) are a multilayer DEF where each layer is Gaussian.

Related work. Graphical models and neural nets have a long and distinguished history. We highlight some key results as they relate to DEFS. More generally, deep exponential families fall into the broad class of stochastic feedforward belief networks (Neal, 1990), but Neal (1990) focuses mainly on one example in this class, the sigmoid belief network, which is a binary latent variable model.

Undirected graphical models have also been used for inferring compositional hierarchies. Salakhutdinov and Hinton (2009) propose deep probabilistic models based on restricted Boltzmann machines (RBMS) (Smolenksy, 1986). RBMS are a two layer undirected probabilistic model with one layer of latent variables and one layer of observations tied together

by a weight matrix. Directed models such as DEFS have the "explaining away" property, where independent latent variables under the prior become dependent, conditional on the observations. This property makes inference harder than in RBMS, but creates a more parsimonious representation where similar features compete to explain the data rather than work in tandem (Goodfellow et al., 2012; Bengio et al., 2013).

RBMS have been extended to general exponential family conditionals in a model called exponential family harmoniums (Welling et al., 2005). A certain infinite DEF with tied weights is equivalent to an exponential family harmonium (Hinton et al., 2006), but as our weights are not tied, deep exponential families represent a broader class of models than exponential family harmoniums (and RBMS).

The literature of latent variable models relates to DEFS through both hierarchical models and Bayesian factor analysis. Finally, latent tree hierarchies have been constructed with Dirchlet distributions (Li and McCallum, 2006).

4.4 Inference

The main computational problem for working with deep exponential families is computing the posterior distribution of the latent units and the weights shared across data. Traditionally, for each DEF, we would need to develop variational approximation algorithms individually (see for examples the models in the literature that are DEFS from the previous section). Each of these would be an individual research project. However, with BBVI we only need to specify the approximating families that meet the black box criteria (sampling from the approximation, computing score function and log density). We choose the mean

field family

$$q(\boldsymbol{z}, \boldsymbol{W}) = q(\boldsymbol{W}_0) \prod_{\ell=1}^{L-1} q(\boldsymbol{W}_\ell) \prod_{n=1}^{N} q(\boldsymbol{z}_{n,\ell}),$$

where each component comes from the same exponential family as in the model. We transform all positively constrained variational parameters to the reals using the inverse softplus function. Similarly, we use the inverse sigmoid function for variables on the unit interval.[3]

We use the Rao-Blackwellized gradient with control variates (Equation (3.5)) and data subsampling. Stochastic optimization requires a learning rate to scale the noisy gradients. We use RMSProp (Tieleman and Hinton, 2012). RMSProp captures both varying length scales and noise by normalizing each dimension by the square root of a running average of the square of that dimension. It tends to be more robust than AdaGrad for problems with high variance gradients early in optimization. Code is available at https://github.com/blei-lab/deep-exponential-families.

4.5 Experiments

We perform extensive evaluations of DEFs. We provide predictive results from 28 different DEF instances. We explore the number of layers (1, 2, or 3), latent variable distributions (gamma, Poisson, Bernoulli), link functions (log, log-softplus), and weight distributions (normal, gamma) using a Poisson observation model. We also show how two DEFs can be composed to build a model for pairwise data.

[3]Due to numerical precision, it is possible to samples zeros from a gamma distribution. To address this, we add small tolerances for the shape, scale (inverse rate), and sampled value.

4.5.1 Text Modeling

We study two large text copora, *Science* and *the New York Times*. *The New York Times* consists of 166K documents and 8K terms, and *Science* consists of 133K documents and 5.9K terms.

Baselines. We consider two baselines. The first is latent Dirichlet allocation (Blei et al., 2003), a popular topic model. The second is a state-of-the-art DocNADE model (Larochelle and Lauly, 2012). DocNADE models the probability of the word in a document given the previous words. The generative structure is shared between each observation. Finally, the one layer sparse gamma DEF is equivalent to Poisson factorization (Canny, 2004; Gopalan et al., 2015), but our approach is fully Bayesian and does not include auxiliary variables.

Evaluation. On a held-out set of 1,000 documents, we compute perplexity. Perplexity measures the amount of randomness in the prediction. Lower numbers are better. Held-out perplexity is

$$\exp\left(\frac{-\sum_{d\in\text{docs}}\sum_{w\in\text{held-out in }d}\log p(w\mid \#\text{ held-out in }d)}{N_{\text{held-out words}}}\right).$$

When we condition on the number of held-out words, the Poisson likelihood becomes a multinomial with mean equal to the normalized Poisson rates. We set the rates to be equal to the expected value from the variational approximation. To estimate per-document parameters, we let all methods see ten percent of each held-out document; the other ninety percent is used in the perplexity calculation. In DEFS, the ten percent is used to compute the document specific variational approximation. For DocNADE, it is used as the conditioning set to predict the rest of the document.

Architectures and hyperparameters. We build one, two, and three layer DEFS with sparse gamma layers, Bernoulli layers, Poisson layers, and Poisson layers with log-link. The sizes of the layers are 100, 30, and 15, respectively. A size of one hundred factors falls into the range of topics searched in the topic modeling literature (Blei and Lafferty, 2007). We set the RMSProp scaling to 0.2 and use a window size of 10 for all of our experiments. We use a batch size of 10,000.

We use the same hyperparameters on gamma distributions on each layer with shape and rate 0.3. For the sigmoid belief network we use a prior of 0.1 to achieve some sparsity. We fix the Poisson prior rate to be 0.1. For gamma $\boldsymbol{W}$, we use shape 0.1 and rate 0.3. For Gaussian $\boldsymbol{W}$, we use a prior mean of 0.0 and variance of 1.0. We let the experiments run for 10,000 iterations at which point the validation likelihood is stable.

We observe two phases of convergence: DEFS converge quickly to a good held-out perplexity (around 2,000 iterations) and then slowly improve until final convergence (around 10,000 iterations).

Results. Table 4.1 summarizes the predictive results on both corpora. DEFS outperform the baselines on both datasets. Furthermore, moving beyond one layer models generally improves performance. The table also reveals that stacking layers of gamma latent variables always leads to similar or better performance.

Finally, as shown by the Poisson DEFS with different link functions, we find gamma distributed weights to outperform normally distributed weights. Somewhat related, we find sigmoid DEFS (with normal weights) to be more difficult to infer, with the deeper version performing poorly.

Model	$p(\boldsymbol{W})$	*NYT*	*Science*
LDA (Blei et al., 2003)		2717	1711
DocNADE (Larochelle and Lauly, 2012)		2496	1725
Sparse Gamma 100	$\emptyset$	2525	1652
Sparse Gamma 100-30	Γ	2303	1539
Sparse Gamma 100-30-15	Γ	**2251**	**1542**
Sigmoid 100	$\emptyset$	2343	1633
Sigmoid 100-30	$\mathcal{N}$	2653	1665
Sigmoid 100-30-15	$\mathcal{N}$	2507	1653
Poisson 100	$\emptyset$	2590	1620
Poisson log-softplus 100-30	$\mathcal{N}$	2423	1560
Poisson log-softplus 100-30-15	$\mathcal{N}$	2416	1576
Poisson log-link 100-30	Γ	2288	**1523**
Poisson log-link 100-30-15	Γ	2366	1545

Table 4.1: Perplexity on a held-out collection of 1K *Science* and *NYT* documents. Lower values are better. The p($\boldsymbol{W}$) column indicates the type of prior distribution over the DEF weights, Γ for the gamma and $\mathcal{N}$ for normal (recall that one layer DEFS consist only of a layer of latent variables, thus we represent their prior with the $\emptyset$). All multilayer Poisson and sparse gamma DEFS outperform the baselines.

4.5.2 Composing DEFs

Previously, we constructed models out of a single DEF, but DEFS can be embedded and composed in more complex models. We describe a model for pairwise data that uses multiple DEFS and one for survival analysis.

Matrix Factorization

We now present *double DEFs*, a factorization model for pairwise data where both the rows and columns are determined by DEFS. At a high level the double DEF corresponds to replacing $\boldsymbol{W}_0$ in the likelihood with another DEF.

Matrices can be used to represent pairwise data. The pairwise data we consider include (user, item) ratings and (user, article) clicks. The observed data are counts. In a double

	Model	Perplexity	NDCG
Netflix	Gaussian MF (Salakhutdinov and Mnih, 2008)	–	0.008
	1 layer Double DEF	2319	0.031
	2 layer Double DEF	2299	0.022
	3 layer Double DEF	2296	0.037
ArXiv	Gaussian MF	–	0.013
	1 layer Double DEF	2138	0.049
	2 layer Double DEF	1893	0.050
	3 layer Double DEF	1940	0.053

Table 4.2: Comparison of matrix factorization methods on Netflix and the ArXiv. We find that deep double DEFs outperform the shallow ones on perplexity. We also find that the NDCG of low-activity users (users with less than 5 and 10 observations in the observed 10% of the held-out set respectively for Netflix and ArXiv). We use Vowpal Wabbit's matrix factorization implementation which does not readily provide held-out likelihoods and thus we do not report the perplexity associated with matrix factorization.

DEF, we model the counts with a Poisson conditional on a DEF for rows (denoted with the superscript r) and a DEF for columns (denoted with the superscript c):

$$p(\boldsymbol{x}_{n,i} \mid \boldsymbol{z}^c_{n,1}, \boldsymbol{z}^r_{i,1}).$$

For example, for rating data, the double DEF instantiates a DEF for both rows (items) and columns (users). The rating for a (item, user) pair comes from the inner product of the lowest layer of the DEF of the nth user $\boldsymbol{z}^c_{n,1}$ and the lowest layer of the DEF for the ith item $\boldsymbol{z}^r_{i,1}$.

We infer double DEFs on *Netflix* ratings and click data from the *arXiv* (www.arXiv.org) which indicates how many times a user clicked on a paper. Our *Netflix* collection consists of 50K users and 17.7K movies. The movie ratings range from zero to five stars, where zero means the movie was unrated by the user. The *ArXiv* collection consists of 18K users and 20K documents. We fit a one, two, and three layer double DEF where the layer sizes of the row DEF match the layer sizes of the column DEF.

The sizes of the layers are 100, 30, and 15. We compare double DEFS to $l2$-regularized (Gaussian) matrix factorization (MF) (Salakhutdinov and Mnih, 2008). We reuse the testing procedure introduced in the previous section (this is referred to as *strong generalization* in the recommendation literature (Marlin, 2004)) where the held-out test set contains one thousand users. For performance and computational reasons we subsample zero-observations for MF, as is standard (Gopalan et al., 2015). We also report the commonly-used multi-level ranking measure (untruncated) NDCG (Järvelin and Kekäläinen, 2000) for all methods.

Table 4.2 shows that two-layer DEFS improve performance over the shallow DEF and that all DEFS outperform Gaussian MF. On perplexity the three layer model performs similarly for Netflix and slightly worse for the ArXiv. The table highlights that when comparing ranking performance on low-activity users, a data regime of particular importance for practical recommender systems, multilayer DEFS help. Learning more complex priors help when working with small data.

Deep Survival Analysis

Survival analysis studies the time to an event. For example, it could be the time to retirement, the time to machine failure, or the time to dialysis from the onset of kidney disease. Estimating survival is a core problem in healthcare. But health data, especially from electronic health records, has many missing values. Furthermore, the relationships between different observations such as age and hormone levels tend to be complex. At its core, survival analysis is a density estimation problem. Deep exponential families can be used to build densities that admit complex relationships between observations and can handle missing data.

If for each patient i we get survival time, covariate pairs $(\boldsymbol{t}_i, \boldsymbol{x}_i)$, we can build a surival model as

$$
\begin{aligned}
\boldsymbol{z}_n &\sim \text{DEF} \\
\boldsymbol{x}_n &\sim p(\cdot \mid \boldsymbol{W}_{\boldsymbol{x}}, \boldsymbol{z}_n) \\
t_n &\sim \text{Weibull}(\log(1 + \exp(f(\boldsymbol{z}_n; \boldsymbol{W}_t), k))).
\end{aligned}
$$

Both the likelihood for the survival time and the covariates can contain nonlinear transformations of the draw $\boldsymbol{z}_n$ from the DEF. This transformation is explicitly denoted by f for the survival time.

Data. We study deep survival analysis for coronary heart disease (CHD) on a collection of 313,000 patients from a large metropolitan hospital. The patient population included all adults (>18 years old) that have at least 5 months (not necessarily consecutive) where at least one observation was recorded. The patient records contain documentation resulting from all settings, including inpatient, outpatient, and emergency department visits. Observations include 9 vital signs, 79 laboratory test measurements, 5,262 medication orders, and 13,153 diagnosis codes.

All real-valued measurements and discrete variables were aggregated at the month level, leading to binned observations for each patient. For each continuous measurement, such as vitals or laboratory values, we set the value of each bin to the average value over the measurements from that month. The presence of discrete elements such as medication orders and diagnosis codes was encoded as a binary variable. We use Student's t-distributions as likelihoods to model the observed labs and vitals, and we use Bernoulli likelihoods that admit computation in time proportional to the number of non-zero Bernoulli entries (see Ranganath et al. (2015a)) for diagnoses and medications.

We set the shape of the Weibull to be two. The exponential family used inside the DEF is a Gaussian. The mean and inverse softplus variance functions for each layer are a two layer perceptron with rectified linear activations. We set all normal priors to have mean zero and variance one.

Baselines. Of the 313,000 patients in the study, 263,000 were randomly selected for training, 25,000 for validation, and 25,000 for testing. We assess convergence with the validation cohort and evaluate concordance[4] on the test cohort. A coronary heart disease (CHD) event is defined as the documentation of any ICD-9 diagnosis code with the following prefixes: 413 (angina pectoris), 410 (myocardial infarction), or 411 (coronary insufficiency). In our experiments, we vary the dimensionality K of $\boldsymbol{z}_n$ to assume the values of $\{5, 10, 25, 75, 100\}$. The layer size for the perceptrons was set equal to the dimensionality of $\boldsymbol{z}_n$ in each experiment. We evaluate both the baseline risk score and deep survival analysis with concordance (Harrell et al., 1982).

While concordance enables the comparison of deep survival analysis to the baseline, it only roughly captures the accuracy of the temporal prediction of the models. In deep survival analysis, we can compute the predictive likelihood on the held-out set according to the model. The predictive likelihood captures how well the model predicts failure in time. It is the expected log probability of the observed time until failure, conditioned on the observed covariates for a given patient in a given month.

We use the Framingham CHD risk score that is used in clinical practice as the baseline. It was developed in 1998 and is one of the earliest validated clinical risk scores. It is a gender-stratified algorithm for estimating the 10-year coronary heart disease risk of an individual. Aside from gender, this score takes into consideration age, sex, LDL cholesterol, HDL cholesterol, blood pressure, diabetes, and smoking. For example, a 43-year-old (1

[4]Concordance measures the fraction of correctly ordered survival times. All pairs may not be comparable due to censoring.

point) male patient with an LDL level of 170 mg/dl (1 point), an HDL level of 43 mg/dl (1 point), a blood pressure of 140/90 (2 points), and no history of diabetes (0 points) or smoking (0 points) would have a risk score of 5, which would correspond to a 10 year CHD risk of 9% (Wilson et al., 1998). The score was validated using curated data from the Framingham Heart Study.

Model	Concordance (%)
Baseline Framingham Risk Score	**65.57**
Deep Survival Analysis; K=10	69.35
Deep Survival Analysis; K=5	70.45
Deep Survival Analysis; K=25	71.20
Deep Survival Analysis; K=75	71.65
Deep Survival Analysis; K=100	72.71
Deep Survival Analysis; K=50	**73.11**

Table 4.3: Concordance on a held-out set of 25,000 patients for different values of K and for the baseline risk score. All deep survival analysis dimensionalities outperform the baseline.

Model performance and predictive likelihood. The baseline CHD risk score yielded 65.57% in concordance on the held-out test set. Table 4.3 shows the concordance of the deep survival analysis for different values of K. When considering full deep survival with all data types considered, the best performance was obtained for $K = 50$.

When examining the deep survival analysis with the best concordance on the held-out set ($K = 50$), we asked how well each individual data type predicts failure. All models included age and gender, and predictive likelihoods were computed on the same month bins, even in the absence of observations of a specific data type. Table 4.4 contains the results for the four data types. The diagnoses only model yielded the best predictive likelihood.

Lastly, we studied models with multilayer nonlinear likelihoods. We found little difference in concordance, but noticeably improved predictive likelihoods. For problems where the absolute time is of interest rather than the relative time across patients (e.g., cancer versus organ transplantation), the multilayer nonlinear likelihoods make a difference.

Data Type	Likelihood
Medications	-1.24899
Laboratory Tests	-0.998774
Vitals	-0.961827
Diagnoses	-0.855385

Table 4.4: Predictive likelihood of deep survival analysis ($K = 50$) for individual data types. The diagnoses model performs best.

4.6 Conclusion

In this chapter, we developed deep exponential families. DEFS describe hierarchical relationships of data and latent variables to capture compositional semantics of data. We presented several examples of DEFS and showed that many models in the literature are also examples of DEFS. DEFS achieve improved predictive power and provide interpretable semantic structures. We used deep exponential families to build survival models for medical events using health records.

Deep exponential families were made possible by BBVI. In the subsequent chapter, we return to inference and explore new kinds of variational approximations enabled by black box variational inference.

Chapter 5

Hierarchical Variational Models

The implementations of variational inference in the previous chapters use the mean field family. In the mean field family, each latent variable is independent and governed by its own parameters. Using the mean field family in conditionally conjugate models leads to simple closed form updates. This property is one of the original motivations for using mean field family approximations.

With black box variational inference this motivation wanes. In the previous chapter, we explored models that were nonconjugate—even with mean field approximations, we do not get closed form updates. Though the mean field family enables efficient inference, it is limited by its strong factorization. It cannot capture posterior dependencies between latent variables, dependencies which both improve the fidelity of the approximation and are sometimes of intrinsic interest. By breaking the requirement that the variational approximation should be chosen to yield closed form updates, BBVI makes it possible to consider new families of variational approximations.

With this motivation, we develop hierarchical variational models (HVMS), a class of variational approximations that goes beyond the mean field family and beyond directly param-

	Data Models	Variational Models
Model	$p(\boldsymbol{x})$	$q(\boldsymbol{z})$
Target	F	$p(\boldsymbol{z} \mid \boldsymbol{x})$
Information	$\boldsymbol{x} \sim F$	$p(\boldsymbol{x}, \boldsymbol{z})$

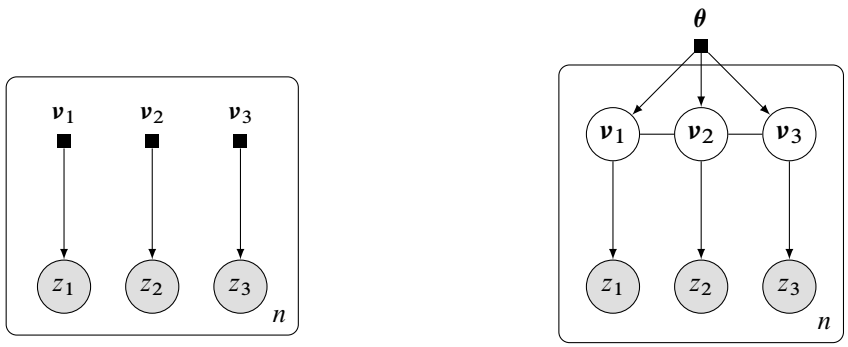
Table 5.1: Models for data and variational approximations are similar. Both try to match distributions. The difference lies in the source of information. In data models, the samples provide information about the target; in variational inference, the unnormalized posterior provides the information.

eterized variational families. The main idea behind our method is to treat the variational family as a model of the latent variables and then to expand this model hierarchically. Just as hierarchical Bayesian models induce dependencies between data, hierarchical variational models induce dependencies between latent variables.

5.1 Hierarchical Variational Models

Our central idea is to draw an analogy between probability models of data and variational distributions (models) of latent variables. A probability model defines a collection of distributions over data; the size of the collection depends on the model's complexity. In the same way, a variational approximating family defines a collection of distributions over latent variables; the size of the collection depends on the variational family's complexity.

A probability model of data works well if it looks like the data generating distribution; a variational model works well if it looks like the posterior distribution. This similarity also reveals the key difference. In modeling, samples (via the observed data) provide information about the data generating distribution F, whose density is unknown. In posterior approximation, the unnormalized posterior provides information about the posterior distribution, while samples from the posterior are hard to get. Table 5.1 summarizes these similarities and differences.



(a) MEAN FIELD MODEL **(b)** HIERARCHICAL MODEL

Figure 5.1: Graphical model representation. **(a)** In mean field models, the latent variables are strictly independent. **(b)** In hierarchical variational models, the latent variables are governed by a prior distribution on their parameters, which induces complex dependence.

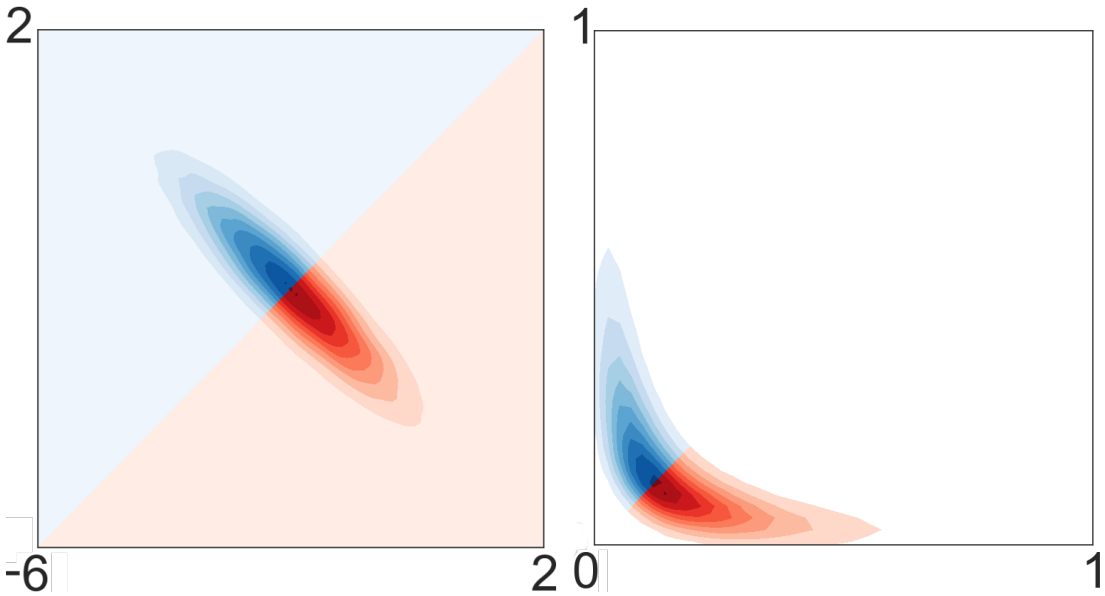
5.1.1 Hierarchical Variational Models

One common approach to expanding the complexity, especially in Bayesian statistics, is to expand a model hierarchically, i.e., by placing a prior on the parameters of the likelihood. Expanding a model hierarchically has distinct advantages. It induces new dependencies between the data, either through shrinkage or an explicitly correlated prior (Efron, 2012), and it enables us to reuse algorithms for the simpler model within algorithms for the richer model (Gelman and Hill, 2007).

We use the same idea to expand the complexity of the mean field variational family and to construct *hierarchical variational models* (HVMS). First, we view the mean field family,

$$q_{\text{MF}}(\boldsymbol{z}; \boldsymbol{\nu}) = \prod_{i=1}^{d} q(\boldsymbol{z}_i; \boldsymbol{\nu}_i), \tag{5.1}$$

as a simple model of the latent variables. Next, we expand it hierarchically. We introduce a "variational prior" $q(\boldsymbol{\nu}; \boldsymbol{\theta})$ with "variational hyperparameters" $\boldsymbol{\theta}$ and place it on the mean field model (a type of "variational likelihood"). Marginalizing out the prior gives



(a) Normal$(\boldsymbol{\nu}; \boldsymbol{\theta})$ **(b)** $\prod_{i=1}^{2}$ Gamma$(z_i \mid \boldsymbol{\nu}_i)$

Figure 5.2: An HVM with mean field gamma likelihood and a multivariate Gaussian variational prior. **(a)** $q(\boldsymbol{\nu}; \boldsymbol{\theta})$: The (reparameterized) natural parameters assume a multivariate prior, with different areas indicated by red and blue. **(b)** $\prod_{i=1}^{2} q(z_i \mid \boldsymbol{\nu}_i)$: The latent variables are drawn from a mean field family, colorized according to the drawn parameters from the multivariate prior. The covariance of the drawn variational parameters induces dependence among the various dimensions

$q_{\text{HVM}}(\boldsymbol{z}; \boldsymbol{\theta})$, a hierarchical family of distributions over the latent variables

$$q_{\text{HVM}}(\boldsymbol{z}; \boldsymbol{\theta}) = \int q(\boldsymbol{\nu}; \boldsymbol{\theta}) \prod_i q(\boldsymbol{z}_i \mid \boldsymbol{\nu}_i) d\boldsymbol{\nu}. \tag{5.2}$$

This family enjoys the advantages of hierarchical modeling in the context of variational inference: it induces dependence among the latent variables and allows us to reuse simpler computation when fitting the more complex family.

Figure 5.1 illustrates the difference between the mean field family and HVMs. Mean field variational inference fits the variational parameters $\boldsymbol{\nu}$ so that the factorized distribution is close to the exact posterior; this ignores posterior correlation. Using the same principle, an HVM fits the variational hyperparameters so $q_{\text{HVM}}(\boldsymbol{z}; \boldsymbol{\theta})$ is close to the exact posterior. These hyperparameters control the variational prior, which induces dependence between the dimensions of $\boldsymbol{z}$. Thus HVMs can capture posterior dependence.

Figure 5.2 presents a simple example. The variational family posits that each z_i is a scalar from an exponential family. The variational parameters $\boldsymbol{\nu}_i$ correspond to the natural parameters transformed to the real line (for example the log transform for positively constrained parameters). Now place a multivariate Gaussian prior on the mean field parameters, with a full covariance matrix. The resulting HVM is a two-level distribution: first draw the set of mean field variational parameters $\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_d$ from a Gaussian (Figure 5.2a); then draw each z_i given its corresponding parameter (Figure 5.2b). The covariance of the drawn variational parameters induces dependence among the various dimensions.

If an HVM can express the same marginals as the mean field approximation, then $q_{\text{HVM}}(\boldsymbol{z}; \boldsymbol{\theta})$ is more expressive than the mean field family. As in the example, an HVM induces dependences among the variables and expands the family of possible marginals that it can capture. The number of parameters in a distribution defines the number of free moments of that distribution, so the added flexibility requires the dimension of $\boldsymbol{\theta}$ to exceed $\boldsymbol{\nu}$'s.

In the next section, we develop a black box algorithm for HVMs. It exploits the mean field structure of the variational likelihood and enjoys the corresponding computation advantages. We first discuss how to specify an HVM.

5.1.2 Specifying an HVM

We can construct an HVM by placing a prior on any existing variational approximation. An HVM has two components: the variational likelihood $q(\boldsymbol{z} \mid \boldsymbol{\nu})$ and the prior $q(\boldsymbol{\nu}; \boldsymbol{\theta})$. The likelihood comes from an existing variational family that meets the black box criteria, such as the mean field or the sequential approximations used in temporal modeling (Blei and Lafferty, 2006b). We focus on the mean field family as the likelihood. For the prior, the distribution of $\{\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_d\}$ should not have the same factorization structure as the variational likelihood—otherwise it will not induce dependence between latent variables. In the

previous section, we saw one example of a variational prior, the multivariate Gaussian. We outline more examples of variational priors.

Variational prior: mixture of Gaussians. One option for a variational prior is to assume the mean field parameters $\boldsymbol{\nu}$ are drawn from a mixture of Gaussians. Let K be the number of components, $\boldsymbol{\pi}$ be a probability vector, and $\boldsymbol{\mu}_k$ and Σ_k be the parameters of a d-dimensional multivariate Gaussian. The variational prior is

$$q(\boldsymbol{\nu}; \boldsymbol{\theta}) = \sum_{i=1}^{K} \boldsymbol{\pi}_k \text{Normal}(\boldsymbol{\mu}_k, \Sigma_k).$$

The parameters $\boldsymbol{\theta}$ contain the probability vector $\boldsymbol{\pi}$ as well as the component means $\boldsymbol{\mu}_k$ and variances Σ_k. The mixture locations $\boldsymbol{\mu}_k$ capture relationships between different latent variables. For example, a two-component mixture with two latent variables (and a mean field variational likelihood) can capture that the latent variables are either very positive or very negative.

Given enough components, mixtures can approximate arbitrary distributions, and have been considered as variational families (Jaakkola and Jordan, 1998; Lawrence, 2000; Gershman and Blei, 2012; Salimans et al., 2013). In the traditional setup, however, the mixtures form the variational approximation on the latent variables directly. Here we use it on the variational parameters; this lets us use a mixture of Gaussians as the variational approximation in many models, including those with discrete latent variables.

Variational prior: normalizing flows. Mixtures offer flexible variational priors. However, in the algorithms we derive, the number of model likelihood evaluations scales with the number of mixture components. In high dimensions, the number of mixture components needed to accurately approximate the posterior can be impractical. We seek a prior

whose computational complexity scales better with its modeling flexibility. This motivates normalizing flows.

Normalizing flows are variational approximations for probability models with differentiable densities (Rezende and Mohamed, 2015) that meet the black box criteria. Normalizing flows build a parameterized probability distribution by transforming a simple random variable $\boldsymbol{\nu}_0$ through a sequence of invertible differentiable functions f_1 to f_K. Each function transforms its input, so the distribution of the output is a complex warping of the original random variable $\boldsymbol{\nu}_0$.

We can use normalizing flows as a variational prior. Let $\boldsymbol{\nu}_k = f_k \circ ... \circ f_1(\boldsymbol{\nu}_0)$; then the flow's density is

$$q(\boldsymbol{\nu}; \boldsymbol{\theta}) = q(\boldsymbol{\nu}_0) \prod_{k=1}^{K} \left| \det \left(\frac{\partial f_k}{\partial \boldsymbol{\nu}_{k-1}} \right) \right|^{-1}.$$

With the normalizing flow prior, the latent variables become dependent because their variational parameters are deterministic functions of the same random variable. General determinant computations are cubic in the dimensionality of the transformation. This can be computationally prohibitive. To avoid this, the transformations f_k should be chosen to have determinants computable in linear time. One general such way to achieve this is to transform each dimension d sequentially:

$$\boldsymbol{\nu}_{k+1}^d = f_{k+1}^d(\boldsymbol{\nu}_k^1, \cdots, \boldsymbol{\nu}_k^d),$$

which has a lower-triangular Jacobian. (See for example Kingma et al. (2016)).

The HVM expands the use of normalizing flows to non-differentiable latent variables, such as those with discrete, ordinal, and discontinuous support. In the experiments, we use normalizing flows to better approximate posteriors of discrete latent variables.

Other variational models. Many modeling tools can be brought to bear on building hierarchical variational models. Copulas explicitly introduce dependence among d random variables by using joint distributions on d-dimensional hypercubes (Nelsen, 2006). HVMs can use copulas as priors on either point mass or general mean field likelihoods. As another example, we can replace the mixture model prior with a factorial mixture (Ghahramani, 1995). This leads to a richer posterior approximation.

Note that there is a tradeoff at play in building variational models. On one hand, the randomness of the variational likelihood smooths out the inference problem and makes for easier optimization. On the other hand, variational likelihoods that approximate randomness-free delta distributions lead to arbitrary approximations. In the appendix of this chapter we connect HVMs to empirical Bayes and reinforcement learning, and consider multilevel hierarchical priors.

5.2 Optimizing HVMs

We derive a black box variational inference algorithm for a large class of probability models that use any hierarchical variational model as the posterior approximation. Our algorithm enables efficient inference by preserving both the computational complexity and variance properties of the stochastic gradients of the variational likelihood.

5.2.1 Hierarchical Evidence Lower Bound

We optimize over the parameters $\boldsymbol{\theta}$ of the variational prior to find the optimal distribution within the class of hierarchical variational models. Using the HVM, the ELBO is

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{q_{\text{HVM}}(\boldsymbol{z};\boldsymbol{\theta})}[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\text{HVM}}(\boldsymbol{z}; \boldsymbol{\theta})]. \tag{5.3}$$

The expectation of the first term is tractable as long as we can sample from q, and q has proper support. The expectation of the second term is the entropy. It contains an integral (Equation (5.2)) with respect to the variational prior, which is analytically intractable in general.

Tractable bound on the entropy. Deriving an analytic expression for the entropy of q_{HVM} is generally intractable due to the integral in the definition of q_{HVM}. However, it is tractable when we know the distribution $q(\boldsymbol{\nu} \mid \boldsymbol{z})$. This can be seen by noting from standard Bayes' rule that

$$q(\boldsymbol{z})q(\boldsymbol{\nu} \mid \boldsymbol{z}) = q(\boldsymbol{\nu})q(\boldsymbol{z} \mid \boldsymbol{\nu}), \tag{5.4}$$

and that the right hand side is specified by the construction of the hierarchical variational model. The distribution $q(\boldsymbol{\nu} \mid \boldsymbol{z})$ can be interpreted as the posterior distribution of the original variational parameters $\boldsymbol{\nu}$ given the latent variables, thus we will denote it as $q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z})$.

In general, computing $q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z})$ from the specification of the hierarchical variational model is as hard as the integral needed to compute the entropy. We instead approximate q_{POST} with an auxiliary distribution $r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})$ parameterized by $\boldsymbol{\phi}$.[1] This yields a bound on the entropy in terms of the analytically known distributions $r(\boldsymbol{\nu} \mid \boldsymbol{z})$, $q(\boldsymbol{z} \mid \boldsymbol{\nu})$, and $q(\boldsymbol{\nu})$.

First note that the KL divergence between two distributions is greater than zero and is precisely zero only when the two distributions are equal. This means the entropy can be

[1] Technically, all distributions including r condition on $\boldsymbol{x}$. We suppress this notation for compactness.

bounded as follows:

$$
\begin{aligned}
&-\mathbb{E}_{q_{\text{HVM}}}[\log q_{\text{HVM}}(\boldsymbol{z})] \\
&\quad = -\mathbb{E}_{q_{\text{HVM}}}[\log q_{\text{HVM}}(\boldsymbol{z}) + \text{KL}(q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z}) || q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z}))] \\
&\quad \geq -\mathbb{E}_{q_{\text{HVM}}}[\log q_{\text{HVM}}(\boldsymbol{z}) + \text{KL}(q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z}) || r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi}))]] \\
&\quad = -\mathbb{E}_{q_{\text{HVM}}}[\mathbb{E}_{q_{\text{POST}}}[\log q_{\text{HVM}}(\boldsymbol{z}) + \log q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z}) - \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})]] \\
&\quad = -\mathbb{E}_{q(\boldsymbol{z}, \boldsymbol{\nu})}[\log q_{\text{HVM}}(\boldsymbol{z}) + \log q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z}) - \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})].
\end{aligned}
$$

Then by Equation (5.4), the bound simplifies to

$$
\begin{aligned}
&-\mathbb{E}_{q_{\text{HVM}}}[\log q_{\text{HVM}}(\boldsymbol{z})] && (5.5) \\
&\quad \geq -\mathbb{E}_{q(\boldsymbol{z}, \boldsymbol{\nu})}[\log q(\boldsymbol{\nu}) + \log q(\boldsymbol{z} \mid \boldsymbol{\nu}) - \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})]. && (5.6)
\end{aligned}
$$

As in variational inference, the bound is exact when $r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})$ matches the *variational posterior* $q(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\theta})$. From this perspective, we can view r as a recursive variational approximation. It is a model for the posterior q of the mean field parameters $\boldsymbol{\nu}$, given a realization of the latent variables $\boldsymbol{z}$. A similar bound in derived by Salimans et al. (2015) directly for $\log p(x)$.

In the above derivation, the approximation r to the variational posterior $q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z})$ is placed as the second argument of a KL divergence term. Replacing the first argument instead yields

a different tractable upper bound:

$$
\begin{aligned}
&-\mathbb{E}_{q_{\text{HVM}}}[\log q_{\text{HVM}}(\boldsymbol{z})] \\
&\quad = \mathbb{E}_{q_{\text{HVM}}}[-\log q_{\text{HVM}}(\boldsymbol{z}) + \text{KL}(q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z}) || q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z}))] \\
&\quad \leq \mathbb{E}_{q_{\text{HVM}}}[-\log q_{\text{HVM}}(\boldsymbol{z}) + \text{KL}(r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi}) || q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z}))] \\
&\quad = \mathbb{E}_{q_{\text{HVM}}}[\mathbb{E}_r[-\log q_{\text{HVM}}(\boldsymbol{z}) - \log q_{\text{POST}}(\boldsymbol{\nu} \mid \boldsymbol{z}) + \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})]] \\
&\quad = \mathbb{E}_{q_{\text{HVM}}}[\mathbb{E}_r[-\log q(\boldsymbol{\nu}) - \log q(\boldsymbol{z} \mid \boldsymbol{\nu}) + \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})]].
\end{aligned}
$$

The bound is also tractable when r and q_{HVM} can be sampled and all distributions are analytic. The derivation of these two bounds parallels the development of expectation propagation (Minka, 2001) and variational Bayes (Jordan, 1999) which are based on alternative forms of the KL divergence. This upper bound combined with the previous lower bound can be used to assess the quality of HVM entropy estimates.

The entropy bound lower bound Equation (5.6) is tighter than the trivial conditional entropy bound of $\mathbb{H}[q_{\text{HVM}}] \geq \mathbb{H}[q \mid \boldsymbol{\nu}]$ (Cover and Thomas, 2012). This bound is attained when specifying the recursive approximation as $r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi}) = q(\boldsymbol{\nu}; \boldsymbol{\theta})$; i.e., it is the special case when when the recursive approximation is the variational prior.

Hierarchical ELBO. Substituting the entropy bound Equation (5.6) into the ELBO gives a tractable lower bound called the *hierarchical* ELBO. It is

$$
\widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{q(\boldsymbol{z}, \boldsymbol{\nu}; \boldsymbol{\theta})}\Big[\log p(\boldsymbol{x}, \boldsymbol{z}) + \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi}) - \sum_{i=1}^{d} \log q(z_i \mid \boldsymbol{\nu}_i) - \log q(\boldsymbol{\nu}; \boldsymbol{\theta})\Big]. \tag{5.7}
$$

The hierarchical ELBO is tractable, as all of the terms are tractable. We jointly fit q and r by maximizing Equation (5.7) with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. Alternatively, the joint maximization can be interpreted as variational expectation-maximization on an expanded probability

model, $r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})\, p(\boldsymbol{z} \mid \boldsymbol{x})$. In this light, $\boldsymbol{\phi}$ are model parameters and $\boldsymbol{\theta}$ are variational parameters, and the hierarchical ELBO lower bounds the ELBO through the extra use of Jensen's inequality on the new random variable $\boldsymbol{\nu}$. Optimizing $\boldsymbol{\theta}$ improves the posterior approximation; optimizing $\boldsymbol{\phi}$ tightens the bound on the KL divergence by improving the recursive variational approximation.

We can also analyze Equation (5.7) by rewriting it in terms of the mean field ELBO,

$$\widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_q[\mathcal{L}_{\text{MF}}(\boldsymbol{\nu})] + \mathbb{E}_q[\log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi}) - \log q(\boldsymbol{\nu}; \boldsymbol{\theta})],$$

where $\mathcal{L}_{\text{MF}} = \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z} \mid \boldsymbol{\nu})]$. This shows that Equation (5.7) is a sum of two terms: a Bayesian model average of the ELBO of the variational likelihood, with weights given by the variational prior $q(\boldsymbol{\nu}; \boldsymbol{\theta})$; and a correction term that is a function of both the auxiliary distribution r and the variational prior. Since mixtures (convex combinations) cannot be bigger than their components, r must not be independent of $\boldsymbol{z}$ in order for this bound to be better than the original bound.

5.2.2 Stochastic Gradient of the Hierarchical ELBO

To optimize the hierarchical ELBO, we need to compute the stochastic gradient with respect to the variational hyperparameters $\boldsymbol{\theta}$ and auxiliary parameters $\boldsymbol{\phi}$. Most common probability models are differentiable with respect to their parameters. As long as we specify the variational prior $q(\boldsymbol{\nu}; \boldsymbol{\theta})$ to be differentiable, we can apply the reparameterization gradient for the random variational parameters $\boldsymbol{\nu}$. Recall that reparameterization implies $\boldsymbol{\nu}$ can be written as a function of $\boldsymbol{\theta}$ and noise $\boldsymbol{\epsilon}$ drawn from a distribution such as the standard normal.

Define the score function

$$\boldsymbol{c} = \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z} \mid \boldsymbol{\nu}).$$

The gradient of the hierarchical ELBO with respect to $\boldsymbol{\theta}$ is

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \widetilde{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\nu}} \mathcal{L}_{\text{MF}}(\boldsymbol{\nu})] \\
&\quad + \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\nu}} [\log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi}) - \log q(\boldsymbol{\nu}; \boldsymbol{\theta})]] \\
&\quad + \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\boldsymbol{c} \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})]]. \qquad (5.8)
\end{aligned}$$

We derive this in the appendix. The first term is the gradient of the original variational approximation scaled by the chain rule from the reparameterization. Thus, hierarchical variational models inherit properties from the original variational approximation. Here HVMS get variance reduced gradients Equation (3.8) from the mean field factorization. The second and third terms try to match r and q. The second term uses reparameterization, so it generally exhibits low variance.

The third term potentially involves a high variance gradient due to the appearance of both the score function and all of the latent variables. Since the distribution $q(\boldsymbol{z} \mid \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}))$ factorizes by definition, we can apply the same variance reduction for r as for the mean field model in Chapter 3 to mitigate this. We examine this issue below.

5.2.3 Choosing r

The practicality of HVMS hinges on the variance of the stochastic gradients during optimization. Specifically, any additional variance introduced by r needs to be minimal. Let r_i be the terms $\log r(\boldsymbol{\nu} \mid \boldsymbol{z})$ containing z_i, and let $\boldsymbol{c}_i = \nabla_{\boldsymbol{\nu}_i} \log q(\boldsymbol{z}_i \mid \boldsymbol{\nu}_i)$. Then the last term in

Equation (5.8) can be rewritten as

$$\begin{aligned}
&\mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[V \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})]] \\
&= \mathbb{E}_{s(\boldsymbol{\epsilon})}\left[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}\left[\sum_{i=1}^{d} \boldsymbol{c}_i \log r_i(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})\right]\right].
\end{aligned}$$

We derive this expression in the appendix. When r_i does not depend on many variables, this gradient combines the computational efficiency of the mean field with reparameterization, enabling fast inference for discrete and continuous latent variable models. This expression also gives us the criteria for building an r that admits efficient stochastic gradients: r should be differentiable with respect to $\boldsymbol{\nu}$, flexible enough to model the variational posterior $q(\boldsymbol{\nu} \mid \boldsymbol{z})$, and factorize with respect to its dependence on each z_i.

One way to satisfy these criteria is by first defining r to be a deterministic transformation. Similar to normalizing flows, the deterministic transformation from $\boldsymbol{\nu}_0$ to $\boldsymbol{\nu}$ can be a sequence of invertible, differentiable functions g_1 to g_k. However unlike normalizing flows, we let the inverse functions g^{-1} have a known parametric form. We call this the *inverse flow*. Under this transformation, the log density of r is

$$\log r(\boldsymbol{\nu} \mid \boldsymbol{z}) = \log r(\boldsymbol{\nu}_0 \mid \boldsymbol{z}) + \sum_{k=1}^{K} \log\left(\left|\det\left(\frac{\partial g_k^{-1}}{\partial \boldsymbol{\nu}_k}\right)\right|\right),$$

where each g_k may depend on $\boldsymbol{z}$. Next the interactions between $\boldsymbol{z}$'s need to be controlled. The simplest way would be to let $\boldsymbol{\nu}_0$ be

$$r(\boldsymbol{\nu}_0 \mid \boldsymbol{z}) = \prod_{i=1}^{d} r(\boldsymbol{\nu}_{0i} \mid \boldsymbol{z}_i), \tag{5.9}$$

and have each g_k be independent of $\boldsymbol{z}$. Another choice comes from looking at the optimal recursive variational family r. The optimal r, denoted by r^*, is the variational posterior,

thus

$$\log r^*(\boldsymbol{z} \mid \boldsymbol{\nu}) = \log q_{\text{POST}} = \sum_{i=1}^{d} \log q(\boldsymbol{z}_i \mid \boldsymbol{\nu}_i) + \log q(\boldsymbol{\nu}; \boldsymbol{\theta}) + C,$$

for some normalization constant C. In the optimal r^*, each $\boldsymbol{\nu}_i$ only interacts locally with $\boldsymbol{z}_i$. This gives a way to construct r. First, transform each $\boldsymbol{\nu}$ using joint transforms with parameters independent of $\boldsymbol{z}$, then transform each $\boldsymbol{\nu}_i$ separately, with functions that depend on $\boldsymbol{z}_i$. Formally,

$$\log r(\boldsymbol{\nu} \mid \boldsymbol{z}) = \log r(\boldsymbol{\nu}_0) + \sum_{k=1}^{K} \log \left(\left| \det \left(\frac{\partial g_k^{-1}}{\partial \boldsymbol{\nu}_k} \right) \right| \right) + \sum_{\ell=K}^{L+K} \sum_{i=1}^{d} \log \left(\left| \det \left(\frac{\partial g_{k,i}^{-1}}{\partial \boldsymbol{\nu}_{\ell,i}} \right) \right| \right).$$

More compactly, to sample from r we would first sample from $\boldsymbol{\nu}_0 \sim r_0$, then transform to get $\boldsymbol{\nu}_k = g_k(\ldots(g_1(\boldsymbol{\nu}_0)))$, then transforms each dimension i independently with the input $\boldsymbol{z}_i$,

$$\boldsymbol{\nu}_{L+K,i} = g_{L+K,i}(\ldots g_{L+K,i}(\boldsymbol{\nu}_{K,i}, \boldsymbol{z}_i)).$$

This choice of r has the same functional form as the optimal r^* and isolates each of the $\boldsymbol{z}_i$ to control variance of the gradient.

For either choice of r, we can quickly compute the sequence of intermediary $\boldsymbol{\nu}$ by applying the known inverse functions—this enables us to quickly evaluate the log density of inverse flows at arbitrary points. This contrasts with normalizing flows, where evaluating the log density of a value (not generated by the flow) requires inversions for each transformation.

These constructions for r meet our criteria. They are differentiable, flexible, and isolate each individual latent variable in a single term. They maintain the locality of mean field inference and are therefore crucial to stochastic optimization.

Algorithm 3: Black box inference with an HVM

Input : Model $\log p(\boldsymbol{x}, \boldsymbol{z})$,
Variational model $q(\boldsymbol{z} \mid \boldsymbol{\nu})q(\boldsymbol{\nu}; \boldsymbol{\theta})$.
Output: Variational Parameters: $\boldsymbol{\theta}$.
Initialize $\boldsymbol{\phi}$ and $\boldsymbol{\nu}$ randomly.
while *not converged* **do**
  Compute unbiased estimate of $\nabla_{\boldsymbol{\theta}} \widetilde{\mathcal{L}}$.
  Compute unbiased estimate of $\nabla_{\boldsymbol{\phi}} \widetilde{\mathcal{L}}$.
  Update $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ using stochastic gradient ascent.
end

5.2.4 Optimizing the Hierarchical ELBO with Respect to $\boldsymbol{\phi}$

We derived how to optimize the hierarchical ELBO with respect to $\boldsymbol{\theta}$. Optimizing with respect to the auxiliary parameters $\boldsymbol{\phi}$ is simple. The expectation in the hierarchical ELBO (Equation (5.7)) does not depend on $\boldsymbol{\phi}$; therefore we can simply pass the gradient operator inside,

$$\nabla_{\boldsymbol{\phi}} \widetilde{\mathcal{L}} = \mathbb{E}_{q(\boldsymbol{z}, \boldsymbol{\nu})}[\nabla_{\boldsymbol{\phi}} \log r(\boldsymbol{\nu} \mid \boldsymbol{z}, \boldsymbol{\phi})]. \tag{5.10}$$

5.2.5 Algorithm

Algorithm 3 outlines the inference procedure, where we evaluate noisy estimates of both gradients using samples from the joint $q(\boldsymbol{z}, \boldsymbol{\nu})$. In general, we can compute these gradients via automatic differentiation systems such as those available in Stan and Theano (Stan Development Team, 2015; Bergstra et al., 2010). These tools remove the need for model-specific computations (note that no assumption has been made on $\log p(\boldsymbol{x}, \boldsymbol{z})$ other than the ability to calculate it).

Table 5.2 outlines variational methods and their complexity requirements. HVMs with a normalizing flow prior have complexity linear in the number of latent variables. The complexity is also proportional to the total length of the flows used to represent q and r.

Black box methods	Compute	Storage	~~MF~~	Class of models
BBVI (Ranganath et al., 2014)	$\mathcal{O}(d)$	$\mathcal{O}(d)$	✗	discrete/continuous
DSVI (Titsias and Lázaro-Gredilla, 2014)	$\mathcal{O}(d^2)$	$\mathcal{O}(d^2)$	✓	continuous-diff.
COPULA VI (Tran et al., 2015)	$\mathcal{O}(d^2)$	$\mathcal{O}(d^2)$	✓	discrete/continuous
MIXTURE (Jaakkola and Jordan, 1998)	$\mathcal{O}(Kd)$	$\mathcal{O}(Kd)$	✓	discrete/continuous
NF (Rezende and Mohamed, 2015)	$\mathcal{O}(Kd)$	$\mathcal{O}(Kd)$	✓	continuous-diff.
HVM w/ NF prior	$\mathcal{O}(Kd)$	$\mathcal{O}(Kd)$	✓	discrete/continuous

Table 5.2: A summary of black box inference methods, which can support either continuous-differentiable distributions or both discrete and continuous. The variable d is the number of latent variables; for MIXTURE, K is the number of mixture components; for NF procedures, K is the number of transformations. ~~MF~~ (not mean field) marks if the approximation captures correlations.

5.2.6 Inference Networks

Classically, variational inference on models with latent variables associated with a data point requires optimizing over variational parameters whose number grows with the size of data. This process can be computationally prohibitive, especially at test time. Inference networks (Dayan, 2000; Stuhlmüller et al., 2013; Kingma and Welling, 2014; Rezende et al., 2014) amortize the cost of estimating these local variational parameters by tying them together through a neural network. Specifically, the data point specific variational parameters are outputs of a neural network that takes data points as input. The parameters of the neural network then become the variational parameters; this reduces the cost of estimating the parameters of all the data points to estimating parameters of the inference network. Inference networks can be applied to HVMS by making both the parameters (of the variational model and recursive posterior approximation) functions of their conditioning sets.

5.2.7 Related Work

There has been much work on learning posterior dependencies. Saul and Jordan (1996) and Ghahramani (1997) develop structured variational approximations: they factorize the vari-

ational family across subsets of variables, maintaining certain dependencies in the model. Unlike HVMS, however, structured approximations can require model-specific considerations and can scale poorly when used with black box methods. Lawrence (2000) develop mixture and Markov chain approximations for belief networks with lower bounds also developed with auxiliary distributions. These families complement the construction of HVMS, and can be applied as variational likelihoods.

Within the context of generic inference, Titsias and Lázaro-Gredilla (2014), Rezende and Mohamed (2015), and Kucukelbir et al. (2015) propose rich approximating families in differentiable probability models. These methods work well in practice; however, they are restricted to probability models with densities differentiable with respect to their latent variables. For undirected models, Agakov and Barber (2004) introduce the auxiliary bound for variational inference which we derived. Salimans et al. (2015) derive the same bound, but limit their attention to differentiable probability models and auxiliary distributions defined by Markov transition kernels, and Maaløe et al. (2016) study auxiliary distributions for semi-supervised learning with deep generative models. Tran et al. (2015) propose copulas as a way of learning dependencies in factorized approximations. Copulas can be efficiently extended to HVMS, whereas the full rank approach taken in Tran et al. (2015) requires computation quadratic in the number of latent variables.

These generic methods can also be building blocks for HVMS, employed as variational priors for arbitrary mean field factors. As in our example with a normalizing flow prior, this extends their scope to perform inference in discrete models (and, more generally, non-differentiable models). In other work (Tran et al., 2016), we build hierarchical variational models from Gaussian processes that can approximate arbitrary smooth posteriors.

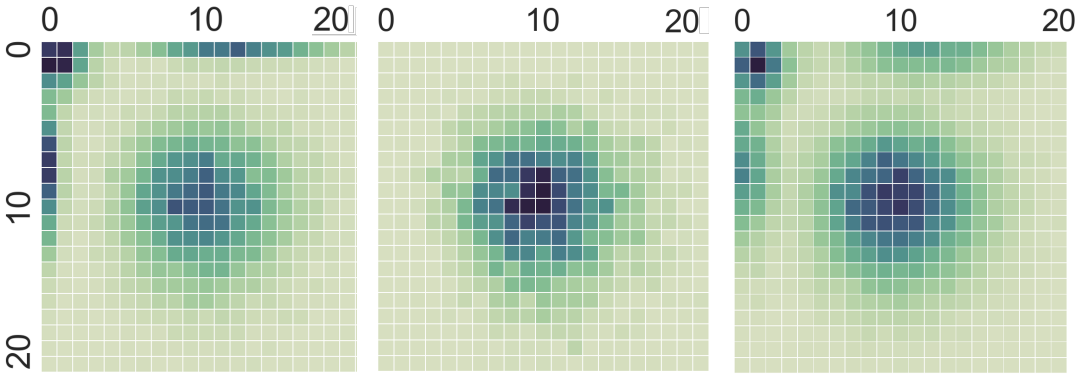


Figure 5.3: **(a)** The true posterior, which has correlated latent variables with countably infinite discrete support. **(b)** Mean field Poisson approximation. **(c)** Hierarchical variational model with a mixture of Gaussians prior. Using this prior, the HVM exhibits high fidelity to the posterior as it capture multimodality on discrete surfaces.

5.3 Empirical Study

We introduced a new class of variational families and developed efficient black box algorithms for their computation. We consider a simulated study on a two-dimensional discrete posterior; we also evaluate our proposed variational models on deep exponential families from Chapter 4. Recall that DEFS achieve state-of-the-art results on text analysis. In total, we train two variational models for the simulated study and 12 models over two datasets.

5.3.1 Correlated Discrete Latent Variables

Consider a model whose posterior distribution is a pair of discrete latent variables defined on the countable support $\{0, 1, 2, \ldots, \} \times \{0, 1, 2, \ldots, \}$; Figure 5.3 depicts its joint probability mass. The latent variables are correlated and form a complex multimodal structure. A mean field Poisson approximation has difficulty capturing this distribution; it focuses entirely on the center mass. This contrasts with hierarchical variational models, where we place a mixture prior on the Poisson distributions' rate parameters (reparameterized to share the

	Model	HVM	Mean Field
Poisson	100	**3386**	3387
	100-30	**3396**	3896
	100-30-15	**3346**	3962
Bernoulli	100	**3060**	3084
	100-30	3394	**3339**
	100-30-15	**3420**	3575

Table 5.3: *The New York Times*. Held-out perplexity (lower is better). Hierarchical variational models outperform mean field in five models. Mean field fails at multi-level Poissons; HVMs make it possible to study multi-level Poissons.

same support). This HVM fits the various modes of the correlated Poisson latent variable model and exhibits a "smoother" surface.

5.3.2 Deep Exponential Families

We now study HVMs on DEFs from the previous chapter. We focus on the Bernoulli DEF, the sigmoid belief network, and the Poisson DEF. In some sense the Poisson DEF creates one of the hardest posterior inference problems. It is nonconjugate and discrete, but marginalization tricks fail because Poisson DEFs have countable support.

Variational models. We consider the variational approximation that adds dependence to the $\boldsymbol{z}$'s. We parameterize each variational prior $q(\boldsymbol{\nu}_{\boldsymbol{z}_i})$ with a normalizing flow of length two, and use the inverse flow of length 10 for $r(\boldsymbol{\nu}_{\boldsymbol{z}_i})$. We use planar transformations (Rezende and Mohamed, 2015). In a pilot study, we found little improvement with longer flow lengths. We compare to the mean field approximation from the previous chapter that achieves state of the art results on text.

Data and evaluation. We consider two text corpora of news and scientific articles—*the New York Times* (NYT) and *Science*. We draw 11K documents from both corpora uniformly

	Model	HVM	Mean Field
Poisson	100	**3327**	3392
	100-30	**2977**	3320
	100-30-15	**3007**	3332
Bernoulli	100	**3165**	3166
	100-30	**3135**	3195
	100-30-15	**3050**	3185

Table 5.4: *Science*. Held-out perplexity (lower is better). HVM outperforms mean field on all six models. Hierarchical variational models identify that multi-level Poisson models are best, while mean field does not.

at random. NYT consists of 8K terms and *Science* consists of 5.9K terms. We train six models for each data set.

We examine held-out perplexity. This is a document completion evaluation metric (Wallach et al., 2009) where the words are tested independently. This evaluation differs slightly from the previous chapter. Instead of evaluating perplexity at the expectation, we sample from the variational approximations and compute the perplexity. This evaluates the spread of the variational approximation as well. Finally, as our evaluation uses data not included in posterior inference, it is possible for the mean field family to outperform HVMS.

Hyperparameters and convergence. We study one, two, and three layer DEFS with 100, 30, and 15 units respectively and set prior hyperparameters as in the previous chapter. For HVMS, we use Nesterov's accelerated gradient with momentum parameter of 0.9, combined with RMSProp with a scaling factor of 10^{-3} to maximize the lower bound. For the mean field family, we use the learning rate hyperparameters from the previous chapter. The HVMS converge faster on Poisson models relative to Bernoulli models. The one layer Poisson model was the fastest to infer.

Results. HVMS achieve better performance over six models and two datasets, with a mean improvement in perplexity of 180 points. (Mean field works better on only the two layer Bernoulli model on NYT.) HVMS make it feasible to work with multi-level Poisson models. This is particularly important on *Science*, where hierarchical variational models identify that multi-level Poisson models are best and mean field does not.

5.4 Discussion

Variational inference hinges on the quality of the approximating family. Traditionally, this family was chosen to match conjugacy constraints, but with the advent of BBVI, we can work with new, more accurate variational approximations. In this chapter, we developed the view that building a variational approximation parallels building a model for data. Using this view, we constructed hierarchical variational models, a rich class of variational approximations constructed by placing priors on existing variational families. We develop an easy-to-use algorithm for HVMS and demonstrate they make it feasible to work with complex, discrete models.

There are several avenues for future work: studying alternative entropy bounds, analyzing HVMS in the empirical Bayes framework, and using other data modeling tools to build new variational models. Perhaps the most interesting direction would be to develop HVMS for models where the number of latent variables creates computational challenges.

5.5 Appendix

5.5.1 Stochastic Gradient of the Hierarchical ELBO

Using the reparameterization $\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\theta})$, where $\boldsymbol{\epsilon} \sim s$, the hierarchical ELBO is

$$\widetilde{\mathcal{L}}(\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_{s(\boldsymbol{\epsilon})}[\mathcal{L}_{\text{MF}}(\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\theta})) + \mathbb{E}_{q(\boldsymbol{z}\mid\boldsymbol{\nu})}[(\log r(\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\theta})\mid \boldsymbol{z};\boldsymbol{\phi}) - \log q(\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\theta});\boldsymbol{\theta}))].$$

We now differentiate the three terms with respect to $\boldsymbol{\theta}$. As in the main text, we define the score function:

$$\boldsymbol{c} = \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}\mid\boldsymbol{\nu}).$$

By the chain rule, the derivative of the first term is

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}_{s(\boldsymbol{\epsilon})}[\mathcal{L}_{\text{MF}}(\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\theta}))] = \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}}\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\theta})\nabla_{\boldsymbol{\nu}}\mathcal{L}_{\text{MF}}(\boldsymbol{\nu})].$$

We now differentiate the second term:

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}}\mathbb{E}_{s(\boldsymbol{\epsilon})}[\mathbb{E}_{q(\boldsymbol{z}\mid\boldsymbol{\nu})}[\log r(\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\theta})\mid \boldsymbol{z};\boldsymbol{\phi}]] \\
&= \nabla_{\boldsymbol{\theta}}\mathbb{E}_{s(\boldsymbol{\epsilon})}\left[\int q(\boldsymbol{z}\mid\boldsymbol{\nu})\log r(\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\nu})\mid \boldsymbol{z};\boldsymbol{\phi})d\boldsymbol{z}\right] \\
&= \mathbb{E}_{s(\boldsymbol{\epsilon})}\left[\nabla_{\boldsymbol{\theta}}\left[\int q(\boldsymbol{z}\mid\boldsymbol{\nu})\log r(\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\theta})\mid \boldsymbol{z};\boldsymbol{\phi})d\boldsymbol{z}\right]\right] \\
&= \mathbb{E}_{s(\boldsymbol{\epsilon})}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\theta})\nabla_{\boldsymbol{\nu}}\left[\int q(\boldsymbol{z}\mid\boldsymbol{\nu})\log r(\boldsymbol{\nu}(\boldsymbol{\epsilon};\boldsymbol{\theta})\mid \boldsymbol{z};\boldsymbol{\phi})d\boldsymbol{z}\right]\right].
\end{aligned}$$

Applying the product rule to the inner derivative gives

$$
\begin{aligned}
&\nabla_{\boldsymbol{\nu}} \left[\int q(\boldsymbol{z} \mid \boldsymbol{\nu}) \log r(\boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mid \boldsymbol{z}; \boldsymbol{\phi}) d\boldsymbol{z} \right] \\
&\quad = \int \nabla_{\boldsymbol{\nu}} q(\boldsymbol{z} \mid \boldsymbol{\nu}) \log r(\boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mid \boldsymbol{z}; \boldsymbol{\phi}) d\boldsymbol{z} + \int q(\boldsymbol{z} \mid \boldsymbol{\nu}) \nabla_{\boldsymbol{\nu}} \log r(\boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mid \boldsymbol{z}; \boldsymbol{\phi}) d\boldsymbol{z} \\
&\quad = \int \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z} \mid \boldsymbol{\nu}) q(\boldsymbol{z} \mid \boldsymbol{\nu}) \log r(\boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mid \boldsymbol{z}; \boldsymbol{\phi}) d\boldsymbol{z} + \int q(\boldsymbol{z} \mid \boldsymbol{\nu}) \nabla_{\boldsymbol{\nu}} \log r(\boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mid \boldsymbol{z}; \boldsymbol{\phi}) d\boldsymbol{z} \\
&\quad = \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\boldsymbol{c} \log r(\boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mid \boldsymbol{z}; \boldsymbol{\phi})] + \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\nabla_{\boldsymbol{\nu}} \log r(\boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mid \boldsymbol{z}; \boldsymbol{\phi})].
\end{aligned}
$$

Substituting this back into the previous expression gives the gradient of the second term

$$
\mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}) \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\boldsymbol{c} \log r(\boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mid \boldsymbol{z}; \boldsymbol{\phi}) + \nabla_{\boldsymbol{\nu}} \log r(\boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mid \boldsymbol{z}; \boldsymbol{\phi})]].
$$

The third term also follows by the chain rule:

$$
\begin{aligned}
&\nabla_{\boldsymbol{\theta}} \mathbb{E}_{s(\boldsymbol{\epsilon})}[\log q(\boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}); \boldsymbol{\theta})] \\
&\quad = \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{\nu}; \boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\nu}; \boldsymbol{\theta})] \\
&\quad = \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{\nu}; \boldsymbol{\theta})],
\end{aligned}
$$

where the last equality uses

$$
\mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\nu}; \boldsymbol{\theta})] = \mathbb{E}_{q(\boldsymbol{\nu}; \boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\nu}; \boldsymbol{\theta})] = \mathbf{0}.
$$

Combining these together gives the total expression for the gradient,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \widetilde{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\nu}} \mathcal{L}_{\text{MF}}(\boldsymbol{\nu})] \\ &\quad + \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\nu}} [\log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi}) - \log q(\boldsymbol{\nu}; \boldsymbol{\theta})]] \\ &\quad + \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\boldsymbol{c} \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})]].\end{aligned}$$

5.5.2 Rao-Blackwellization and the Recursive Approximation r

One term of the gradient involves the product of the score function with all of the recursive approximation r,

$$\mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}) \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\boldsymbol{c} \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})]].$$

Rao-Blackwellizing the inner expectation can drastically reduce the variance. Recall that

$$q(\boldsymbol{z} \mid \boldsymbol{\nu}) = \prod_{i=1}^{d} q(\boldsymbol{z}_i \mid \boldsymbol{\nu}_i).$$

Next, we define $\boldsymbol{c}_i$ to be the score function of the factor. That is

$$\boldsymbol{c}_i = \nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}_i \mid \boldsymbol{\nu}_i).$$

This is a vector with nonzero entries corresponding to $\boldsymbol{\nu}_i$. Substituting the factorization into the gradient term yields

$$\mathbb{E}_{s(\boldsymbol{\epsilon})} \left[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \sum_{i=1}^{d} \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\boldsymbol{c}_i \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})] \right]. \tag{5.11}$$

Now we define r_i to be the terms in $\log r$ containing z_i and r_{-i} to be the remaining terms. Then the inner expectation in the gradient term is

$$\begin{aligned}
&\sum_{i=1}^{d} \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\boldsymbol{c}_i(\log r_i(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi}) + \log r_{-i}(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi}))] \\
&\quad = \sum_{i=1}^{d} \mathbb{E}_{q(z_i \mid \boldsymbol{\nu})}[\boldsymbol{c}_i \mathbb{E}_{q(\boldsymbol{z}_{-i} \mid \boldsymbol{\nu})}[\log r_i(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi}) + \log r_{-i}(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})]], \\
&\quad = \sum_{i=1}^{d} \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\boldsymbol{c}_i \log r_i(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})],
\end{aligned}$$

where the last equality follows from the expectation of the score function of a distribution being zero. Substituting this back into Equation (5.11) gives the desired result

$$\begin{aligned}
&\mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}[\boldsymbol{c} \log r(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})]] \\
&= \mathbb{E}_{s(\boldsymbol{\epsilon})}\left[\nabla_{\boldsymbol{\theta}} \boldsymbol{\nu}(\boldsymbol{\epsilon}; \boldsymbol{\theta}) \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{\nu})}\left[\sum_{i=1}^{d} \boldsymbol{c}_i \log r_i(\boldsymbol{\nu} \mid \boldsymbol{z}; \boldsymbol{\phi})\right]\right].
\end{aligned}$$

5.5.3 Relationship to Empirical Bayes+Reinforcement Learning

Augmentation with a variational prior has strong ties to empirical Bayesian methods, which use data to estimate hyperparameters of a prior distribution (Robbins, 1964; Efron and Morris, 1973). In general, empirical Bayes considers the fully Bayesian treatment of a hyperprior on the original prior—here, the variational prior on the original mean field—and proceeds to integrate it out. As this is analytically intractable, much work has been on parametric estimation, which seeks point estimates rather than the whole distribution encoded by the hyperprior. We avoid this at the level of the hyperprior (variational prior) via the hierarchical ELBO. However, our procedure can be viewed in this framework at one level higher. That is, we seek a point estimate of the "variational hyperprior" which governs the parameters on the variational prior.

A similar methodology also arises in the policy search literature (Rückstieß et al., 2008; Sehnke et al., 2008). Policy search methods aim to maximize the expected reward for a sequential decision-making task, by positing a distribution over trajectories and proceeding to learn its parameters. This distribution is known as the policy, and an upper-level policy considers a distribution over the original policy. This encourages exploration in the latent variable space and can be seen as a form of smoothing or annealing.

5.5.4 Multi-level $q(\boldsymbol{\nu}; \boldsymbol{\theta})$ and Optimizing with Discrete Variables in the Variational Prior

As mentioned in the main text, hierarchical variational models with multiple layers can contain both discrete and differentiable latent variables. Higher level differentiable variables follow directly from our derivation above. Discrete variables in the prior can pose a difficulty due to high variance. Local expectation gradients (Titsias and Lázaro-Gredilla, 2015) provide an efficient gradient estimator for variational approximations over discrete variables with small support. This approach can be combined with the gradient of the main text to form an efficient gradient estimator.

Chapter 6

Operator Variational Inference

In Chapter 2, we summarized variational inference with a picture (reproduced here as Figure 6.1). This figure displays the variational family and the posterior distribution, along with some notion of closeness between them. The majority of variational inference algorithms use the KL divergence as a measure of closeness. However, this choice was not justified. There are many other notions of closeness between probability distributions such as f-divergences (Csiszár et al., 2004) and integral probability metrics (Müller, 1997).

When optimizing the KL divergence, there are two issues with the posterior approximation that we highlight. First, it typically underestimates the variance of the posterior. Second, it can result in degenerate solutions that zero out the probability of certain configurations of the latent variables. While both of these issues can be partially circumvented by using more expressive approximating families like HVMS, they ultimately stem from the choice of the objective. Under the KL divergence, we pay a large price when the spread of q is bigger than the spread of p; this price becomes infinite when q has larger support than p.

The popularity of the KL divergence for variational inference partly stems from the ability to derive nice algorithms for conditionally conjugate models. Continuing the theme of

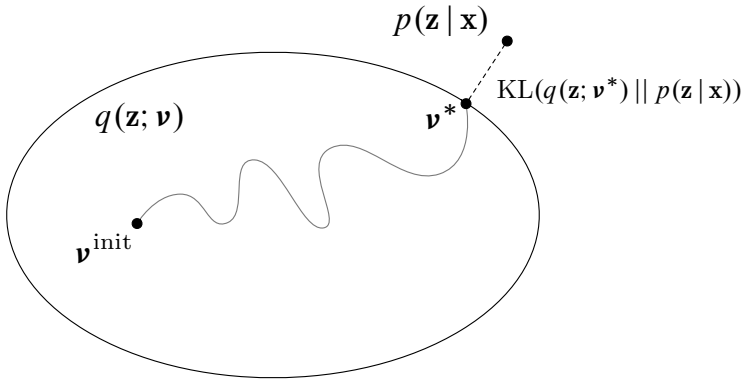


Figure 6.1: A pictorial description of variational inference. The family of variational approximations is the oval, $\boldsymbol{\nu}^{\text{init}}$ denotes the initial approximation, and $\boldsymbol{\nu}^*$ is the approximation after running variational inference. The posterior lives outside the approximating family. The distance between distributions in the picture maps to the KL divergence. Many other choices of divergence are made possible by black box variational inference style methods.

the previous chapters, BBVI style algorithms weaken this motivation as we are no longer limited to models, variational approximations, and probability distances that lead to nice updates.

In this chapter, we reexamine variational inference from its roots as an optimization problem. We use *operators*, or functions of functions, to design variational objectives. We develop a black box algorithm, OPVI, for optimizing any operator objective. Importantly, operators enable us to make explicit the statistical and computational tradeoffs for variational inference. We can characterize different properties of variational objectives, such as objectives that admit *data subsampling*—allowing inference to scale to massive data—as well as objectives that admit *variational programs*—a rich class of posterior approximations that does not require a tractable density.

6.1 Operator Variational Objectives

We define operator variational objectives and the conditions needed for an objective to be useful for variational inference. We develop a new objective, the Langevin-Stein operator objective, show how to place the classical KL divergence into this class, and explore connections between control variates, model estimation, and posterior inference. In the following section, we develop a general algorithm for optimizing operator variational objectives.

6.1.1 Operator Variational Objectives

We define a new class of variational objectives, *operator variational objectives*. An operator objective has three components. The first component is an operator $O^{p,q}$ that depends on $p(\boldsymbol{z} \mid \boldsymbol{x})$ and $q(\boldsymbol{z})$. (Recall that an operator maps functions to other functions.) The second component is a family of test functions $\mathcal{F}$, where each $f(z) \in \mathcal{F}$ maps realizations of the latent variables to real vectors $\mathbb{R}^d$. In the objective, the operator and a test function are combined in an expectation $\mathbb{E}_{q(\boldsymbol{z})}[(O^{p,q}\, f)(\boldsymbol{z})]$ designed such that values close to zero indicate that q is close to p. The third component is a distance function $t(a) : \mathbb{R} \to [0, \infty)$, which is applied to the expectation so that the objective is nonnegative. (Our example uses the square function $t(a) = a^2$.)

These three components combine to form the operator variational objective. It is a nonnegative function of the variational distribution,

$$\mathcal{L}(q; O^{p,q}, \mathcal{F}, t) = \sup_{f \in \mathcal{F}} t(\mathbb{E}_{q(\boldsymbol{z})}[(O^{p,q}\, f)(\boldsymbol{z})]). \tag{6.1}$$

Intuitively, the objective is the worst-case expected value among all test functions $f \in \mathcal{F}$. Operator variational inference seeks to minimize this objective with respect to the variational family $q \in \mathcal{Q}$.

We use operator objectives for posterior inference. This requires two conditions on the operator and the function family.

1. *Closeness*. The minimum of the variational objective is required to be the posterior, $q(\boldsymbol{z}) = p(\boldsymbol{z} \mid \boldsymbol{x})$. We meet this condition by requiring that $\mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x})}[(O^{p,p}\, f)(\boldsymbol{z})] = 0$ for all $f \in \mathcal{F}$. Thus, optimizing the objective will produce $p(\boldsymbol{z} \mid \boldsymbol{x})$ if it is the only member of $\mathcal{Q}$ with zero expectation (otherwise it will produce a distribution in the equivalence class, $q \in \mathcal{Q}$ with zero expectation). In practice, the minimum of the objective will be the closest member of $\mathcal{Q}$ to $p(\boldsymbol{z} \mid \boldsymbol{x})$.

2. *Tractability*. We can calculate the variational objective up to a constant without involving the exact posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$. We do not require calculating the normalizing constant of the posterior, which is typically intractable. We meet this condition by requiring that the operator $O^{p,q}$—originally in terms of $p(\boldsymbol{z} \mid \boldsymbol{x})$ and $q(\boldsymbol{z})$—can be written in terms of the joint distribution $p(\boldsymbol{x}, \boldsymbol{z})$ and the variational approximation $q(\boldsymbol{z})$. Tractability also imposes conditions on $\mathcal{F}$: it must be feasible to find the supremum. Below, we satisfy this by defining a parametric family for $\mathcal{F}$ that is amenable to stochastic optimization.

Equation (6.1) and these two conditions provide a mechanism to design meaningful variational objectives for posterior inference. Operator variational objectives try to match expectations with respect to $q(\boldsymbol{z})$ to those with respect to $p(\boldsymbol{z} \mid \boldsymbol{x})$.

6.1.2 Understanding Operator Variational Objectives

Consider operators where $\mathbb{E}_{q(\boldsymbol{z})}[(O^{p,q}\, f)(\boldsymbol{z})]$ only takes positive values. In this case, distance to zero can be measured with the identity $t(a) = a$, so tractability implies the operator need only be known up to a constant. This family includes tractable forms of familiar divergences like the KL divergence (ELBO), Rényi's α-divergence (Li and Turner, 2016), and the χ-divergence (Nielsen and Nock, 2013).

When the expectation can take positive or negative values, operator variational objectives are closely related to Stein divergences (Gorham and Mackey, 2015).[1] Consider a family of scalar test functions $\mathcal{F}^*$ that have expectation zero with respect to the posterior, $\mathbb{E}_{p(\boldsymbol{z}\mid\boldsymbol{x})}[f^*(\boldsymbol{z})] = 0$. Using this family, a *Stein divergence* is

$$D_{\text{Stein}}(p, q) = \sup_{f^* \in \mathcal{F}^*} |\mathbb{E}_{q(\boldsymbol{z})}[f^*(\boldsymbol{z})] - \mathbb{E}_{p(\boldsymbol{z}\mid\boldsymbol{x})}[f^*(\boldsymbol{z})]|.$$

Now recall the operator objective of Equation (6.1). The closeness condition implies that

$$\mathcal{L}(q; O^{p,q}, \mathcal{F}, t) = \sup_{f \in \mathcal{F}} t(\mathbb{E}_{q(\boldsymbol{z})}[(O^{p,q}\ f)(\boldsymbol{z})] - \mathbb{E}_{p(\boldsymbol{z}\mid\boldsymbol{x})}[(O^{p,p}\ f)(\boldsymbol{z})]).$$

Operators with positive or negative expectations lead to Stein divergences with a more generalized notion of distance.

6.1.3 Langevin-Stein Operator Variational Objective

We developed the operator variational objective. It is a class of tractable objectives, each of which can be optimized to yield an approximation to the posterior. An operator variational objective is built from an operator, function class, and distance function to zero. We now use this construction to design a new type of variational objective.

An operator objective involves a class of functions that has known expectations with respect to an intractable distribution. There are many ways to construct such classes (Barbour, 1988; Assaraf and Caffarel, 1999). Let π be a density of interest, then the classic univariate Stein operator comes from the density method (Stein et al., 2004; Ley et al., 2017). It has the

[1] The use of Stein divergence like quantities appears in multiple places in the literature, but the concept was not formalized until the cited work.

form

$$(O^{\pi} f)(z) = \frac{(\pi(z) f(z))'}{\pi(z)} = f'(z) + \frac{\pi'(z)}{\pi(z)} f(z) = f'(z) + (\log \pi(z))' f(z). \quad (6.2)$$

This operator has π expected value zero under mild technical conditions. The density method operator has a discrete analogue which looks at local differences of the product of f and π rather than derivatives.

This operator can be generalized to a family of operators on multivariate densities by considering the conditional distributions of the multivariate density. Consider the two variable case where $\boldsymbol{z} = (z_1, z_2)$; now apply the density method to the conditional $\pi(z_1 \mid z_2)$.

$$\begin{aligned}(O^{\pi(z_1 \mid z_2)} f_1)(z) &= \nabla_{z_1} f_1(z_1) + \nabla_{z_1} \log \pi(z_1 \mid z_2) f_1(z_1) \\ &= \nabla_{z_1} f_1(\boldsymbol{z}) + \nabla_{z_1} \log \pi(z_1, z_2) f_1(\boldsymbol{z}). \end{aligned} \quad (6.3)$$

The last equality follows because there can be a different f_1 for each value of z_2 (i.e., $f_1(\boldsymbol{z})$ is a function over z_1 with fixed z_2), and

$$\nabla_{z_1} \log \pi(z_1 \mid z_2) = \nabla_{z_1} [\log \pi(z_1 \mid z_2) + \log \pi(z_2)] = \nabla_{z_1} \log \pi(z_1, z_2).$$

The operator cares about distributions up to normalization constants. The conditional operator Equation (6.3) has expected value zero with respect to the conditional distribution $\pi(z_1 \mid z_2)$. This implies that the conditional operator also has expectation zero with respect to the joint distribution.

We can similarly apply the operator to the other conditional to get

$$(O^{\pi(z_2 \mid z_1)} f)(z) = \nabla_{z_2} f_2(\boldsymbol{z}) + \nabla_{z_2} \log \pi(z_1, z_2) f_2(\boldsymbol{z}).$$

This operator also has expectation zero with respect to the joint distribution of π. Two operators that have expectation zero can be summed to produce an operator that has expectation zero. Generalizing this to n dimensions, we get

$$(O^{\pi} f)(z) = \sum_{i=1}^{n} \nabla_{z_i} f_i(\boldsymbol{z}) + \nabla_{z_i} \log \pi(z_1, z_2) f_i(\boldsymbol{z}) = \nabla_z \log \pi(\boldsymbol{z})^{\top} f(\boldsymbol{z}) + \nabla^{\top} f(\boldsymbol{z}),$$

where f is vector valued and $\nabla^{\top} f$ denotes the divergence of f. We call this operator the Langevin-Stein (LS) operator after Gorham and Mackey (2015), which construct this operator based on ideas from the generator method (Barbour, 1988). Related constructions also appear in Assaraf and Caffarel (1999), Mira et al. (2013), and Oates et al. (2017). Our conditional construction of the LS operator provides an alternative interpretation of its construction as a Stein operator that breaks the model's joint distribution down into its complete conditionals. It provides a way to create operators for distributions over mixed discrete and continuous spaces by adding operators of the respective complete conditionals.

Applying this to the posterior distribution, we get

$$(O^{p}_{\text{LS}} f)(\boldsymbol{z}) = \nabla_z \log p(\boldsymbol{x}, \boldsymbol{z})^{\top} f(\boldsymbol{z}) + \nabla^{\top} f. \tag{6.4}$$

We obtain the corresponding variational objective by using the squared distance function and substituting Equation (6.4) into Equation (6.1),

$$\mathcal{L}(q; O^{p}_{\text{LS}}, \mathcal{F}) = \sup_{f \in \mathcal{F}} (\mathbb{E}_q[\nabla_z \log p(\boldsymbol{x}, \boldsymbol{z})^{\top} f(\boldsymbol{z}) + \nabla^{\top} f])^2. \tag{6.5}$$

The LS operator satisfies both our conditions. First, it satisfies closeness because it has expectation zero under the posterior (Appendix A) and its unique minimizer is the posterior (Appendix B). Loosely this follows from the fact that the operator tries to match complete conditionals, and distributions with the same complete conditionals are equivalent (think the Gibbs sampler). Second, it is tractable because it only requires the joint distribution. The

test functions f will also be a parametric family, which we detail later. Finally, it does not have the zero-forcing property of the KL divergence that causes variance underestimation. That is, unlike the KL, when q has mass where p does not, the LS operator can be finite.

6.1.4 The KL Divergence as an Operator Variational Objective

We demonstrate how classical variational methods fall inside the operator family. For example, traditional variational inference minimizes the KL divergence from an approximating family to the posterior (Jordan et al., 1999). This can be construed as an operator variational objective,

$$(O_{\mathrm{KL}}^{p,q}\, f)(\boldsymbol{z}) = \log q(\boldsymbol{z}) - \log p(\boldsymbol{z}|\boldsymbol{x}) \quad \forall f \in \mathcal{F}. \tag{6.6}$$

This operator does not use the family of functions—it trivially maps all functions f to the same function. Further, because KL is strictly positive, we use the identity distance $t(a) = a$.

The operator satisfies both conditions. It satisfies closeness because $\mathrm{KL}(p(\boldsymbol{z} \mid \boldsymbol{x}) || p(\boldsymbol{z} \mid \boldsymbol{x})) = 0$. It satisfies tractability because it can be computed up to a constant when used in the operator objective of Equation (6.1). Tractability comes from the fact that $\log p(\boldsymbol{z} \mid \boldsymbol{x}) = \log p(\boldsymbol{z}, \boldsymbol{x}) - \log p(\boldsymbol{x})$.

The KL divergence is an example of an f-divergence (Csiszár et al., 2004). The conditions of closeness and tractability to construct operator variational objectives provide constraints on f-divergences feasible for variational inference. The choice of f is limited to where the joint distribution can be used in place of the posterior (e.g., χ-divergence based variational inference (Dieng et al., 2016)).

6.1.5 Control Variates

Properties of distributions provide ways to estimate those distributions. Control variates (from Chapter 3) use known properties of distributions to compute expectation-zero functions. In this sense, control variates are intimately tied to distribution estimation.

As an example, let r be a distribution with parameters $\boldsymbol{\lambda}$ that we want to estimate. Then from Chapter 3 we know that the the expected value of the score function is zero,

$$\mathbb{E}_r[\nabla_{\boldsymbol{\lambda}} \log r(\boldsymbol{z}; \boldsymbol{\lambda})] = 0. \tag{6.7}$$

Now suppose we have access to a distribution F in the same family as r with unknown parameter $\boldsymbol{\lambda}^*$, and that we can sample from F. A natural way to estimate the true parameter $\boldsymbol{\lambda}^*$ would be to change $\boldsymbol{\lambda}$ until the invariant Equation (6.7) is met. Formally,

$$\arg\min_{\boldsymbol{\lambda}} ||\mathbb{E}_F[\nabla_{\boldsymbol{\lambda}} \log r(\boldsymbol{z}; \boldsymbol{\lambda})]||^2$$

This is a score-matching-type estimator of distributions (Hyvärinen, 2005). However, from Chapter 3, we know that control variates can be built using the score function. In a similar vein, if we know moments of r, we can estimate r with generalized method of moments (Hansen, 1982). The known moments can also be used as control variates.

Operator variational objectives provide a similar view. Consider a control variate h, now for an unnormalized density π. We have by the definition of control variates $\mathbb{E}_\pi[h] - \mathbb{E}_\pi[h] = 0$. Changing the first expectation to be with respect to an approximation q, we get an operator variational objective where $\mathcal{F}$ contains only h. The more functions that we know that have known expectation with respect to the unnormalized distribution, the better this objective. In short, every control variate technique for unnormalized densities yields a variational objective. Furthermore, these variational objectives fall into the class of Stein divergences.

6.2 Operator Variational Inference

We described operator variational objectives, a broad class of objectives for variational inference. We now examine how this class of objectives can be optimized. We develop a black box algorithm in the style of black box variational inference (BBVI) based on Monte Carlo estimation and stochastic optimization. Our algorithm applies to a general class of models and any operator objective.

Minimizing the operator objective involves two optimizations: minimizing the objective with respect to the approximating family $\mathcal{Q}$ and maximizing the objective with respect to the function class $\mathcal{F}$ (which is part of the objective).

We index the family $\mathcal{Q}$ with *variational parameters* $\boldsymbol{\nu}$ and require that it satisfies properties typically assumed by black box methods (Ranganath et al., 2014): the variational distribution $q(\boldsymbol{z}; \boldsymbol{\nu})$ has a known and tractable density; we can sample from $q(\boldsymbol{z}; \boldsymbol{\nu})$; and we can tractably compute the score function $\nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu})$. We index the function class $\mathcal{F}$ with parameters $\boldsymbol{\theta}$, and require that $f_{\boldsymbol{\theta}}(\cdot)$ is differentiable. In the experiments, we use neural networks, which are flexible enough to approximate a general family of test functions (Hornik et al., 1989).

Given parameterizations of the variational family and test family, operator variational inference (OPVI) seeks to solve a minimax problem,

$$\boldsymbol{\nu}^* = \inf_{\boldsymbol{\nu}} \sup_{\boldsymbol{\theta}} t(\mathbb{E}_{\boldsymbol{\nu}}[(O^{p,q} f_{\boldsymbol{\theta}})(\boldsymbol{z})]).$$

We will use stochastic optimization (Robbins and Monro, 1951; Kushner and Yin, 1997). In principle, we can find stochastic gradients of $\boldsymbol{\nu}$ by rewriting the objective in terms of the optimized value of $\boldsymbol{\theta}$, $\boldsymbol{\theta}^*(\boldsymbol{\nu})$. In practice, however, we simultaneously solve the maximization and minimization. Though computationally beneficial, this produces saddle points. In

our experiments we found this choice to be stable enough. We derive gradients for the variational parameters $\boldsymbol{\nu}$ and test function parameters $\boldsymbol{\nu}$. (We fix the distance function to be the square $t(a) = a^2$; the identity $t(a) = a$ also readily applies.)

Gradient with respect to $\boldsymbol{\nu}$. For a fixed test function with parameters $\boldsymbol{\theta}$, denote the objective $\mathcal{L}_{\boldsymbol{\theta}} = t(\mathbb{E}_{\boldsymbol{\nu}}[(O^{p,q}\, f_{\boldsymbol{\theta}})(\boldsymbol{z})])$. The gradient with respect to variational parameters $\boldsymbol{\nu}$ is

$$\nabla_{\boldsymbol{\nu}} \mathcal{L}_{\boldsymbol{\theta}} = 2\, \mathbb{E}_{\boldsymbol{\nu}}[(O^{p,q}\, f_{\boldsymbol{\theta}})(\boldsymbol{z})]\, \nabla_{\boldsymbol{\nu}} \mathbb{E}_{\boldsymbol{\nu}}[(O^{p,q}\, f_{\boldsymbol{\theta}})(\boldsymbol{z})].$$

Now write the second expectation with the score function gradient from Chapter 3. This gradient is

$$\nabla_{\boldsymbol{\nu}} \mathcal{L}_{\boldsymbol{\theta}} = 2\, \mathbb{E}_{\boldsymbol{\nu}}[(O^{p,q}\, f_{\boldsymbol{\theta}})(\boldsymbol{z})]\, \mathbb{E}_{\boldsymbol{\nu}}[\nabla_{\boldsymbol{\nu}} \log q(\boldsymbol{z}; \boldsymbol{\nu})(O^{p,q}\, f_{\boldsymbol{\theta}})(\boldsymbol{z}) + \nabla_{\boldsymbol{\nu}}(O^{p,q}\, f_{\boldsymbol{\theta}})(\boldsymbol{z})]. \quad (6.8)$$

Equation (6.8) lets us calculate unbiased stochastic gradients. We first generate two sets of independent samples from q; we then form Monte Carlo estimates of the first and second expectations.[2] For the second expectation, we can use the variance reduction techniques developed for black box variational inference, such as Rao-Blackwellization.

We described the score gradient because it is general. An alternative is to use the reparameterization gradient for the second expectation (Kingma and Welling, 2014; Rezende et al., 2014) also detailed in Chapter 3. Reparameterization gradients require that the operator be differentiable with respect to $\boldsymbol{z}$ and that samples from q can be drawn as a transformation of a parameter-free noise source $\boldsymbol{\epsilon}$, $\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{\epsilon}, \boldsymbol{\nu})$. In our experiments, we use the reparameterization gradient.

[2] It is possible to use one set of samples using $\mathbb{E}[x]\mathbb{E}[y] = \mathbb{E}[xy] - \mathrm{Cov}(x, y)$ and the unbiased covariance estimator.

Gradient with respect to $\boldsymbol{\theta}$. Mirroring the notation above, the operator objective for fixed variational parameters $\boldsymbol{\nu}$ is

$$\mathcal{L}_{\boldsymbol{\nu}} = t(\mathbb{E}_{\boldsymbol{\nu}}[(O^{p,q}\, f_{\boldsymbol{\theta}})(\boldsymbol{z})]).$$

The gradient with respect to the test function parameters $\boldsymbol{\theta}$ is

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\nu}} = 2\, \mathbb{E}_{\boldsymbol{\nu}}[(O^{p,q}\, f_{\boldsymbol{\theta}})(\boldsymbol{z})]\, \mathbb{E}_{\boldsymbol{\nu}}[\nabla_{\boldsymbol{\theta}} (O^{p,q}\, f_{\boldsymbol{\theta}})(\boldsymbol{z})]. \tag{6.9}$$

Again, we can construct unbiased stochastic gradients with two sets of Monte Carlo estimates. Note that gradients for the test function do not require score function gradients (or reparameterization gradients) because the expectation does not depend on $\boldsymbol{\theta}$.

Algorithm. Algorithm 4 outlines OPVI. We simultaneously minimize the variational objective with respect to the variational family $q_{\boldsymbol{\nu}}$ while maximizing it with respect to the function class $f_{\boldsymbol{\theta}}$. Given a model, operator, and function class parameterization, we can use automatic differentiation to calculate the necessary gradients (Carpenter et al., 2015). Provided the operator does not require model-specific computation, this algorithm satisfies the black box criteria.

Practicalities. For operators constructed from sums of operators, like the Langevin-Stein (LS) operator, performance improves by considering a sum of operator variational objectives rather than an operator variational objective of a sum of operators. For the squared distance,

Algorithm 4: Operator Variational Inference

Input : Model $\log p(\boldsymbol{x}, \boldsymbol{z})$, variational approximation $q(z; \boldsymbol{\nu})$.
Output: Variational parameters $\boldsymbol{\nu}$.
Initialize $\boldsymbol{\nu}$ and $\boldsymbol{\theta}$ randomly.
while *not converged* **do**
  Compute unbiased estimates of $\nabla_{\boldsymbol{\nu}} \mathcal{L}_{\boldsymbol{\theta}}$ from Equation (6.8).
  Compute unbiased esimates of $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\nu}}$ from Equation (6.9).
  Update $\boldsymbol{\nu}$, $\boldsymbol{\theta}$ with unbiased stochastic gradients.
end

the performance improvement follows from

$$\sum_i \left(\mathbb{E}\, O_i^{p,q}\right)^2 = 0 \rightarrow \left(\sum_i \mathbb{E}\, O_i^{p,q}\right)^2 = 0,$$
$$\left(\sum_i \mathbb{E}\, O_i^{p,q}\right)^2 = 0 \not\rightarrow \sum_i \left(\mathbb{E}\, O_i^{p,q}\right)^2 = 0.$$

That is, minimizing the sum of operator variational objectives implies minimizing the operator variational objective of the sum, but the reverse does not hold. Finally, this decomposition into a sum allows for the use of model structure to parallelize computation, e.g., in a hierarchal model, all operators for data-point-specific latent variables can be optimized independently.

6.3 Characterizing Variational Methods with Operators

The operator view can be used to characterize classes of variational methods such as those that admit simple data subsampling and those that support variational approximations with intractable densities called *variational programs*.

6.3.1 Data Subsampling and OPVI

With stochastic optimization, data subsampling scales up traditional variational inference to massive data (Hoffman et al., 2013). The idea is to calculate noisy gradients by repeatedly subsampling from the data set, without needing to pass through the entire data set for each gradient. We review it briefly in the next paragraph.

Stochastic variational inference works for hierarchical models. Hierarchical models consist of global latent variables $\boldsymbol{\beta}$ that are shared across data points and local latent variables $\boldsymbol{z}_i$ each of which is associated to a data point $\boldsymbol{x}_i$. The model's log joint density is

$$\log p(\boldsymbol{x}_{1:n}, \boldsymbol{z}_{1:n}, \boldsymbol{\beta}) = \log p(\boldsymbol{\beta}) + \sum_{i=1}^{n} \Big[\log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i, \boldsymbol{\beta}) + \log p(\boldsymbol{z}_i \mid \boldsymbol{\beta}) \Big].$$

Hoffman et al. (2013) calculate unbiased estimates of the log joint density (and its gradient) by subsampling data and appropriately scaling the sum.

We can characterize whether OPVI with a particular operator supports data subsampling. OPVI relies on evaluating the operator and its gradient at different realizations of the latent variables (Equation (6.8) and Equation (6.9)). We can subsample data to calculate estimates of the operator when the operator derives from linear operators of the log density of the model. This is because as a linear operator of sums is a sum of linear operators, so the gradients in Equation (6.8) and Equation (6.9) decompose into a sum. Examples of linear operators include differentiation and the identity operator. This reveals that since both the Langevin-Stein and KL operator are linear in the log density of the model, they support data subsampling.

6.3.2 Variational Programs

Given an operator and variational family, Algorithm 4 optimizes the corresponding operator objective. Certain operators require the density of q. For example, the KL operator (Equation (6.6)) requires its log density. This potentially limits the construction of rich variational approximations for which the density of q is difficult to compute.[3]

Some operators, however, do not depend on having an analytic density for the variational approximation q; the Langevin-Stein operator (Equation (6.4)) is an example. These operators can be used with a much richer class of variational approximations; those that can be sampled from, but might not have analytically tractable densities. We call such approximating families *variational programs*.

Inference with one type of variational programs requires the family to be reparameterizable (Kingma and Welling, 2014; Rezende et al., 2014).[4] A reparameterizable variational program consists of a parametric deterministic transformation R of random noise $\boldsymbol{\epsilon}$. Let

$$\boldsymbol{\epsilon} \sim \text{Normal}(0, 1), \quad \boldsymbol{z} = R(\boldsymbol{\epsilon}; \boldsymbol{\nu}). \tag{6.10}$$

This generates samples for $\boldsymbol{z}$, is differentiable with respect to $\boldsymbol{\nu}$, and its density may be intractable. For operators that do not require the density of q, variational programs can be used as a powerful variational approximation. This is in contrast to the standard Kullback-Leibler (KL) operator.

As an example, consider the following variational program for a one-dimensional random variable. Let ν_i denote the ith dimension of $\boldsymbol{\nu}$ and make the corresponding definition for $\boldsymbol{\epsilon}$.

[3] It is possible to construct rich approximating families using hierarchical variational models with $\text{KL}(q||p)$, but this requires the introduction of an auxiliary distribution or recursive variational approximation.

[4] We can also define variational programs with tractable joint distributions, but intractable marginal distributions, using the score function gradient.

Then we have

$$z = (\epsilon_3 > 0) R(\epsilon_1; \nu_1) - (\epsilon_3 \leq 0) R(\epsilon_2; \nu_2). \tag{6.11}$$

When R outputs positive values, the previous equation separates the parameterization of the density to the positive and negative half of the reals; its density is generally intractable. In the experiments, we will use this distribution as a variational approximation.

Variational programs Equation (6.10) contain many densities when the function class R can approximate arbitrary continuous functions (see the discussion in Chapter 4). Variational programs combined with a q-independent operator that has the posterior as the unique minimizer lead to variational inference algorithms that approximate the posterior arbitrarily well.

Operators that support variational programs must also be Markov chain Monte Carlo (MCMC) sample quality metrics in the style of Gorham and Mackey (2015)—they measure closeness to the posterior of a set of samples and do not require the density of the samples. These operators must also be control variates for MCMC in the style of Assaraf and Caffarel (1999), Mira et al. (2013), and Oates et al. (2017). We summarize this in the commutative diagram Figure 6.2.

6.4 Empirical Study

We evaluate operator variational inference on a mixture of Gaussians, comparing different choices in the objective. We then study logistic factor analysis for images.

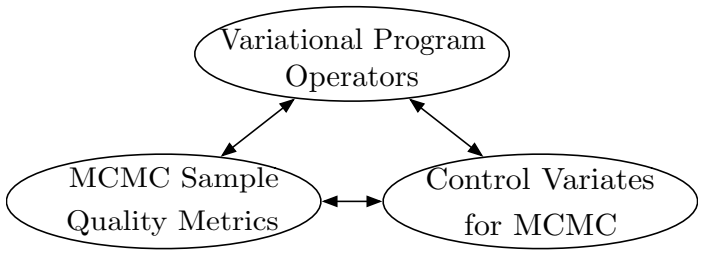


Figure 6.2: Operator variational objectives that support variational programs derive from expectation zero functions with respect to the posterior. In this sense they can measure the quality of samples from the posterior (Gorham and Mackey, 2015) and are control variates for computing posterior expectations (Assaraf and Caffarel, 1999; Mira et al., 2013; Oates et al., 2017).

6.4.1 Mixture of Gaussians

Consider a one-dimensional mixture of Gaussians as the posterior of interest,

$$p(z) \;=\; \frac{1}{2}\text{Normal}(z; -3, 1) + \frac{1}{2}\text{Normal}(z; 3, 1).$$

The posterior contains multiple modes. We seek to approximate it with three variational objectives: Kullback-Leibler (KL) with a Gaussian approximating family, Langevin-Stein (LS) with a Gaussian approximating family, and LS with a variational program.

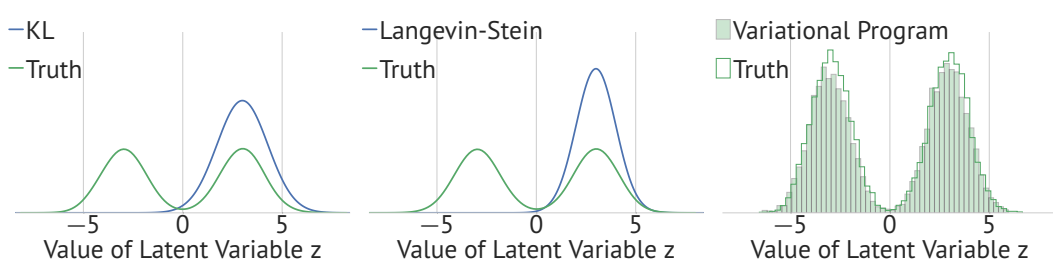


Figure 6.3: The true posterior is a mixture of two Gaussians (the green curve). We approximate it with a Gaussian using two operators (the blue curves). The density on the far right is a variational program given in Equation (6.11) and using the Langevin-Stein operator; it approximates the truth well. The density of the variational program is intractable. We plot a histogram of its samples and compare this to the histogram of the true posterior.

Figure 6.3 displays the posterior approximations. We find that the KL divergence and LS divergence choose a single mode and have slightly different variances. These operators do not produce good results because a single Gaussian is a poor approximation to the mixture. The remaining distribution in Figure 6.3 comes from the toy variational program described by Equation (6.11) with the LS operator objective. Because this program captures different distributions for the positive and negative half of the real line, it is able to capture the posterior.

In general, the choice of an objective trades off the statistical and computational properties of variational inference. We highlight one tradeoff: the LS objective admits the use of a variational program; however, the objective is more difficult to optimize than the KL.

6.4.2 Logistic Factor Analysis

Logistic factor analysis models binary vectors $\boldsymbol{x}_i$ with a matrix of parameters $\boldsymbol{W}$ and biases $\boldsymbol{b}$,

$$\begin{aligned} \boldsymbol{z}_i &\sim \text{Normal}(0, 1) \\ x_{i,k} &\sim \text{Bernoulli}(\sigma(\boldsymbol{w}_k^\top \boldsymbol{z}_i + b_k)), \end{aligned}$$

where $\boldsymbol{z}_i$ has fixed dimension K and σ is the sigmoid function. This model captures correlations of the entries in $\boldsymbol{x}_i$ through $\boldsymbol{W}$.

We apply logistic factor analysis to analyze the binarized MNIST data set (Salakhutdinov and Murray, 2008), which contains 28x28 binary pixel images of handwritten digits. (We set the latent dimensionality to 10.) We fix the model parameters to those learned with variational expectation-maximization using the KL divergence, and focus on comparing posterior inferences.

Inference method	Completed data log likelihood
Mean field Gaussian + KL	-59.3
Mean field Gaussian + LS	-75.3
Variational Program + LS	-58.9

Table 6.1: Benchmarks on logistic factor analysis for binarized MNIST. The same variational approximation with LS performs worse than KL on likelihood. The variational program with LS performs better, without directly optimizing for likelihoods.

We compare the KL operator to the LS operator and study two choices of variational models: a fully factorized Gaussian distribution and a variational program. The variational program generates samples by transforming a K-dimensional standard normal input with a two layer neural network, using rectified linear activation functions and a hidden size of twice the latent dimensionality. Formally, the variational program we use generates samples of $\boldsymbol{z}$ as follows:

$$
\begin{aligned}
\boldsymbol{z}_0 &\sim \text{Normal}(0, I) \\
\boldsymbol{h}_0 &= \text{ReLU}(\boldsymbol{W}_0^{q\top} \boldsymbol{z}_0 + \boldsymbol{b}_0^q) \\
\boldsymbol{h}_1 &= \text{ReLU}(\boldsymbol{W}_1^{q\top} \boldsymbol{h}_0 + \boldsymbol{b}_1^q) \\
\boldsymbol{z} &= \boldsymbol{W}_2^{q\top} \boldsymbol{h}_1 + \boldsymbol{b}_2^q.
\end{aligned}
$$

The variational parameters are the weights $\boldsymbol{W}^q$ and biases $\boldsymbol{b}^q$. For f, we use a three-layer neural network with the same hidden size as the variational program and hyperbolic tangent activations where unit activations are bounded to have norm two. Bounding the unit norm bounds the divergence. We use the Adam optimizer (Kingma and Ba, 2014) with learning rates 2×10^{-4} for f and 2×10^{-5} for the variational approximation.

There is no standard for evaluating generative models and their inference algorithms (Theis et al., 2016). Following Rezende et al. (2014), we consider a missing data problem. We remove half of the pixels in the test set (at random) and reconstruct them from a fitted posterior predictive distribution. Table 6.1 summarizes the results on 100 test images; we

report the log likelihood of the completed image. LS with the variational program performs best. It is followed by KL and the simpler LS inference. The LS performs better than KL even though the model parameters were learned with KL.

6.5 Summary

We presented operator variational objectives, a broad, yet tractable, class of optimization problems for approximating posterior distributions. Operator objectives are built from an operator, a family of test functions, and a distance function. We outlined the connection between operator objectives and existing divergences such as the KL divergence, and developed a new variational objective using the Langevin-Stein operator. In general, operator objectives produce new ways of posing variational inference.

Given an operator objective, we develop a black box algorithm for optimizing it and show which operators allow scalable optimization through data subsampling. Unlike the popular KL divergence, not all operators explicitly depend on the approximating density. This freedom permits flexible approximating families, called variational programs, where the distributional form is not tractable. We demonstrated this approach on a mixture model and a factor model of images.

There are several possible avenues for future directions. There is room for more cross fertilization between control variates, two-sample testing, and sample quality metrics. For example, we could use newer operators derived for sample quality in variational inference (Gorham et al., 2016), and use their characterizations of operators via mixing time to investigate the complexity of learning an accurate variational approximation.

Other directions include developing new variational objectives, adversarially learning (Goodfellow et al., 2014) model parameters with operators, and learning model parameters with operator variational objectives.

6.6 Appendix

6.6.1 Technical Conditions for Langevin-Stein Operators

We establish the conditions needed on the function class $\mathcal{F}$ and/or the posterior distribution (shorthand p) for the operators to have expectation zero for all $f \in \mathcal{F}$. We derive properties using integration by parts for supports that are bounded open sets. Then we extend the result to unbounded supports using limits.

We start with the Langevin-Stein operator. Let S be the set over which we integrate and let B be its boundary. Let v be the unit normal to the surface B, and v_i be the ith component of the surface normal (which is d dimensional). Then we have that

$$\begin{aligned}
\int_S p(O^p_{\mathrm{LS}} f) dS &= \int_S p \nabla_z \log p^\top f + p \nabla^\top f dS \\
&= \sum_{i=1}^d \int_S \frac{\partial}{\partial z_i}[p] f_i + p \frac{\partial}{\partial z_i}[f_i] dS \\
&= \sum_{i=1}^d \int_S \frac{\partial}{\partial z_i}[p] f_i dS + \int_B f_i p v_i dB - \int_S \frac{\partial}{\partial z_i}[p] f_i dS \\
&= \int_B v^\top f p dB.
\end{aligned}$$

A sufficient condition for this expectation to be zero is that either p goes to zero at its boundary or that the vector field f is zero at the boundary.

For unbounded sets, the result can be written as a limit for a sequence of increasing sets $S_n \to S$ and a set of boundaries $B_n \to B$ using the dominated convergence theorem (Cinlar, 2011). To use dominated convergence, we establish absolute integrability. Sufficient conditions for absolute integrability of the Langevin-Stein operator are for the gradient of $\log p$ to be bounded and the vector field f and its derivatives to be bounded. Via dominated convergence, we get that $\lim_n \int_{B_n} v^\top f p dB_n = 0$ for the Langevin-Stein operator to have expectation zero.

6.6.2 Characterizing the Zeros of the Langevin-Stein Operators

We provide analysis on how to characterize the equivalence class of distributions defined as $(O^{p,q} f)(z) = 0$. One general condition for equality in distribution comes from equality in probability on all Borel sets. We can test this equality with functions that have expectation zero with respect to the posterior. For any Borel set A, let δ_A be the indicator function, these functions on A have the form

$$\delta_A(\boldsymbol{z}) - \int_A p(\boldsymbol{y}) d\boldsymbol{y}.$$

We show that if the Langevin-Stein operator satisfies $\mathcal{L}(q; O^p_{\text{LS}}, \mathcal{F}) = 0$, then q is equivalent to p in distribution. We do this by showing the above functions are in the span of O^p_{LS}. Expanding the Langevin-Stein operator, we have

$$(O^p_{\text{LS}} f) = p^{-1} \nabla_z p^\top f + \nabla^\top f = p^{-1} \sum_{i=1}^{d} \frac{\partial f_i p}{\partial z_i}.$$

Setting this equal to the desired function above yields the differential equation

$$\delta_A(z) - \int_A p(y) dy = p^{-1}(z) \sum_{i=1}^{d} \frac{\partial f_i p}{\partial z_i}(z).$$

To solve this, set $f_i = 0$ for all but $i = 1$. This yields

$$\delta_A(z) - \int_A p(y)dy = p^{-1}(z)\frac{\partial f_1 p}{\partial z_1}(z),$$

which is an ordinary differential equation with solution for f_1

$$f_1^A(z) = \frac{1}{p(z)} \int_{-\infty}^{z_1} p(a, z_{2\ldots d}) \left(\delta_A(a, z_{2\ldots d}) - \int_A p(\boldsymbol{y}) d\boldsymbol{y} \right) da.$$

This function is differentiable with respect to z_1, so this gives the desired result. Plugging the function back into the operator variational objective gives

$$\mathbb{E}_q \left[\delta_A(\boldsymbol{z}) - \int_A p(\boldsymbol{y}) d\boldsymbol{y} \right] = 0 \iff \mathbb{E}_q[\delta_A(\boldsymbol{z})] = \mathbb{E}_p[\delta_A(\boldsymbol{z})],$$

for all Borel measurable A. This implies that the LS operator captures total variation.

Chapter 7

Discussion

Generative probabilistic models underlie many of the tools for data in a range of areas including language, imaging, personalization, and healthcare. For example, they find hidden structure in a collection of news articles; they help make recommendations for users joining a new system; they form the foundation for realistic image completion; they make survival predictions in the face of noise and missing observations. These were just the uses presented in this thesis. More broadly, probabilistic models lay the foundation for reasoning in the face of uncertainty.

The core challenge in working with generative probabilistic models stems from the intractability of computing the posterior distribution. Approximate methods, whether based on sampling like Markov chain Monte Carlo or optimization methods like variational inference, form the main tools for managing this intractability. Variational inference was hampered by the need for model-specific analysis for the majority of new models. This added work made variational inference unpalatable to scientists outside the fields of machine learning and statistics.

To remedy this, we developed black box variational inference. Black box variational inference provides an easy-to-use technique for deploying variational inference in a broad class of models. It makes probabilistic modeling accessible to scientists and allows for models to be built for data rather than for ease of analysis.

Black box variational inference not only frees us to build new kinds of models, it also opens the door to improvements in inference itself. Black box variational inference based ideas allow for better posterior approximations. We were also able to develop new distance measures that trade off different statistical and computational properties. These advances allow the posterior approximations to be designed for the task at hand.

Combined with other advances in approximate inference, the methods in this thesis open exciting new possibilities in analyzing data with models that are now possible to infer well. But, many challenges still remain. The path forward requires systems for inference, tools to understand the difficulty of inference in models, methods to use weak prior information, and improved techniques to diagnose and debug models. I discuss a couple of these directions in depth.

Systems for inference. New methods for inference simplify the process of working with new models. However, reimplementing these algorithms wastes time and is prone to bugs. Developing systems that provide standard implementations of inference algorithms addresses this problem and leads to new research questions.

One big question for variational inference is how to automate nuisance parameters like learning rates and initialization. As an example of this kind of work in sampling, setting Hamiltonian Monte Carlo's (Duane et al., 1987) nuisance parameters was greatly simplified by Hoffman and Gelman (2014) which helped automate inference in the Stan probabilistic programming system (Stan Development Team, 2015).

Another question is how to check the quality of inference for a particular model. This question is paramount for users to debug failures when using approximate inference. Model checking tools are also be needed to help users figure out how to improve their models. Systems for inference can be used as platforms for tackling the challenges of designing tools for debugging models and checking inference.

Using prior information. The value of models comes from a mixture of prior information and observed data. Much of the focus in recent model-building work in machine learning has been on models that approximate any distribution, empirical models. This approach works for tasks where data are readily available. However for many problems, like rare medical conditions, data are limited. This setting necessitates prior information. Building models that mix weak prior information with efficient machine learning models is an area ripe for further development. Such models would allow possibly incorrect knowledge to form the basis for the model, and the observed data to correct it when possible.

Ultimately, generative probabilistic modeling requires posterior inference. The value of this approach hinges on the ease-of-use, reliability, and scalability of posterior approximation algorithms. With recent advances, we have pushed the boundary of feasible models and improved the quality of inference. The time is right to push forward on inference, so that generative probabilistic models can be go-to instruments in the tool boxes of datasmiths and scientists everywhere.

Bibliography

Agakov, F. V. and Barber, D. (2004). An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer.

Airoldi, E., Blei, D., Fienberg, S., and Xing, E. (2008). Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014.

Allanach, B. C., Cranmer, K., Lester, C. G., and Weber, A. M. (2007). Natural priors, CMSSM fits and LHC weather forecasts. *Journal of High Energy Physics*, 2007(08):023.

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276.

Assaraf, R. and Caffarel, M. (1999). Zero-variance principle for Monte Carlo algorithms. *Physical review letters*, 83(23):4682.

Barbour, A. D. (1988). Stein's method and Poisson process convergence. *Journal of Applied Probability*, 25(A):175–184.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence*, 35(8).

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer New York.

Blei, D. and Lafferty, J. (2006a). Correlated topic models. In *Advances in Neural Information Processing Systems*, pages 147–154. MIT Press.

Blei, D. and Lafferty, J. (2006b). Dynamic topic models. In *International Conference on Machine Learning*.

Blei, D. and Lafferty, J. (2007). A correlated topic model of Science. *Annals of Applied Statistics*, 1(1):17–35.

Blei, D., Ng, A., and Jordan, M. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *Proceedings of Conference on Computational Natural Language Learning*.

Braun, M. and McAuliffe, J. (2010). Variational inference for large-scale models of discrete choice. *Journal of American Statistical Association*, 105(489).

Brown, L. (1986). *Fundamentals of Statistical Exponential Families*. Institute of Mathematical Statistics, Hayward, CA.

Buntine, W. and Jakulin, A. (2004). Applying discrete PCA in data analysis. In *Uncertainty in Artificial Intelligence*, pages 59–66. AUAI Press.

Canny, J. (2004). GaP: A factor model for discrete data. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Carbonetto, P., King, M., and Hamze, F. (2009). A stochastic approximation method for inference in probabilistic graphical models. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems*, pages 216–224.

Carpenter, B., Hoffman, M. D., Brubaker, M., Lee, D., Li, P., and Betancourt, M. (2015). The Stan Math Library: Reverse-mode automatic differentiation in C++. *arXiv preprint arXiv:1509.07164*.

Casella, G. and Robert, C. P. (1996). Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94.

Cinlar, E. (2011). *Probability and Stochastics*. Springer.

Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 215–242.

Csiszár, I., Shields, P. C., et al. (2004). Information theory and statistics: A tutorial. *Foundations and Trends® in Communications and Information Theory*, 1(4):417–528.

Davis, P. J. and Rabinowitz, P. (2007). *Methods of numerical integration*. Courier Corporation.

Dayan, P. (2000). Helmholtz machines and wake-sleep learning. *Handbook of Brain Theory and Neural Network. MIT Press, Cambridge, MA*.

Dieng, A. B., Tran, D., Ranganath, R., Paisley, J., and Blei, D. M. (2016). The χ-divergence for approximate inference. *arXiv preprint arXiv:1611.00328*.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Efron, B. (2012). *Large-scale inference: empirical Bayes methods for estimation, testing, and prediction*, volume 1. Cambridge University Press.

Efron, B. and Morris, C. (1973). Combining possibly related estimation problems. *Journal of the Royal Statistical Society, Series B*, 35(3):379–421.

Feynman, R. P. (1982). Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488.

Fu, M. C. (2006). Gradient estimation. *Handbooks in operations research and management science*, 13:575–616.

Gelfand, A. and Smith, A. (1990). Sampling based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409.

Gelman, A., Carlin, J., Stern, H., and Rubin, D. (2003). *Bayesian Data Analysis, Second Edition*. Chapman & Hall, London, 2nd edition.

Gelman, A. and Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge Univ. Press.

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

Gershman, S. J. and Blei, D. M. (2012). A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1 – 12.

Ghahramani, Z. (1995). Factorial learning and the EM algorithm. In *Advances in Neural Information Processing Systems*, pages 617–624.

Ghahramani, Z. (1997). On structured variational approximations. Technical report, Citeseer.

Ghahramani, Z. and Beal, M. J. (2001). Propagation algorithms for variational Bayesian learning. In *Advances in Neural Information Processing Systems*, pages 507–513.

Glasserman, P. (2013). *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media.

Glynn, P. W. (1990). Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84.

Goodfellow, I., Bengio, Y., and Courville, A. C. (2012). Large-scale feature learning with spike-and-slab sparse coding. In *International Conference on Machine Learning*, pages 1439–1446.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Gopalan, P., Hofman, J. M., and Blei, D. M. (2015). Scalable recommendation with hierarchical Poisson factorization. In *Uncertainty in Artificial Intelligence*, pages 326–335.

Gorham, J., Duncan, A. B., Vollmer, S. J., and Mackey, L. (2016). Measuring sample quality with diffusions. *arXiv preprint arXiv:1611.06972*.

Gorham, J. and Mackey, L. (2015). Measuring sample quality with Stein's method. In *Advances in Neural Information Processing Systems*, pages 226–234.

Hansen, L. P. (1982). Large sample properties of generalized method of moments estimators. *Econometrica: Journal of the Econometric Society*, pages 1029–1054.

Harrell, F., Califf, R., Pryor, D., Lee, K., and Rosati, R. (1982). Evaluating the yield of medical tests. *JAMA*, 247(18):2543–2546.

Hastings, W. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109.

Hernández-Lobato, D., Hernández-Lobato, J. M., and Dupont, P. (2013). Generalized spike-and-slab priors for Bayesian group feature selection using expectation propagation. *The Journal of Machine Learning Research*, 14(1):1891–1945.

Hinton, G., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hjort, N., Holmes, C., Mueller, P., and Walker, S. (2010). *Bayesian Nonparametrics: Principles and Practice*. Cambridge University Press, Cambridge, UK.

Hoffman, M., Blei, D., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14(1303–1347).

Hoffman, M. D. and Gelman, A. (2014). The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709.

Ishwaran, H. and Rao, J. S. (2005). Spike and slab variable selection: Frequentist and Bayesian strategies. *The Annals of Statistics*, 33(2):730–773.

Jaakkola, T. and Jordan, M. (1997). A variational approach to bayesian logistic regression models and their extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*, volume 82, page 4.

Jaakkola, T. S. and Jordan, M. I. (1998). Improving the Mean Field Approximation Via the Use of Mixture Distributions. In *Learning in Graphical Models*, pages 163–173. Springer Netherlands, Dordrecht.

Järvelin, K. and Kekäläinen, J. (2000). IR evaluation methods for retrieving highly relevant documents. In *In International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 41–48. ACM.

Jordan, M., editor (1999). *Learning in Graphical Models*. MIT Press, Cambridge, MA.

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. (1999). Introduction to variational methods for graphical models. *Machine Learning*, 37:183–233.

Kalman, R. (1960). A new approach to linear filtering and prediction problems a new approach to linear filtering and prediction problems. *Transaction of the AMSE: Journal of Basic Engineering*, 82:35–45.

Kingma, D. and Welling, M. (2014). Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR-14)*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751.

Kleijn, B. J. and van der Vaart, A. W. (2006). Misspecification in infinite-dimensional Bayesian statistics. *The Annals of Statistics*, pages 837–877.

Knowles, D. and Minka, T. (2011). Non-conjugate variational message passing for multinomial and binary regression. In *Advances in Neural Information Processing Systems*.

Kucukelbir, A., Ranganath, R., Gelman, A., and Blei, D. (2015). Automatic variational inference in Stan. In *Advances in Neural Information Processing Systems*, pages 568–576.

Kushner, H. and Yin, G. (1997). *Stochastic Approximation Algorithms and Applications*. Springer New York.

Larochelle, H. and Lauly, S. (2012). A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*, pages 2708–2716.

Lawrence, N. (2000). *Variational Inference in Probabilistic Models*. PhD thesis.

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

Ley, C., Reinert, G., Swan, Y., et al. (2017). Stein's method for comparison of univariate distributions. *Probability Surveys*, 14:1–52.

Li, W. and McCallum, A. (2006). Pachinko allocation: Dag-structured mixture models of topic correlations. In *International Conference on Machine Learning*, pages 577–584. ACM.

Li, Y. and Turner, R. E. (2016). Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*.

Linderman, S., Johnson, M., Miller, A., Adams, R., Blei, D., and Paninski, L. (2017). Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 914–922.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. (2016). Auxiliary deep generative models. In *International Conference on Machine Learning (ICML 2016)*.

Manning, J. R., Ranganath, R., Norman, K. A., and Blei, D. M. (2014). Topographic factor analysis: a Bayesian model for inferring brain networks from neural data. *PloS one*, 9(5):e94914.

Marlin, B. (2004). Collaborative filtering: A machine learning perspective. Technical report, University of Toronto.

McCullagh, P. and Nelder, J. (1989). *Generalized Linear Models*. London: Chapman and Hall.

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

Minka, T. (2001). *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology.

Mira, A., Solgi, R., and Imparato, D. (2013). Zero variance Markov chain Monte Carlo for Bayesian estimators. *Statistics and Computing*, pages 1–10.

Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*.

Mnih, A. and Rezende, D. (2016). Variational inference for monte carlo objectives. In *International Conference on Machine Learning*, pages 2188–2196.

Mohamed, S., Heller, K., and Ghahramani, Z. (2008). Bayesian exponential family PCA. In *Advances in Neural Information Processing Systems*.

Müller, A. (1997). Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443.

Nallapati, R. M., Ahmed, A., Xing, E. P., and Cohen, W. W. (2008). Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–550. ACM.

Neal, R. (1990). Learning stochastic feedforward networks. *Tech. Rep. CRG-TR-90-7: Department of Computer Science, University of Toronto*.

Neal, R. (2000). Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

Nelsen, R. B. (2006). *An Introduction to Copulas (Springer Series in Statistics)*. Springer-Verlag New York, Inc.

Nielsen, F. and Nock, R. (2013). On the chi square and higher-order chi distances for approximating f-divergences. *arXiv preprint arXiv:1309.3029*.

Oates, C. J., Girolami, M., and Chopin, N. (2017). Control functionals for Monte Carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):695–718.

Owen, A. B. (2013). *Monte Carlo theory, methods and examples*.

Paisley, J., Blei, D., and Jordan, M. (2012). Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning*.

Pritchard, J., Stephens, M., and Donnelly, P. (2000). Inference of population structure using multilocus genotype data. *Genetics*, 155:945–959.

Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In *International Conference on Artificial Intelligence and Statistics*, pages 814–822.

Ranganath, R., Perotte, A., Elhadad, N., and Blei, D. (2016a). Deep survival analysis. In *Machine Learning for Healthcare Conference*, pages 101–114.

Ranganath, R., Perotte, A. J., Elhadad, N., and Blei, D. M. (2015a). The survival filter: Joint survival analysis with a latent time series. In *Uncertainty in Artificial Intelligence*, pages 742–751.

Ranganath, R., Tang, L., Charlin, L., and Blei, D. M. (2015b). Deep Exponential Families. In *Artificial Intelligence and Statistics*.

Ranganath, R., Tran, D., Altosaar, J., and Blei, D. (2016b). Operator variational inference. In *Advances in Neural Information Processing Systems*, pages 496–504.

Ranganath, R., Tran, D., and Blei, D. (2016c). Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333.

Regier, J., Miller, A., McAuliffe, J., Adams, R., Hoffman, M., Lang, D., Schlegel, D., and Prabhat, M. (2015). Celeste: Variational inference for a generative model of astronomical images. In *International Conference on Machine Learning*, pages 2095–2103.

Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286.

Robbins, H. (1964). The empirical bayes approach to statistical decision problems. *The Annals of Mathematical Statistics*, pages 1–20.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):pp. 400–407.

Ross, S. M. (2002). *Simulation*. Elsevier.

Rückstieß, T., Felder, M., and Schmidhuber, J. (2008). State-dependent exploration for policy gradient methods. In *Machine Learning and Knowledge Discovery in Databases*, pages 234–249. Springer.

Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In *International Conference on Artifical Intelligence and Statistics*, pages 448–455.

Salakhutdinov, R. and Mnih, A. (2008). Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264.

Salakhutdinov, R. and Murray, I. (2008). On the quantitative analysis of deep belief networks. In *International Conference on Machine Learning*.

Salimans, T., Kingma, D., and Welling, M. (2015). Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226.

Salimans, T., Knowles, D. A., et al. (2013). Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882.

Saul, L. K. and Jordan, M. I. (1996). Exploiting tractable substructures in intractable networks. *Advances in Neural Information Processing Systems*, pages 486–492.

Sedigh-Sarvestani, M., Albers, D. J., and Gluckman, B. J. (2012). Data assimilation of glucose dynamics for use in the intensive care unit. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 5437–5440. IEEE.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. (2008). Policy gradients with parameter-based exploration for control. In *Artificial Neural Networks-ICANN 2008*, pages 387–396. Springer.

Smolenksy, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1.

Stan Development Team (2015). Stan: A c++ library for probability and sampling, version 2.8.0.

Stein, C., Diaconis, P., Holmes, S., Reinert, G., et al. (2004). Use of exchangeable pairs in the analysis of simulations. In *Stein's Method*, pages 1–25. Institute of Mathematical Statistics.

Stuhlmüller, A., Taylor, J., and Goodman, N. (2013). Learning stochastic inverses. In *Advances in Neural Information Processing Systems*, pages 3048–3056.

Sudderth, E. B. and Jordan, M. I. (2009). Shared segmentation of natural scenes using dependent pitman-yor processes. In *Advances in Neural Information Processing Systems*, pages 1585–1592.

Sunehag, P., Trumpf, J., Schraudolph, N. N., and Vishwanathan, S. (2009). Variable metric stochastic approximation theory. In *International Conference on Artificial Intelligence and Statistics*, pages 560–566.

Theis, L., van den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. In *International Conference on Learning Representations*.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning*.

Titsias, M. and Lázaro-Gredilla, M. (2014). Doubly stochastic variational bayes for non-conjugate inference. In *International Conference on Machine Learning*, pages 1971–1979.

Titsias, M. K. and Lázaro-Gredilla, M. (2015). Local expectation gradients for black box variational inference. In *Neural Information Processing Systems*.

Tokdar, S. T. and Kass, R. E. (2010). Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60.

Tran, D., Blei, D. M., and Airoldi, E. M. (2015). Copula variational inference. In *Neural Information Processing Systems*.

Tran, D., Ranganath, R., and Blei, D. M. (2016). The variational Gaussian process. In *International Conference on Learning Representations*.

Wainwright, M. and Jordan, M. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305.

Wallach, H., Murray, I., Salakhutdinov, R., and Mimno, D. (2009). Evaluation methods for topic models. In *International Conference on Machine Learning (ICML)*.

Wang, C. and Blei, D. M. (2011). Collaborative topic modeling for recommending scientific articles. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*.

Wang, C. and Blei, D. M. (2013). Variational inference for nonconjugate models. *JMLR*.

Wang, C., Chen, X., Smola, A. J., and Xing, E. P. (2013). Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, pages 181–189.

Welling, M., Rosen-Zvi, M., and Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. In *Advances in Neural Information Processing Systems*, pages 1481–1488.

Weng, S.-K., Kuo, C.-M., and Tu, S.-K. (2006). Video object tracking using adaptive Kalman filter. *Journal of Visual Communication and Image Representation*, 17(6):1190–1208.

Wilkinson, D. J. (2011). *Stochastic modelling for systems biology*. CRC press.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pages 229–256.

Wilson, P., D'Agostino, R., Levy, D., Belanger, A., Silbershatz, H., and Kannel, W. (1998). Prediction of coronary heart disease using risk factor categories. *Circulation*, 97(18):1837–1847.

Wingate, D. and Weber, T. (2013). Automated variational inference in probabilistic programming. *ArXiv e-prints*.