Advances in Monte Carlo Variational Inference and Applied Probabilistic Modeling

Advances in Monte Carlo Variational Inference and Applied Probabilistic Modeling

a dissertation presented

by

Andrew Colin Miller

to

The John A. Paulson School of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

April 2018

Dissertation advisor: Professor Ryan P. Adams Andrew Colin Miller

Advances in Monte Carlo Variational Inference and Applied Probabilistic Modeling

Abstract

Galvanized by the accelerated pace and ease of data collection, researchers in more and more disciplines are turning to large, heterogeneous datasets to answer scientific questions. Divining insight from massive and complex data, however, requires flexible models and efficient inference of meaningful factors of variation. This thesis develops new statistical models and methods to help practitioners answer quantitative questions and more efficiently explore their data.

The first part of this thesis presents three applied probabilistic modeling case studies in a diverse set of domains: astronomy, healthcare, and sports analytics. For each application we develop a probabilistic model for high-dimensional observations to address a particular goal—to make robust, portable predictions, find latent structure, or to organize and visualize interpretable factors of variation in the data. Guided by these examples, we discuss the common challenges of specifying interpretable-yet-flexible probabilistic models in applied settings.

Motivated by the challenges of applying probabilistic models to large datasets, the second part of this thesis develops new algorithms for approximate Bayesian inference. We focus on improving variational inference, a widely used class of approximation algorithms. We develop two new techniques to improve the accuracy and computational efficiency of variational inference methods. We further generalize one technique into a class of computationally efficient Monte Carlo estimators.

Contents

Listing of figures

Acknowledgments

I am deeply grateful to my advisor, Ryan Adams. Working closely with Ryan has been a fun and challenging and immensely rewarding experience. With Ryan's guidance, my time in graduate school reshaped my perspective on research, problem solving, and the role of statistics and machine learning in a broader scientific and societal context.

The HIPS lab was an enriching environment for a young graduate student, and I am fortunate to have met and acquired so much from its members, including Diana Cai, David Duvenaud, Mike Gelbart, José Miguel Hernández-Lobato, Jonathan Huggins, Scott Linderman, Dougal Maclaurin, Shamim Nemati, Yakir Reshef, Oren Rippel, Jasper Snoek, and Albert Wu. I would like to single out and thank Matt Johnson, who, on many occasions, patiently detailed a technical concept or discussed a research idea with an unflagging and infectious enthusiasm.

I am indebted to the members of the XY Research group, Alex D'Amour, Dan Cervone, Alex Franks, Kirk Goldsberry, and Luke Bornn. My statistics education is rooted in discussions dissecting basketball analytics with this group.

I would also like to thank Finale Doshi-Velez, who welcomed me into her research group, along with other members of the broader ML community at Harvard, including Mike Hughes, Arjumand Masood, Deborah Hanus, Taylor Kilian, Omer Gottesman, Andrew Ross, Yoon Kim, and Jon Malmaud.[1]

My time in graduate school has been deeply collaborative, and I am fortunate to have worked closely with a mix of researchers with an eclectic range of expertise, including Nick Foti, Jeff Regier, Jon McAuliffe, Sendhil Mullainathan, and Ziad Obermeyer.

I want thank my parents, Chris and Timm, and my sister, Ashley, whose support over the last six years never wavered (that much). Though my family always stressed the importance of education, I am sure they are surprised I took that advice so literally.

And finally, thank you, Cassie.

[1] And Sam Wiseman, I guess.

The best thing about being a statistician is that you get to play in everyone's backyard.

John Tukey

1

Introduction

THE ANALYSIS of noisy and uncertain data is an exercise common to nearly every field of scientific inquiry. The resolution of a scientific question requires weighing evidence associated with alternative hypotheses. An astronomer might ask, what is a quasar's redshift[1] given a noisy image? More specifically, how plausible is a particular value of redshift out of all possible values given our observation (visualized in Figure 1.1)? How can we reliably estimate such quantities for hundreds of thousands of objects? Answering these questions raises challenges that are physical, statistical, and computational in nature.

Alternatively, a scientist may want to explore a large dataset for previously unknown

[1] Redshift is a physical property of a source (e.g. a star, galaxy, quasar) that characterizes cosmological expansion observed between the source and the Earth.

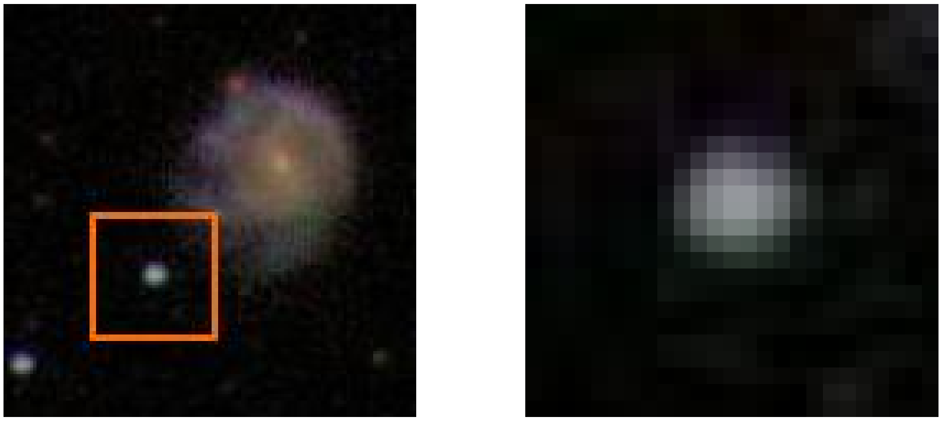
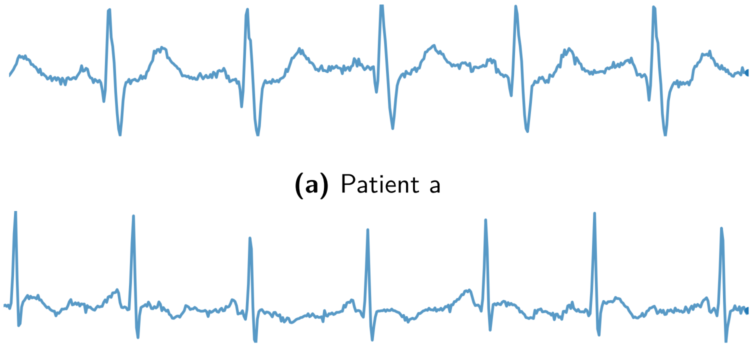

(a) Small SDSS patch **(b)** Zoom in (25×25 pixels)

Figure 1.1: Example photometric image from the Sloan Digital Sky Survey (SDSS) [Kent et al.]. Each pixel in each band represents a noisy photon count, conveying some information about the physical parameters of the imaged source. How can we reliably measure these parameters of interest, including our uncertainty about them?

patterns. For example, a doctor might want to systematically inspect a large collection of electrocardiogram tracings, two examples of which appear in Figure 1.2. This researcher may want to describe how these tracings vary within a patient (e.g. from beat to beat) and between patients. Further, they may want to discovery dimensions of variation associated with patient types. For instance, some patients may have paroxysmal atrial fibrillation[2] and exploring the variation between these and healthy patients could shed light on the underlying mechanism that causes this abnormality. How can we systematically explore variation relevant to the researcher's goals? How can we explore variation associated with a massive number of patients?

This thesis develops new statistical models and methods to help scientists answer quantitative questions and more efficiently explore their data—to perform better data analysis. We develop models of complex phenomena to measure an otherwise immeasurable quantity of interest— e.g. a prediction, a counterfactual (or potential) outcome, a correlation, a treatment effect, or unobserved structure that explains variation in the observed data.

[2] Paroxysmal atrial fibrillation is a classification that indicates a patient goes in and out of atrial fibrillation, a heart arrhythmia.



(a) Patient a

(b) Patient b

Figure 1.2: Example electrocardiogram traces. How can we characterize within-patient variation, and between-patient variation? How can we measure variation associated with certain patient-types?

Reliably estimating some quantity or finding a meaningful decomposition of the variation of a dataset for exploration (among other data analysis tasks) can be challenging. This task typically requires us to explain some high-dimensional, possibly noisy signal in terms of simpler latent structure. For example, a single-lead electrocardiogram (EKG) tracing is a high-dimensional (i.e. thousands of samples) observation of real-valued voltages that encodes the structure of a patient's cardiac cycle—the quasi-periodicity of the heart rate and the morphology of each individual cycle. Explaining an EKG observation in terms of latent structure allows us to create simpler, more intuitive representations that explain the variation in our observations at a level of abstraction useful to cardiologists. In order to make these representations more meaningful, we want this latent structure to reflect scientific knowledge of the underlying phenomenon—the physical process by which the data come to be measured. By incorporating known and invariant information, we can specify more portable models—models not tied to a particular statistical sample, but can make reasonable predictions when observing the phenomenon within a new context.

We also want to quantify our uncertainty. Typically, the analysis of data is in support of a decision—e.g. to make a diagnosis, select a treatment, or collect more data. If the information in the data is insufficient to answer a query, our model should reliably report this uncertainty. We also want to interpret the learned structure of a model. Particularly when the goal is scientific discovery, a model that maps an input to an output via some black-box will be insufficient for our understanding the phenomenon. When specifying models, there will be a tension between flexibility of model components and interpretability, and navigating this tradeoff is a constant challenge when working on applied modeling problems.

Probabilistic modeling is a natural framework to address this set of challenges. A probabilistic model defines a simplified *data generating procedure* in terms of random variables—observed data, global parameters, and latent variables. These random variables have parameterized relationships that describe their dependence structure. This framework enables the user to specify interpretable, scientifically-informed latent variables. Encoding this structure in a probabilistic model allows us to use statistical inference to infer these hidden quantities, which can then be used to make predictions, measure unobserved quantities, report uncertainties—to answer quantitative questions about the underlying phenomenon.

A probabilistic model specifies the relationship between each random variable, hidden or observed. However, variation in high-dimensional signals (e.g. electrocardiogram tracings or images) can be difficult to prescribe in a generative way. For these high-dimensional signals, advances in flexible function approximation (e.g. deep learning) has refined a set of flexible modeling tools that can approximate difficult-to-prescribe relationships—e.g. deep neural networks, deep generative models, and Gaussian processes (among other non-linear function approximators). However, balancing the flexibility of these model components with the interpretability of simpler, parametric components can be challenging. For a particular model, the domain and the task typically call for a level of abstraction for the model to operate. This can be dictated by particular physical quantities we need to infer or by an existing understanding of the phenomenon that we wish to augment.[3] We will see examples of this balance in Chapter 3.

Additionally, the application of a probabilistic model to a large dataset presents a computational challenge—statistical inference. The goal of statistical inference is to characterize all of the hypotheses (e.g. setting of model parameters) that are consistent with the observed data. Statistical inference combines the simplified representation of the underlying phenomenon posited by the model with the information in the data—the probabilistic model lays the scaffolding and the process of statistical inference fills in the details. Making statistical inferences, however, can be computationally challenging, and often requires devising tractable approximations that trade off compute time and statistical accuracy.

This thesis addresses some of these challenges associated with building interpretable-yet-expressive probabilistic models, and performing computationally tractable inference. In the first part of this thesis, we describe models for data in a variety of application domains. The common theme is that these models that balance high-dimensional, non-parametric components with interpretable probabilistic structure.

In the second part of this thesis, we develop new approximate Bayesian inference techniques that improve upon existing variational inference algorithms. We devise a method that allows variational inference approximations to become more expressive. We also develop a new estimator that makes a general class of variational inference algorithms more computationally efficient. We further generalize our reduced variance estimator, enabling its use in more general approximations, and characterize its variance properties.

[3] As a concrete example, the trajectories of basketball players are not often reasoned about as an autoregressive process or a linear dynamical system, but as a sequence of distinct actions at a level of abstraction discussed by coaches and players—e.g. a baseline cut, a V-cut, or a pick and roll.

1.1 Summary of contributions

Chapter 2 Chapter 2 reviews some of the foundational concepts upon which this thesis builds. We describe probabilistic modeling and its component parts, including common probability distributions and their properties, modeling techniques, inference algorithms, model checking and validation, and common difficulties and pitfalls.

Chapter 3 In Chapter 3 we present a selection of data analysis and applied modeling projects. We will look closely at three modeling case studies for data in (i) astronomy, (ii) medicine, and (iii) sports analytics. Though the applications are varied, these statistical models share similar approaches to describing sources of interpretable parametric variation as well as difficult-to-prescribe variation, non-parametric variation. These examples highlight important principles of statistical modeling, challenges with modern data sources, and the need for reliable inference algorithms.

Chapter 4 Variational inference is a widely used and scalable approach to approximate Bayesian inference. However, variational inference algorithms can be frustrated by inexpressive families of approximating distributions. This can have a variety of negative consequences, including local minima in the objective surface, under-dispersed marginal variances, and poor estimation of the marginal likelihood. These drawbacks can be mitigated by using a more expressive family of approximating distributions. Using expressive variational families can be challenging—the tractability of the typical variational objective and number of free variational parameters can make optimization practically difficult or analytically intractable. In Chapter 4 we present a method for fitting expressive variational approximations by greedily adding capacity to our existing variational family.

Chapter 5 When probabilistic models are non-conjugate (i.e. in general), the variational objective can be analytically intractable to compute. This is because the variational objective (and its gradient) is an expectation with respect to the approximate distribution, and a non-conjugate models may not admit an analytical expression. Variational inference algorithms for non-conjugate models often rely on stochastic gradient estimators to optimize the variational objective—practitioners compute relatively cheap Monte Carlo estimates of the gradient as a subroutine for a stochastic optimization algorithm. However, when these gradient estimators are too noisy, typical gradient-based optimization routines must take smaller steps to reliably increase the variational objective. This makes variational inference more time consuming or more brittle. In Chapter 5 we present a method for reducing the variance of these gradient estimators in a computationally inexpensive way, resulting in faster and more stable learning of variational approximations.

Chapter 6 Monte Carlo estimation underpins many statistical algorithms. Drawing samples from a probability distribution, and computing the sample average of some test function is a general way to approximate an intractable integral. The quality of an unbiased Monte Carlo estimate is a function of its variance—the lower the better. In many situations, however, Monte Carlo estimators do not take advantage of common structure—the smoothness and curvature of the test function; the known moments of the probability distribution. We present a simple and computationally inexpensive way to reduce the variance of a Monte Carlo estimator with a few additional assumptions, based on the relationship between Taylor expansions and moment generating functions. We study the variance of our estimators, and characterize a condition under which we are guaranteed to perform no worse than the original estimator. We also examine empirically inference algorithms for which our new estimator can improve performance.

Chapter 7 In Chapter 7 we conclude the thesis with a summary of contributions and a discussion of directions of future research.

2

Background

For a particular application, our focus is on building a probabilistic model—a distillation of useful information contained within a large set of data. To do this, a probabilistic model encodes a "generative story" for how a set of data came to be measured—a coarsening of the physical process that generated our measurements, explicitly specifying factors of "important" and "unimportant" variation.[1]

Specifying a probabilistic models, as the name suggests, draws upon probability theory and statistics for its core components. Probability theory provides a calculus that we can use to represent and reason about our uncertain measurements. As probabilistic modeling is rooted in probability and statistics, this chapter will detail some important

[1] The the definition of "important" will be domain (and task) specific—the distinction between what is "signal" and what is "noise".

concepts in probability and statistics that will be used throughout this thesis.

Using a probabilistic model in practice typically introduces a computational challenge in the form of statistical inference—the process by which we generalize properties of a sample to a population through a probabilistic model. Algorithms for statistical inference present a computation-accuracy tradeoff—perfect characterization of model uncertainty is often computationally intractable, and we rely on approximate methods to make practical use of probabilistic models.

The following sections describe the component parts of probabilistic models and probabilistic inference algorithms. We start with a light overview of probability theory and a sampling of important properties of random variables. We then describe some probabilistic model building blocks. The second part of this chapter focuses on statistical inference methods and some of their properties.

2.1 Probability: Light Overview

Probabilistic models are a composition of primitives—random variables and their relationships. As such, some light background in probability theory is necessary for understanding both the construction of probabilistic models and the algorithms we use for inference.

2.1.1 Random Variables and Distributions

A random variable is a deterministic function from an underlying random event space to some measurement space (e.g. real numbers, the unit interval, discrete classes, etc.). A probability space is defined by the tuple $(\Omega, \mathcal{B}, \mathbb{P})$, where

- Ω is the sample space, with $\omega \in \Omega$ some outcome. An outcome ω conceptually represents "the state of the world" at the finest resolution measurable, and Ω represents all possible "realities" that could be measured.
- $\mathcal{B}$ is a collection of measurable events, where each event is a set of elements in Ω for which we define probability measure (i.e. a sigma algebra).

- $\mathbb{P}$ is a probability measure that maps events in $\mathcal{B}$ to values in $[0, 1]$, and allocates exactly one unit of probability to all measurable events, $\sum_{b \in \mathcal{B}} \mathbb{P}(b) = 1$.

A real-valued random variable X is a map $X : \Omega \mapsto \mathbb{R}$, and the distribution (or law) of the original space allows us to define a probability measure over the random variable space X

$$P(X \in A) = \mathbb{P}(\omega \in X^{-1}(A)) \tag{2.1}$$

for some set A, and the *pre-image* of A, $X^{-1}(A)$, defined as the set of events in Ω that map to A under the transformation X.

The intuitive purpose of this formalism is to push the “randomness” or uncertainty into the abstract event space, allowing us to analyze deterministic mappings. Within this thesis, we will operate at the level of the random variables themselves. That is, we will typically define a probabilistic model with random variables that correspond to observed measurements or latent structure. However, the abstract event space of random variables is an instructive formalism—we view our measurements as a coarsening of the underlying “true state” of the world, and our level of abstraction for such a coarsening will strongly influence how interpretable, computationally tractable, and ultimately useful our model will be. Further, the principles of a deterministic mapping of randomness, and meticulous preservation of measure applies to random variables themselves, which we address in more detail in Section 2.1.4.

Under mild conditions, a random variable X admits a *probability density function*, denoted $p_X(\cdot)$, which relates to the probability distribution measure via integration

$$P(X \in A) = \int_{x \in A} p_X(x) dx \,. \tag{2.2}$$

When X lives on a discrete space, $p_X(x)$ is often referred to as the *probability mass function*.

We can also define the *cumulative distribution function* (CDF) of a univariate random variable, denoted $P_X(x)$, as

$$P_X(x) = P(X < x) \tag{2.3}$$

$$= \int_{x' < x} p_X(dx)\,. \tag{2.4}$$

Within a probabilistic model, the probability density and cumulative distribution functions are the primitives with which we define fundamental model objects: the likelihood, prior, and posterior distributions, which we describe in further detail in Section 2.2.1.[2] Probability theory is a rich field that provides the understanding and tools to manipulate random variables within a statistical modeling framework. See Williams [1991] for a more thorough treatment of probability theory.

2.1.2 Some properties of distributions

This section describes a handful of useful properties of random variables that we will use when formulating probabilistic models for data and deriving statistical inference algorithms. Consider a random variable $X \sim P_X$ on a general space with probability density (or mass) function given by $p_X(x)$.

Expectations and Moments

Some properties of random variables formalize intuitive questions about the variable. If we were to sample X, what is a "typical" value that we might expect? What *range* of

[2] Within this thesis, we will try to maintain a consistent capitalization scheme for random variables:

- X: a random variable with distribution P_X
- x: a realization of X with density $p_X(x)$ and cdf $P_X(x)$

Further, it will sometimes be convenient to use the density as a stand in for the entire distribution, e.g. $\boldsymbol{x} \sim p_X(\boldsymbol{x})$.

values can we expect? Given a large sample, would we see more samples in the higher range, or the lower range (or is the distribution symmetric)? The moments of a random variable give us a way to precisely characterize these properties. The most used and manipulated moment of a random variable is its first, also known as its expectation.

Definition 2.1.1 Expectation . The expectation of a random variable X is $\int_{x\in\mathcal{X}} xp_X(x)dx$, and is denoted $\mathbb{E}[X]$ or $\mathbb{E}_{P_X}[X]$.

Also referred to as the mean of X or first moment, the expectation of a random variable is a fundamental property used in many types of inference algorithms and model definitions—it will be used throughout this thesis.

Definition 2.1.2 Moments . The n^{th} moment of a distribution X is given by $\mathbb{E}[X^n]$.

The moments of the distribution are expectation of the powers of that random variable. The second moment is related to the variance of the random variable (also known as the centered second moment)

$$\begin{aligned}
\mathbb{V}[X] &\triangleq \mathbb{E}[(X-\mu)^2] && (2.5)\\
&= \mathbb{E}[X^2+\mu^2-2X\mu] = \mathbb{E}[X^2]+\mu^2-2\mu^2 && (2.6)\\
&= \mathbb{E}[X^2]-\mu^2 && (2.7)
\end{aligned}$$

where $\mu \triangleq \mathbb{E}[X]$ is common shorthand for the first moment.

Definition 2.1.3 Moment-Generating Function . The moment-generating function of a random variable X is defined

$$M_X(t) = \mathbb{E}\left[e^{tX}\right] \ , t \in \mathbb{R} \tag{2.8}$$

While its use may not be immediately obvious, a moment-generating function is an alternative characterization of the distribution of X that admits a recipe for computing each moment $\mathbb{E}[X^n]$ via differentiation

$$\mathbb{E}[X^n] = \frac{\partial^n M_X}{\partial t^n}(0)\,. \tag{2.9}$$

In words, the n^{th} moment of X can be computed by evaluating the n^{th} derivative of the moment-generating function at $t = 0$. We will use the moment-generating function and efficient automatic differentiation software to construct estimators with better variance properties than typical Monte Carlo estimators, which we describe further in Chapter 6.

The average of a sample of random variables has an important convergence property that relates to the expectation of the distribution.

Theorem 2.1.1 Strong Law of Large Numbers . Given a sequence of independent random variables $X_1, X_2, \dots$ such that

$$\mathbb{E}[X_k] = 0\,, \quad \mathbb{E}[X_k^4] \leq K \;\text{ for all } k \tag{2.10}$$

Let $S_n = X_1 + \cdots + X_n$, then

$$\frac{1}{n} S_n \to 0\,, \text{almost surely} \tag{2.11}$$

The law of large numbers tells us that the average of realized samples will converge to the expectation of the distribution over X. Proof and further details can be found in Chapter 7 of Williams [1991]. The law of large numbers directly motivates the use of Monte Carlo estimators—sample-based estimators of expectations with respect to a random variable (described further in Section 2.4.1. It (and convergence rates) also illustrates the computation-efficiency tradeoff of these estimators—more samples will more likely provide a better estimate, but at an additional collection or computational cost.

Conditioning and Independence

Consider two random variables, X and Y. The sum and product rule are two fundamental manipulations of their joint distribution. These manipulations will be useful when specifying a probabilistic model, and when inferring latent structure.

Definition 2.1.4 Product Rule . The product rule of probability describes how joint distributions can be composed of conditionals and marginals. Given random variables X and Y, we can factor the joint distribution

$$\begin{aligned} P(X,Y) &= P(X)P(Y|X) && (2.12) \\ &= P(X|Y)P(Y)\,. && (2.13) \end{aligned}$$

Definition 2.1.5 Sum Rule . The sum rule of probability describes how joint marginals relate to each other via the conservation of measure

$$P(X) = \sum_{Y} P(Y,X) \tag{2.14}$$

Definition 2.1.6 Bayes' Rule . Bayes' rule describes how conditional distributions in both directions relate to each other.

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{2.15}$$

Bayes' rule is a consequence of the sum and product rules of probability.

Definition 2.1.7 Independence . Two random variables X and Y are independent if they contain no information about each other, that is

$$X \perp Y \implies P(X,Y) = P(X)P(Y)\,. \tag{2.16}$$

The CDF and PDF also factorize for independent random variables.

Information and Divergence

The *entropy* of a random variable is a measure of how much information that random variable carries.

Definition 2.1.8 Shannon Entropy . The Shannon entropy of a random variable X is given by

$$H(X) = -\mathbb{E}_{X \sim P_X} \left[\ln p_X(X) \right] \tag{2.17}$$

For a continuous-valued random variable, this is often referred to as *differential entropy.* Intuitively, the entropy can be thought of as the complexity of a random variable—the higher the entropy, the more information a random variable conveys. Consider a uniform random variable on a discrete space, $X(\omega) \in \{1, \ldots, C\}$. The underlying mechanism that generates X is as unpredictable as can be, so observing the value of X gives us a lot of information about the underlying event ω driving the observation. Now consider a random variable $Y(\omega) \in \{1, \ldots, C\}$ such that $P(Y = 1) = 1$ and $P(Y \neq 1) = 0$—that is, we always observe $Y = 1$. This variable has no information about the underlying event ω; it will always reveal a measurement of $Y = 1$. This random variable has very low entropy (in fact, zero entropy).

The *mutual information* between two random variables is a measure of how much information is shared between the two variables.

Definition 2.1.9 Mutual Information . The mutual information between two random variables X and Y is given by

$$\begin{aligned} I(X;Y) &= \mathbb{E}_{X,Y} \left[\ln p(X,Y) - \ln p(X)p(Y) \right] && (2.18) \\ &= H(X) - H(X|Y) = H(Y) - H(Y|X) && (2.19) \\ &= H(X) + H(Y) - H(X,Y) = H(X,Y) - H(X|Y) - H(Y|X) \,. && (2.20) \end{aligned}$$

where the *conditional entropy* is defined $H(X|Y) = \mathbb{E}_{x \sim P_X}\left[H(Y|X=x)\right]$.

Mutual information has the appealing property of *transformation invariance*—the mutual information between two variables remains the same under continuous transformations of each variable Kraskov et al. [2004].

Entropy and mutual information are fundamental properties of a random variables that are relevant to both the specification of models and statistical inference. The principle of maximum entropy [Jaynes, 1957] provides guidance for specifying the form of a probability model or the selection of a prior distribution [Jaynes, 1968]. Entropy, and its related information theoretic quantities (e.g. KL divergence and mutual information) appear in the variational objective for approximate Bayesian inference [Jordan et al., 1999, Wainwright and Jordan, 2008, Blei et al., 2017b]. The information bottleneck is another modeling principle that draws upon entropy and conditional entropy to learn useful representations of high-dimensional data [Tishby et al., 2000, Chechik et al., 2005, Tishby and Zaslavsky, 2015].

Random variables can have different distributions, and it is often desirable to quantify how the difference between the two distributions via some kind of *divergence*. A useful and commonly used divergence is the *Kullback-Leibler divergence*.

Definition 2.1.10 Kullback–Leibler divergence (KL divergence) . Given two random variables, X and Y, defined on the same space with probability density functions $p_X(\cdot)$ and $p_Y(\cdot)$, the KL divergence from X to Y is given by

$$\mathrm{KL}(X \mid\mid Y) = \mathbb{E}_{X \sim P_X}\left[\ln \frac{p_X(X)}{p_Y(X)}\right] = \int p_X(dx) \ln \frac{p_X(x)}{p_Y(x)} \tag{2.21}$$

Note that the KL divergence is not a symmetric measure of divergence—that is $\mathrm{KL}(X \mid\mid Y) \neq \mathrm{KL}(Y \mid\mid X)$ in general.

KL divergence is a natural measure of how different two distributions are, and arises in many areas; maximum likelihood can be interpreted as minimizing the KL divergence between an empirical distribution and the parametric family of distributions (via an information projection) [Murphy, 2012]; the Hessian of the KL divergence is the Fisher information metric that defines a statistical manifold, used to characterize the statistical manifold [Cover and Thomas, 2012]; and as the most popular divergence for variational inference, a class of approximate Bayesian inference algorithms [Jordan et al., 1999, Wainwright and Jordan, 2008, Blei et al., 2017b].

Example: Gaussian Random Variables

Perhaps the most useful family of random variables is the Gaussian family.[3] A Gaussian distributed random variable X is a continuous-valued random variable with probability density function given by

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{1}{\sigma^2}(x-\mu)^2\right) \tag{2.22}$$

and denoted $X \sim \mathcal{N}(\mu, \sigma^2)$.[4]

A multivariate normal random variable $X \in \mathbb{R}^D$ is a continuous-valued random vector with probability density function given by

$$p(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = |2\pi\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{2.23}$$

characterized by mean vector $\boldsymbol{\mu} \in \mathbb{R}^D$ and positive definite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times D}$

Normal random variables have many useful properties. We can analytically compute moments, differential entropy, mutual information and KL divergence. For random variables with a given mean and variance, they are the *maximum entropy* distribution, motivating their use within statistical models [Jaynes, 1957]. Further, the *central limit theorem* states that the sum of finite variance independent random variables tend toward

[3] Also referred to as the normal family, we will use Gaussian and normal interchangeably.

[4] When convenient we overload notation and use $\mathcal{N}(x|\mu, \Sigma)$ as a stand in for the probability density function of a normal random variable.

a normal distribution as their numbers grow, providing yet another motivation for their use modeling complex, emergent phenomenon [Williams, 1991, Section 18.4]. Normal random variables are also computationally straightforward to manipulate—we typically assume that drawing spherical normals (i.e. $\epsilon \sim \mathcal{N}(0, I)$) is a computational primitive. Given access to a spherical normal sampler, it is a simple operation to compute samples with any mean and covariance

$$\epsilon \sim \mathcal{N}(0, I_D) \tag{2.24}$$

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\epsilon \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{2.25}$$

where the matrix square root can be any matrix $\boldsymbol{C}$ such that $\boldsymbol{C}\boldsymbol{C}^\intercal = \boldsymbol{\Sigma}$. The Cholesky decomposition of $\boldsymbol{\Sigma}$ is frequently used for this computation.

Normal random variables are invariant under linear transformations—a linear function of a normal random variable remains normal with a new (easily characterized) mean and covariance. This is an immensely useful property that we will exploit in Chapter 6 to construct lower variance Monte Carlo estimators.

2.1.3 Mixture Distributions

A mixture distribution is constructed by taking a convex combination of component distribution probability density functions. A random variable X is distributed according to a mixture of component distributions $P_c(X)$ (each with density $p_c(x)$) if its density is

$$p(x) = \sum_{c=1}^{C} \rho_c p_c(x)\,, \text{ such that } \rho_c \geq 0 \text{ and } \sum \rho_c = 1\,. \tag{2.26}$$

Mixtures provide a way to construct a more expressive distribution out of simpler component distributions. For instance, a mixture of Gaussians allows us to construct distributions that have non-zero skew or multiple modes. The method developed in Section 4 builds upon mixture distributions.

2.1.4 Transformations of Random Variables

A deterministic function of a random variable is itself a random variable, and naturally we might want to characterize its distribution. If $X \sim P_X$, then what is the distribution of $F \triangleq f(X)$ for some invertible f?

$$p_X(x)dx = p_F(f(x))df(x) \tag{2.27}$$

$$p_F(f(x)) = p_X(x)\left|\frac{dx}{df(x)}\right| = p_X(x)\left|\frac{df}{dx}\right|^{-1} \tag{2.28}$$

This change of variables rule preserves local probability measure. As a simple example, imagine a uniform random variable X, on the interval $[0, 1]$. Its probability density function is simple; $p(x) = 1$ when $x \in [0, 1]$ and 0 otherwise. What is the probability density of a simple rescaling of X, $Y = 2 \cdot X$? Random variable Y is now a uniform variable on $[0, 2]$. However, $p(y)$ can't be 1 on that interval as the integral of $p(y)$ over $[0, 2]$ has to equal 1 (i.e. preservation of probability measure), and in this case integrating $p(y) = 1$ over $[0, 2]$ is 2. The change of variables formula tells us that we only need to compute the $\frac{dy}{dx} = 2$, and re-scale the density function $p(y) = p(x)2^{-1} = 1/2$ on the interval $[0, 2]$, which properly integrates to one.

The change of variables formula tells us that point-wise evaluation of the transformed density requires only local geometric information—we only need to evaluate the Jacobian determinant at x (or y). Carefully tracking this local change in volume ensures the global preservation of probability measure.

This change of variables formula is a commonly used tool when manipulating probabilistic models. In variational inference, the normalizing flows [Rezende and Mohamed, 2015] class of distributions is designed to exploit this relationship to construct flexible and tractable posterior approximations.

2.2 Probabilistic Modeling

A probabilistic model defines a coarsened data generating procedure that describes certain factors of variation that lead to our observation. A good probabilistic model can be quite useful—given a model that well-describes the observed data, we can manipulate the model to make predictions about future observations, to fill in missing observations, to reason about counterfactual situations, to detect and explore relationships among variables, among many other use cases.

Although defining a probabilistic generative model of data is agnostic to statistical paradigm, this thesis will focus on Bayesian inference. While we will focus on approximating Bayesian posteriors, model validation and comparison can (and should) make use of frequentist tests.[5]

2.2.1 Priors, Likelihoods, and Posteriors

A probabilistic model is defined by the observed and unobserved random variables used to describe observed data. Given observed data $\mathcal{D}$ and model parameters θ, a probabilistic model is specified by the *likelihood* and the *prior*. The *likelihood* is the conditional probability density of the data viewed as a function of model parameters

$$\ell(\theta) = p(\mathcal{D} \mid \theta) \,. \tag{2.29}$$

The *prior*, $p(\theta)$, is a distribution we place over model parameters, specified before we see any data (e.g. it defines a "reasonable" setting of parameters). Despite the sim-

[5] *Bayesian vs. Frequentist Inference* The frequentist paradigm considers model parameters to be "non-random"—that is, $\boldsymbol{\theta}$ are fixed population quantities—which simply means that we do not use a probability distribution to account for their uncertainty. Frequentist inference quantifies the uncertainty of our estimators with respect to replicates of future data. We note that there are inherent trade-offs for each approach [Jordan, 2009]. The Bayesian paradigm offers a recipe for defining a probabilistic model (including informative priors) and inferring a coherent posterior distribution. This, coupled with frequentist tests for model checking and validation provide a robust recipe for fitting useful models [Gelman and Shalizi, 2013].

plicity of the above presentation, each model component can be quite complex. The likelihood might encode know structure behind the true data generating procedure, or it might include a composition of flexible, non-linear maps (e.g. multi-layer perceptrons or neural networks); the prior may encode information from previous scientific studies—information we might already know about the population distribution.

The *posterior* distribution is defined by Bayes' rule

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})} \tag{2.30}$$

$$\propto \ell(\theta)p(\theta) \tag{2.31}$$

Given the assumptions baked into $\ell(\theta)$ and $p(\theta)$, the posterior distribution characterizes all of the information about θ that we can learn from our data $\mathcal{D}$. We can use the posterior distribution to make predictions, report uncertainties in our measurements, explore structured variation in our data—essentially answer any statistical query that our model supports. For this reason, the goal of statistical inference is to characterize the posterior distribution in some way. In a restricted class of models, this can be done exactly. More often, however, we will have to approximate $p(\theta \mid \mathcal{D})$.

2.2.2 Model Components

The prior and likelihood are made up of model components. The components of a probabilistic model describe either marginal distributions or conditional distributions between two groups of random variables. In the next few sections we give examples of some types of model components used in this thesis.

Example: Linear Regression

It is often necessary to describe the relationship between two variables, $X \in \mathbb{R}^D$ and $Y \in \mathbb{R}$ within a probabilistic model. Linear regression is one approach to parametrically specifying such a relationship. This model component assumes that the dependence of Y upon X can be described by a linear function

$$X^{\intercal}\beta = Y + \epsilon \tag{2.32}$$

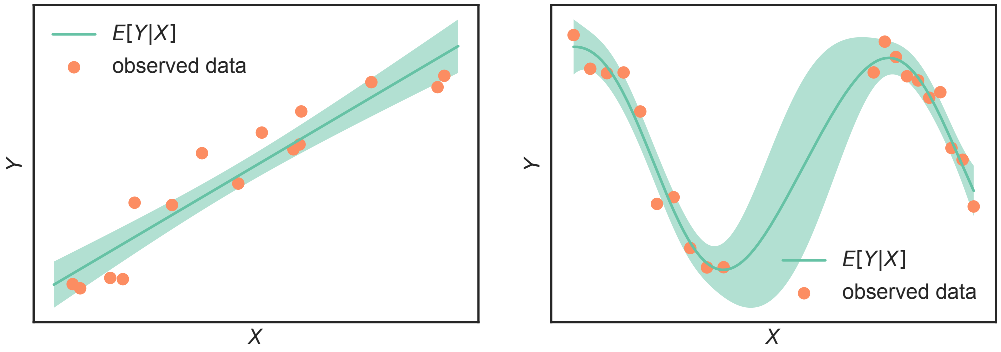
where β is some D-dimensional parameter (also called regression coefficients), and ϵ is a random variable with mean 0 and some variance σ_ϵ^2. We can make a distributional assumption about the noise term, for example $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$, which will influence our inference of unknown parameter β (and noise variance σ_ϵ^2).

Figure 2.1a depicts a linear relationship between X and Y with modest noise, σ_ϵ^2. The posterior distribution over linear relationships is depicted by the solid area (with the posterior mean $\mathbb{E}_{p(\beta|Y,X)}[\beta]$ depicted by the solid line). Linear relationships are easy to manipulate—the posterior distribution in this case can be characterized in closed form.

While a linear relationship may seem unrealistically simple, linear models can be quite useful—a linear relationship might not be as restrictive as it first appears. For instance, a linear relationship may be appropriate for a different choice of "basis" (e.g. parameterization or representation of variable X). As a concrete example, we use a linear component to model the complex, non-linear trajectories of basketball players (Section 3.3.6), using a non-linear basis as X. In fact, deep neural networks can be viewed as an approach to learn an appropriate basis X such that the relationship between this basis and the output is linear. In this way, a linear relationship can be used as a component within a more complex model.

Example: Gaussian Processes

The relationship between X and Y may be non-linear and difficult to specify with a known parametric form. In this case, a flexible, non-linear function approximation may be appropriate. A Gaussian process (GP) is a stochastic process that can be used as



(a) Linear regression example. **(b)** Gaussian process regression example.

Figure 2.1: Linear and Gaussian process regression examples. In both cases the solid area depicts the posterior distribution over the (noiseless) linear (left) and non-linear (right) functions (with two standard deviations visualized). Note that this is not a depiction of the predictive distribution, but of the posterior distribution over functions.

a prior distribution over a class of flexible functions. These functions can be used to model a potentially non-linear and difficult-to-prescribe relationship between two variables. Formally, a Gaussian process is a stochastic process, $f : \mathcal{X} \to \mathbb{R}$, such that any finite collection of random variables, $f(x_1), \dots, f(x_N) \in \mathbb{R}$, is distributed according to a multivariate normal distribution. GPs are frequently used as priors over unknown functions, f, where the random variables $f(x_1), \dots, f(x_N)$ correspond to evaluations of the function at inputs $x_1, \dots, x_N \in \mathcal{X}$. The covariance between any two outputs, $f(x_i)$ and $f(x_j)$, encodes prior beliefs about the function f; carefully chosen covariance functions can encode beliefs about a wide range of properties, including smoothness and periodicity.

There is a rich collection of covariance functions that can be used, and a calculus for composing them to create new covariance functions [Duvenaud, 2014]. One common covariance function that we will use in this thesis is the Matérn [Matérn, 1986] covariance function

$$k_{\text{Matern}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{\nu} r}{\ell} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} r}{\ell} \right) \tag{2.33}$$

where $r = |x_1 - x_2|$, and K_ν is a modified Bessel function. The parameter ν controls the smoothness and ℓ is the length scale of the function. The Matérn class is an example of a stationary covariance function—that is, the covariance is determined strictly by the distance between two points in the space $\mathcal{X}$. See Rasmussen and Williams [2006] for a thorough treatment of Gaussian processes in machine learning.

Figure 2.1b depicts a Gaussian process model for a non-linear relationship between X and Y. In this Figure, we visualize the posterior distribution over *functions* that relate the two variables—notice that the uncertainty over the function value grows as we move farther away from observed x values. Gaussian processes admit a closed form posterior distribution for $p(f|X, Y)$ when the noise model is Gaussian (and approximate methods exist when the noise model is non-Gaussian). One drawback is computation—inference in Gaussian process models require computation $O(N^3)$, where N is the number of observed X, Y pairs. This complexity makes scaling to massive datasets infeasible—as a result, approximate methods for scaling Gaussian processes is an active area of research.

Example: Clustering Models

One type of inductive bias we might want to encode in our model is that observations may come from some small number (relative do number of data observations) of clusters. That is, two similar observations may be similar because they belong to the same unobserved group (and two dissimilar observations may be dissimilar because they belong to different unobserved groups).

One way to encode this idea in a probabilistic model is to use an observation-specific latent variable that indicates cluster membership. For each cluster, a probability distribution describes what the members of each cluster tend to look like. Given data $\mathcal{D} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, a clustering latent variable model can be defined

$$Z_n \sim \mathrm{Cat}(\pi) \tag{2.34}$$

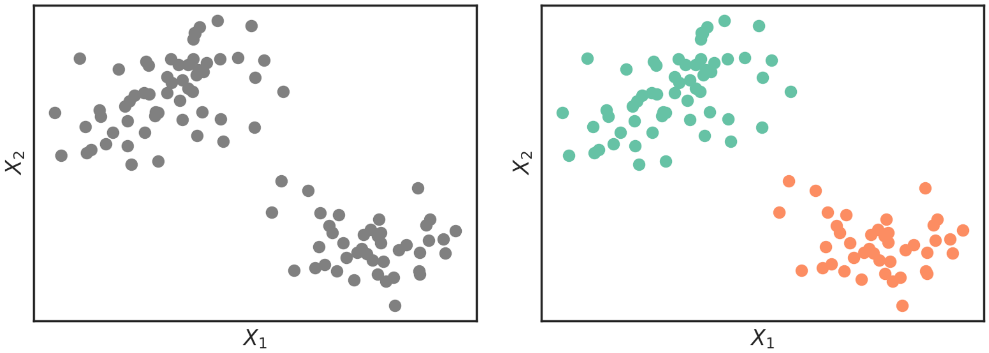
$$X_n \mid z_n \sim P^{(z_n)}(X) = \sum_{c=1}^{C} \mathbb{1}(z_n = c) P^{(c)}(X) \tag{2.35}$$

where $\mathbb{1}(Z_n = c)$ is an indicator that takes the value of 1 when its argument is true (e.g. $Z_n = c$). The probability density function of our observed data in this clustering model is a mixture distribution

$$\begin{aligned} p(\boldsymbol{x}_n) &= \sum_{c=1}^{C} p(\boldsymbol{x}_n, \boldsymbol{z}_n = c) && (2.36) \\ &= \sum_{c=1}^{C} p(\boldsymbol{z}_n = k) p(\boldsymbol{x}_n | \boldsymbol{z}_n = c) && (2.37) \\ &= \sum_{c} \pi_c p_c(\boldsymbol{x}_n) \,. && (2.38) \end{aligned}$$

When fitting a mixture model, the inferential goal is to estimate the component mixture probabilities, π, and component-specific parameters, e.g. parameters that characterize each distribution $p_c(\boldsymbol{x}_n)$. For example, $p_c(\boldsymbol{x}_n)$ is commonly chosen to be Gaussian, with component-specific parameters μ_c and σ_c^2. A common way to estimate these parameters is a procedure called *expectation maximization*, which we detail further in Section 2.3.1.

Figure 2.2 depicts this kind of clustering model and inference. Within X space, we see two unlabeled groups—we can use a clustering latent variable model to infer which group each element belongs to.



(a) Original, unlabeled observations.

(b) Result of a clustering model.

Figure 2.2: Clustering example (best viewed in color). *Left*: observations in the original space, locations x_1, x_2. *Right*: individual observations are labeled with their observation-specific cluster identity variable (green and orange).

Example: Latent Factor Models

High-dimensional data can be a challenge to model. One inductive bias that is widely applicable is that high-dimensional data do not vary meaningfully along each dimension—there are (sometimes quite large) correlations between dimensions that induce structured variation. This effectively lowers the overall dimension of the observed data.

One way to encode this idea is with a latent factor model. Concretely, $\boldsymbol{x} \in \mathbb{R}^D$ is a high-dimensional vector, we might imagine that there exists some basis $\boldsymbol{B} = \boldsymbol{B}_1, \ldots, \boldsymbol{B}_K$, $\boldsymbol{B}_k \in \mathbb{R}^D$ such that

$$\boldsymbol{w} \triangleq w_1, \ldots, w_K \sim p(w_1, \ldots, w_K) \tag{2.39}$$

$$\boldsymbol{x} = \sum_{k=1}^{K} w_k \boldsymbol{B}_k + \epsilon\,, \;\; \epsilon \sim \mathcal{N}(0, \sigma_n^2 I_D) \tag{2.40}$$

is a good description of the original vector $\boldsymbol{x}$ (i.e. σ_n^2 is low).[6] That is, our model posits

[6] Note that additive Gaussian noise may be appropriate for continuous-valued data, but this latent factor model construction generalizes to other noise models (e.g. Bernoulli, Poisson, Multinomial, etc.).

that there exists some (potentially unknown a priori) basis $\boldsymbol{B}$ that allows us to describe our high-dimensional observation with just a few basis weights, $\boldsymbol{w} = w_1, \ldots, w_K$ for $K \ll D$.

Modeling the dimensions of “useful” or “meaningful” variation is at the heart of probabilistic modeling—the representation $\boldsymbol{w}$ of $\boldsymbol{x}$ can often be a useful summary for data exploration, to define a meaningful distance between units, and the basis itself $\boldsymbol{B}$ can reveal meaningful (and potentially unknown) structure in the space that $\boldsymbol{x}$ lives. We can push this representation to be more useful by incorporating meaningful constraints—for instance we could force the basis vector $\boldsymbol{B}_k$ to take on only positive values. This may be a sensible constraint if the latent bases correspond to something physical (e.g. a spectral energy distribution), or if they are used to encode rates that must be positive (e.g. for a Poisson noise model). Models that have this (or a similar) form include probabilistic principal components analysis [Tipping and Bishop, 1999] and factor analysis, Latent Dirichlet Allocation [Blei et al., 2003], and in models presented in Section 3 as well as methods we describe in Section 4. Expectation maximization or its variational generalization is a common way to infer both the basis $\boldsymbol{B}$ and data-specific weights $\boldsymbol{w}$ in latent factor models.

Example: Deep Generative Models

Describing the generative process of a high-dimensional signal, such as an image, can be difficult. A deep generative model is another way to define a low-dimensional latent variable space that induces a distribution on the high-dimensional space. Unlike a linear latent factor model, a deep generative model typically defines a non-linear relationship between latent variable and observation, which can often admit a more efficient and useful representation of the data.

Given high-dimensional vector $\boldsymbol{x} \in \mathbb{R}^D$, a deep generative model defines the relationship

$$\boldsymbol{z} \sim p(\boldsymbol{z}) \ , \boldsymbol{z} \in \mathbb{R}^K \tag{2.41}$$

$$\boldsymbol{x} = f(\boldsymbol{z}; \boldsymbol{\theta}) + \epsilon \ , \epsilon \sim \mathcal{N}(0, \sigma_n^2 I_D) \tag{2.42}$$

where $f : \mathbb{R}^K \times \boldsymbol{\theta} \mapsto \mathbb{R}^D$ describes a non-linear mapping of our latent variable $\boldsymbol{z}$ to the data space, given by generative global parameter $\boldsymbol{\theta}$, such as a multi-layer perceptron or other functions differentiable in both arguments.

The mapping f is usually defined to be much more flexible than the linear function in a latent factor model, which can enable the representation $\boldsymbol{z}$ to be more compact than the weights from the previous section. More concretely, we can often achieve the same reconstruction error of the original data with a smaller value for K.

One drawback is that there is no longer a learned basis, $\boldsymbol{B}$ that can be interpreted in the data space. This is an example of trading off flexibility for interpretability—often direct inspection of $\boldsymbol{B}_k$ can tell you what variation is being captured by each latent basis in a latent factor model. For a deep generative model, inspection of samples along latent dimensions is an alternative way to understand the variation that the model encodes. Another drawback is that this non-linear model can be much harder to fit than a similar linear model. Though methods exist [Kingma and Welling, 2013, Rezende et al., 2014], reliably estimating $\boldsymbol{\theta}$ and shaping $\boldsymbol{z}$ given a set of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ data vectors is an open problem [Chen et al., 2018].

Example: Composing Model Components

Composing model components allows us to define models with richer and more expressive structure. As an example, we can compose a mixture model with a latent factor model to create a *mixture of factor analyzers* [Ghahramani and Hinton, 1996]. Consider the simple dataset depicted in Figure 2.3. The data clearly cluster into two groups *and* within each cluster there seems to be only one dimension of meaningful variation. Inferring cluster membership and these cluster-specific dimensions can be accomplished by composing a mixture model with a latent factor model. For each observation n, the data generating procedure can be specified

$$\boldsymbol{z}_n \sim \mathrm{Cat}(\pi) \tag{2.43}$$

$$\boldsymbol{w}_n | \boldsymbol{z}_n \sim p(w|\boldsymbol{z}_n) \tag{2.44}$$

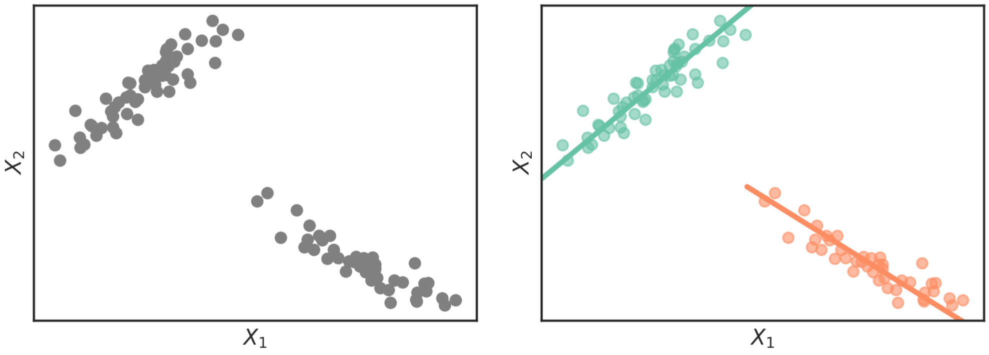
$$\boldsymbol{x}_n | \boldsymbol{z}_n, \boldsymbol{w}_n = \boldsymbol{w}_n^\intercal \boldsymbol{B}^{(\boldsymbol{z}_n)} + \epsilon_n \ \ , \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{2.45}$$

This model defines C clusters, and each cluster has its own cluster-specific basis $\boldsymbol{B}^{(c)}$. This enables us to capture both the grouping structure, and the low-dimensional structure within each group, which can be useful when modeling high-dimensional observations.

Both mixture and latent factor models are examples of *latent variable models*, a useful abstraction when defining model structure and inference algorithms. In general, a latent variable model draws a distinction between *local latent variables* (often denoted $\boldsymbol{z}$) and *global parameters* (often denoted $\boldsymbol{\theta}$). A local latent variable is an unobserved variable associated with a particular observation unit (e.g. the latent weights $\boldsymbol{w}$ or cluster indicator $\boldsymbol{z}$). Global parameters govern the structure shared by all of the data. This type of model structure not only gives us unit-specific latent structure, but also may give us some computational or algorithmic traction when inferring global parameters $\boldsymbol{\theta}$. We will see an example of the algorithmic advantage of some latent variable models in Section 2.3.1.

Probabilistic Modeling Discussion

This previous section has addressed ways to specify latent and conditional structure to describe the data generating procedure. This included ways to low-dimensional latent



(a) Original, unlabeled observations. **(b)** Result of the mixture of latent factors.

Figure 2.3: A mixture of latent factor models example (best viewed in color). *Left*: observations in the original space, locations x_1, x_2. These observations visually cluster into two groups, and within each group, only one of the two dimensions contains most of the variation. *Right*: individual observations are labeled with their cluster identity (blue and orange) and within each cluster we have inferred and labeled the dimension of maximum variation.

structure, latent clusters, autoregressive properties, linear and non-linear conditional relationships, and function smoothness as a prior. Composing these primitives is a useful approach for analyzing, exploring, and making decisions with a large, complex dataset. However, the specification of a useful model is one distinct challenge in the statistical analysis of data. Another fundamental challenge is inference—or the algorithms we use to distill the information in a dataset into the model we have specified.

2.3 Inference

Specifying a probabilistic model for data introduces a new problem—inference. In this section, consider a dataset $\mathcal{D}$, model parameters $\boldsymbol{\theta}$, a likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ defined by a data generating procedure, and a prior distribution $p(\boldsymbol{\theta})$. Further, assume the data $\mathcal{D}$ were generated according to the model, with some parameter $\boldsymbol{\theta}^{(true)}$. In practice, $\boldsymbol{\theta}^{(true)}$ is unobserved, and the goal of inference is to construct some estimate, $\hat{\boldsymbol{\theta}}$, from our data and model assumptions.

2.3.1 Maximum Likelihood Estimation

A widely used estimator of $\boldsymbol{\theta}^{(true)}$ is the *maximum likelihood estimate* (MLE). The principle behind the MLE is that we want to find the setting of $\boldsymbol{\theta}$ that makes our observed data the most probable—the setting that is most consistent with our observations. The maximum likelihood estimator is the value of $\boldsymbol{\theta}$ that maximizes the probability of observing our dataset $\mathcal{D}$ under the model assumptions

$$\hat{\boldsymbol{\theta}}^{(ml)} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})\,. \tag{2.46}$$

We can also incorporate a prior distribution over our model parameters, $\boldsymbol{\theta}$, which serves as a form of regularization. This leads to the *maximum a posteriori* (MAP) estimator

$$\hat{\boldsymbol{\theta}}^{(map)} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,. \tag{2.47}$$

Under certain conditions, the maximum likelihood estimator has a handful of desirable properties. Given certain assumptions it can be shown that the maximum likelihood estimator is *consistent* and *efficient* as sample size grows [Cramér, 1946, Chapter 33, Section 3]. By consistent, we mean that the estimator converges to the true model parameter, $\boldsymbol{\theta}^{(ml)} \to \boldsymbol{\theta}^{(true)}$ as the number of samples grows. By efficient, we mean that the estimator variance $\mathbb{V}(\boldsymbol{\theta}^{(ml)})$ achieves the *Cramér-Rao lower bound*, a lower bound on the variance that any (asymptotically) unbiased estimator can achieve. These two properties provide strong motivation to use maximum likelihood to estimate parameters, as opposed to alternative statistical estimators, such as the method of moments. There is, however, a computation-efficiency tradeoff—though the maximum likelihood estimator may theoretically have optimal properties, it may be intractable to compute and alternative estimators may be a better option. Maximum likelihood estimation also connects to empirical risk minimization, M-estimation [Van der Vaart, 1998, Chapter 5], among other optimization-based forms of statistical modeling and inference.

Expectation-Maximization

Expectation maximization is an algorithm for performing maximum likelihood (or MAP) inference for latent variable models. As discussed in the previous section, it is often useful to define a set of latent variables (e.g. cluster assignments, factor weights) within our model, which can be defined with respect to the *complete-data* likelihood

$$p(\mathcal{D}, \boldsymbol{z} \mid \boldsymbol{\theta}) = p(\mathcal{D} \mid \boldsymbol{z}, \boldsymbol{\theta}) p(\boldsymbol{z} \mid \boldsymbol{\theta})\,. \tag{2.48}$$

where $\boldsymbol{z}$ are our "latent variables" and $\boldsymbol{\theta}$ are "model parameters".[7] In this situation, the principle of maximum likelihood dictates that we maximize the *marginal likelihood* of our data

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \int p(\mathcal{D} \mid \boldsymbol{z}, \boldsymbol{\theta}) p(\boldsymbol{z} \mid \boldsymbol{\theta}) d\boldsymbol{z}\,, \tag{2.49}$$

where we marginalize out our latent variable $\boldsymbol{z}$. In some situations, however, the integral in Equation 2.49 is intractable—that is, we cannot even compute the likelihood of our data given this model, let alone maximize it. *Expectation-maximization* is a method for maximizing a general class of models of the form in Equation 2.48 [Dempster et al., 1977].

The EM algorithm starts with some initial setting of $\boldsymbol{\theta}$, $\boldsymbol{\theta}^{(0)}$. At every iteration n, the EM iterates between two steps: the E-step and the M-step.

- The E-step defines an objective function, the *expected complete-data log likelihood*,

[7] The only distinction here is that we're maximizing the likelihood with respect to $\boldsymbol{\theta}$, but not $\boldsymbol{z}$. The reasoning behind this choice might be that we care about the posterior distribution over $\boldsymbol{z}$, but not $\boldsymbol{\theta}$—for instance we may have a $\boldsymbol{z}$ for each data observation, but only one set of global $\boldsymbol{\theta}$ parameters. If we were to maximize over $\boldsymbol{z}$ in that situation, we may overfit our data and return a poor estimate of the distribution of our data.

that is a function of the posterior distribution over $\boldsymbol{z}$ given $\mathcal{D}$ and $\boldsymbol{\theta}^{(n)}$.

$$Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(n)}) = \mathbb{E}_{p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{(n)})} \left[\ln p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\theta}) \right] \tag{2.50}$$

- The M-step maximizes this objective function with respect to free parameters $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^{(n+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(n)}) \tag{2.51}$$

Note that the current setting $\boldsymbol{\theta}^{(n)}$ is fixed only with respect to the expectation—the free parameter $\boldsymbol{\theta}$ that appears within the expectation is the target of our optimization procedure. Also note that the Q function in Equation 2.50 would include a term $p(\boldsymbol{\theta})$ if we were using EM to do MAP estimation.

The Q function that EM defines is a lower bound to the marginal likelihood, $p(\mathcal{D}|\boldsymbol{\theta})$. We can see this by first deriving a general lower bound to the marginal likelihood

$$\begin{aligned}
\ln p(\mathcal{D}|\boldsymbol{\theta}) = \ln \int p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\theta}) d\boldsymbol{z} &= \ln \int q(\boldsymbol{z}) \frac{p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\theta})}{q(z)} d\boldsymbol{z} && (2.52) \\
&= \ln \mathbb{E}_q \left[\frac{p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\theta})}{q(z)} \right] && (2.53) \\
&\geq \mathbb{E}_q \left[\ln \frac{p(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\theta})}{q(z)} \right] \qquad \text{by Jensen's inequality} && (2.54)
\end{aligned}$$

where we have introduced a distribution $q(\boldsymbol{z})$ as a free parameter. Regardless of the choice of this distribution $q(\boldsymbol{z})$, the resulting right hand side function will lower bound the marginal likelihood.

How tight this lower bound is will be determined by our choice of $q(\boldsymbol{z})$. One sensible choice for q is the posterior distribution $p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta})$, which will actually result in a tight lower bound at that value of $\boldsymbol{\theta}$. This connects us back to the Q function in Equation 2.50.

At iteration n, we have a Q function

$$\ln p(\mathcal{D}|\boldsymbol{\theta}) \geq \mathbb{E}_{p(\boldsymbol{z}|\mathcal{D},\boldsymbol{\theta}^{(n)}} \left[\ln \frac{p(\mathcal{D},\boldsymbol{z}|\boldsymbol{\theta})}{p(\boldsymbol{z}|\mathcal{D},\boldsymbol{\theta}^{(n)})} \right] \tag{2.55}$$

$$= Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n)}) + H(p(\boldsymbol{z}|\mathcal{D},\boldsymbol{\theta}^{(n)})) \tag{2.56}$$

where we see that the lower bound to the marginal likelihood decomposes into our Q function, and an entropy term. This entropy term is fixed, so it can be ignored when performing the M-step.

A simple illustrative use case for EM is fitting a cluster model with a mixture distribution. Recall that a cluster assignment $\boldsymbol{z} \in \{1, \ldots, C\}$ can be viewed as a latent variable for each observation. Given a clustering model with Gaussian observations, we can write the generative model as

$$Z \sim \mathrm{Cat}(\pi)\,, \;\; \pi \in [0,1]^C, \sum_l \pi_c = 1 \tag{2.57}$$

$$\boldsymbol{x}|\boldsymbol{z} \sim \mathcal{N}(\mu_{\boldsymbol{z}}, \Sigma_{\boldsymbol{z}}) \tag{2.58}$$

where we have parameters $\boldsymbol{\theta} = \left(\pi, \{\mu_c, \Sigma_c\}_{c=1}^C\right)$. Given observations $\boldsymbol{X} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, and unobserved cluster assignments $\boldsymbol{Z} = \boldsymbol{z}_1, \ldots, \boldsymbol{z}_N$, the complete-data log likelihood is

$$\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}) = \ln \prod_n \prod_c (p(\boldsymbol{x}_n|\boldsymbol{z}_n)p(\boldsymbol{z}_n))^{\mathbb{1}(\boldsymbol{z}_n = c)} \tag{2.59}$$

$$= \sum_n \sum_c \mathbb{1}(\boldsymbol{z}_n = c) \left(\ln \mathcal{N}(\boldsymbol{x}_n|\mu_{\boldsymbol{z}}, \Sigma_{\boldsymbol{z}}) + \ln \pi_c\right) \tag{2.60}$$

where the indicator $\mathbb{1}(\boldsymbol{z}_n = c)$ keeps track of which cluster each observation n comes from, and incorporates that term into the likelihood accordingly.

E-step will involve computing the posterior distribution

$$p(\boldsymbol{z}_n = c|\boldsymbol{x}_n, \boldsymbol{\theta}^{(n)}) = \frac{p(\boldsymbol{x}_n|\mu_c^{(n)}, \Sigma_c^{(n)})}{\sum_{c'} p(\boldsymbol{x}_n|\mu_{c'}^{(n)}, \Sigma_{c'}^{(n)})} \triangleq \gamma_{n,c} \tag{2.61}$$

for each cluster. Because the number of clusters is finite, it is easy to normalize this distribution. The resulting Q function is straightforward to write down

$$\begin{aligned} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n)}) &= \mathbb{E}_{p(\boldsymbol{z}_n=c|\boldsymbol{x}_n,\boldsymbol{\theta}^{(n)})}\left[\sum_n \sum_c \mathbb{1}(\boldsymbol{z}_n = c)\left(\ln \mathcal{N}(\boldsymbol{x}_n|\mu_{\boldsymbol{z}} + \ln \pi_c\right)\right] && (2.62) \\ &= \sum_n \sum_c \gamma_{n,c}\left(\ln \mathcal{N}(\boldsymbol{x}_n|\mu_{\boldsymbol{z}} + \ln \pi_c\right) && (2.63) \end{aligned}$$

The M-step maximizes this Q function with respect to mixture parameters. For a Gaussian mixture model, the M-step is a weighted maximum likelihood solution which is easy to compute in closed form. This is a recurring theme in EM—in Chapter 3 we will look at a model that is essentially a mixture of linear regressors, which results in an M-step that can be solved with weighted least squares.

EM is an effective inference tool for the class of models that admit closed form expressions for the latent variable posterior $p(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta})$. However, only a limited class of models admit this posterior and the expected complete-data log likelihood in an easily-maximized closed form. In these instances, this direct application of EM may not be feasible, however applying this same principle—maximizing a lower bound to the marginal log likelihood of the data—will yield useful inference algorithms.

Generalizations of EM, such as variational EM and other free-energy optimization approaches, exist for more general models and inference outcomes (e.g. full Bayesian inference over both $\boldsymbol{\theta}$ and $\boldsymbol{z}$). However, the main idea behind EM is the iterative imputation of missing data—we estimate properties of our $\boldsymbol{z}$ given our data and our current best guess at a model (either via analytic expectations or sampling), and then use this inference to improve our best guess at the model. This iterative approach to fitting latent variable models conceptually underpins many inference algorithms—for example, Gibbs sampling.

In some cases, the marginal likelihood is computable (i.e. the integral in Equation 2.49 is tractable). In these instances, the user is presented with a choice for maximum-likelihood optimization—do EM or directly optimize the marginal likelihood. There are tradeoffs between the two approaches that are model and data dependent, and algorithms that switch between the two have been explored [Salakhutdinov et al., 2003]. In cases where direct optimization of the marginal likelihood is impossible, EM is a useful tool.

In Chapter 3, we use expectation maximization to fit a functional clustering model to a massive dataset, and use a variational generalization to fit a deep generative latent variable model (component).

2.4 Bayesian Inference

In some applications we will want to estimate the uncertainty of all parameters, including "global" parameters in hierarchical latent variable models. The Bayesian paradigm uses conditional probabilities to characterize uncertainty about model parameters—uncertainty is not with respect to replications of data, but with respect to the posterior distribution. The posterior distribution is a simple consequence of Bayes' rule. Given a prior $p(\boldsymbol{\theta})$ and a likelihood for data $p(\mathcal{D}|\boldsymbol{\theta})$, the posterior distribution is

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \,. \tag{2.64}$$

By assuming a parametric model and using the rules of conditional probability, the posterior distribution distills all of the information about $\boldsymbol{\theta}$ that we can hope to learn from $\mathcal{D}$. Making predictions, reporting credible intervals, exploring inferred structure—just about any desirable operation can be performed by computing some functional of the posterior distribution. These model queries can be computed as expectations with respect to the posterior distribution

$$\mu_f = \mathbb{E}_{p(\boldsymbol{\theta} | \mathcal{D})} \left[f(\boldsymbol{\theta}) \right] . \tag{2.65}$$

The main drawback with Bayesian inference is computational tractability—computing posterior expectations in general is infeasible. The main goal of approximate Bayesian inference methods is approximating these expectations in a computationally efficient way. Herein lies the computation-accuracy tradeoff—some methods can take a long time and eventually produce a very accurate estimator, while other methods can be quick, but produce an inaccurate estimator. Navigating this computation-accuracy tradeoff is a challenge for applied Bayesian practitioners.

2.4.1 Monte Carlo Estimators

As the target of Bayesian inference are expectations against the posterior distribution, a general purpose way to estimate these functions is via Monte Carlo estimation. A Monte Carlo estimator uses samples drawn from the distribution with which the expectation is taken, and simply computes a sample average of the function within the expectation. With respect to a posterior distribution, these estimates can be computed

$$\boldsymbol{\theta}^{(\ell)} \sim p(\boldsymbol{\theta} \mid \mathcal{D}) \ , \text{ for } \ell = 1, \dots, L \tag{2.66}$$

$$\hat{\mu} = \frac{1}{L} \sum_{\ell} f(\boldsymbol{\theta}^{(\ell)}) \, . \tag{2.67}$$

These estimators will be useful in combination with Markov chain Monte Carlo or variational inference samples to compute arbitrary expectations against the posterior distribution.

The quality of an estimator is typically measured by its variance. Under certain conditions, the variance of these Monte Carlo estimators decreases as more samples are incorporated (as one would expect given the law of large numbers). If each sample is independent with variance $\mathbb{V}(f(b\theta)) = \sigma^2$ the variance decreases linearly in L

$$\mathbb{V}(\hat{\mu}) = \mathbb{V}\left(\frac{1}{L}\sum_{\ell=1}^{L} f(\boldsymbol{\theta}^{(\ell)})\right) = \frac{1}{L^2}\sum_{\ell=1}^{L}\mathbb{V}(f(\boldsymbol{\theta}^{(\ell)}) \tag{2.68}$$

$$= \frac{1}{L^2}\sum_{\ell=1}^{L}\sigma^2 = \frac{1}{L}\sigma^2 \,. \tag{2.69}$$

When samples are correlated, the variance of the estimator can have worse properties—intuitively, each additional correlated sample provides less information than an additional independent sample. While low-variance estimators are desirable, simply incorporating more samples may be infeasible due to the cost of collection or the cost of computing the estimator itself. In Chapter 5 we improve upon typical Monte Carlo estimators to more efficiently solve a variational inference problem.

2.5 Conjugacy

For a certain classes of models, the posterior distribution can be analytically characterized. When a *conjugate* prior-likelihood pair is used to model data, the posterior distribution can be characterized analytically. A conjugate prior-likelihood pair is formed when the posterior distribution (over model parameters) remains in the same family as the prior distribution (over those same model parameters). As a concrete example, consider a model parameter that is a the mean of a Gaussian likelihood with a Gaussian prior

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{2.70}$$

$$= \mathcal{N}(\mathcal{D}|\boldsymbol{\theta}, \sigma^2)\mathcal{N}(\boldsymbol{\theta}|\mu_0, \sigma_0^2) \,. \tag{2.71}$$

Because of the functional form of the likelihood and prior, the functional form of the resulting posterior distribution remains an exponentiated quadratic with respect to $\boldsymbol{\theta}$. This implies that the posterior distribution is Gaussian, and the normalizing constant is straightforward to compute.

Conjugate prior-likelihood pairs need to be the same family—some exampls include Beta-Bernoulli, Poisson-Gamma, Pareto-Gamma [Gelman et al., 2014]. Conjugacy is closely tied to exponential family models—models with finite *sufficient statistics* vectors[Casella and Berger, Theorem 6.2.10].

More complex models can have *conditional conjugacy* structure—interacting components can have conjugate prior-likelihood relationships. This conditional conjugacy structure can admit efficient inference algorithms. For example, fast variational inference updates can be devised by iterating between conditionally conjugate components, or easy-to-sample conditional distributions appropriate for a Gibbs sampler can be computed. These sort of inference strategies are discussed in the next two sections.

2.6 Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a class of methods designed to simulate samples from unnormalized probability distributions, and it is frequently used to estimate posterior summaries in computationally intractable Bayesian models [Brooks et al., 2011]. MCMC methods are rooted in the observation that, though independent posterior samples are infeasible to compute, *dependent* samples can be instantiated and used as a surrogate to estimate posterior expectations.

MCMC draws samples from a stochastic process on the parameter space whose equilibrium distribution is the posterior distribution of interest. The empirical distribution of these samples approximates the posterior distribution, and statistics of this empirical distribution approximate the statistics of the true posterior.

For a given posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$, MCMC instantiates samples

$$\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(T)} \tag{2.72}$$

using a local sampling step

$$\boldsymbol{\theta}^{(t+1)} \sim \mathcal{T}(\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\theta}^{(t)}) \tag{2.73}$$

where $\mathcal{T}(\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\theta}^{(t)})$ is an algorithm-specific transition kernel defined to perform local updates that leave the posterior distribution invariant along the margins

$$p(\boldsymbol{\theta}^{(t)}) \to p(\boldsymbol{\theta}|\mathcal{D}) \text{ as } t \to \infty\,. \tag{2.74}$$

The typical goal when using MCMC is to choose a transition kernel that leads to as many effectively independent samples as possible given a fixed computational budget. Transition kernels often have hyper-parameters that require tuning, such as step sizes, proposal variances, and temperature schedules. These hyper-parameters can greatly effect the computational efficiency of an MCMC sampler.

There are many variants of MCMC: Metropolis-Hastings [Hastings, 1970], slice-sampling [Neal, 2003], Hamiltonian Monte Carlo [Neal, 2011], the no u-turn sampler [Hoffman and Gelman, 2014], parallel tempering [Swendsen and Wang, 1986], Gibbs sampling Geman and Geman [1987], among others. Some algorithms target specific difficulties (e.g. multi-modal posterior distributions), whereas some are considered general workhorse algorithms that sit behind robust probabilistic programming environments [Bob Carpenter and Riddell, 2015]. In many cases one algorithm can be combined with another (e.g. slice-sampling within parallel-tempering) to efficiently sample from a tricky posterior distribution.

One issue with MCMC algorithms is that they are difficult to scale. To generate an asymptotically correct Markov chain, each likelihood evaluation must incorporate information from the entire dataset.[8] When *optimizing* an objective, on the other hand, we can use noisy gradient estimators computed from a subsample of our data—which can be computationally advantageous when we have a massive dataset. This next section describes a set of methods that frame approximate Bayesian inference as an optimization problem, which can be a more scalable alternative to MCMC.

2.7 Variational Inference

Variational inference (VI) is an alternative family of approximate Bayesian inference methods. VI methods view the posterior approximation problem as an optimization problem over a space of approximating distributions.

Given a model, $p(\boldsymbol{\theta}, \mathcal{D}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$, of data $\mathcal{D}$ and parameters/latent variables $\boldsymbol{\theta}$, the goal of VI is to approximate the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. VI approximates this intractable posterior distribution with one from a simpler family, $\mathcal{Q} = \{q(\boldsymbol{\theta}; \boldsymbol{\lambda}), \boldsymbol{\lambda} \in \boldsymbol{\Lambda}\}$, parameterized by *variational parameters* $\boldsymbol{\lambda}$. VI procedures search for the member of that family, $q(\cdot; \boldsymbol{\lambda}) \in \mathcal{Q}$, that minimizes some divergence between the approximation q and the true posterior $p(\boldsymbol{\theta}|\mathcal{D})$.

Variational inference can be framed as an optimization problem, usually in terms of Kullback-Leibler (KL) divergence, of the following form

$$\begin{aligned}
\boldsymbol{\lambda}^* &= \operatorname*{arg\,min}_{\boldsymbol{\lambda} \in \Lambda} \mathrm{KL}(q(\boldsymbol{\theta}; \boldsymbol{\lambda}) \;||\; p(\boldsymbol{\theta}|\mathcal{D})) \\
&= \operatorname*{arg\,min}_{\boldsymbol{\lambda} \in \Lambda} \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} \left[\ln q(\boldsymbol{\theta}; \boldsymbol{\lambda}) - \ln p(\boldsymbol{\theta}|\mathcal{D})\right] \;.
\end{aligned}$$

The task is to find a setting of $\boldsymbol{\lambda}$ that makes $q(\boldsymbol{\theta}; \boldsymbol{\lambda})$ close to the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ in

[8] Though, for some structured models there exist methods to only compute a subsample, e.g. in Maclaurin and Adams [2014].

KL divergence.[9] Directly computing the KL divergence requires evaluating the posterior itself; therefore, VI procedures use the *evidence lower bound* (ELBO) as the optimization objective

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} \left[\ln p(\boldsymbol{\theta}, \mathcal{D}) - \ln q(\boldsymbol{\theta}; \boldsymbol{\lambda}) \right], \tag{2.75}$$

which, when maximized, minimizes the KL divergence between $q(\boldsymbol{\theta}; \boldsymbol{\lambda})$ and $p(\boldsymbol{\theta}|\mathcal{D})$. In special cases, parts of the ELBO can be expressed analytically (e.g. the entropy form or KL-to-prior form [Hoffman and Johnson, 2016]) — we focus on the general form in Equation 5.1.

Monte Carlo VI and Gradient Estimators

The ELBO objective in Equation 5.1 (and its gradient) is an expectation against the variational approximation $q_{\boldsymbol{\lambda}}$. In many interesting cases, this expectation cannot be computed analytically, making optimization more difficult. One way to circumvent this issue is to use an unbiased estimator of the ELBO and its gradient to do optimization [Paisley et al., 2012, Ranganath et al., 2014]. Because we've chosen the family q, we can generate inexpensive estimates of the ELBO and its gradient with respect to $\boldsymbol{\lambda}$.

The target of these estimators is the gradient of the ELBO

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} \left[f(\boldsymbol{\theta}, \boldsymbol{\lambda}) \right] . \tag{2.76}$$

where we have defined $f(\boldsymbol{\theta}, \boldsymbol{\lambda}) \triangleq \ln p(\boldsymbol{\theta}, \mathcal{D}) - \ln q(\boldsymbol{\theta}; \boldsymbol{\lambda})$ for notational simplicity.

One such estimator of the gradient is the *score function estimator*, based around the

[9] We use $q(\boldsymbol{\theta}; \boldsymbol{\lambda})$ and $q_{\boldsymbol{\lambda}}(\boldsymbol{\theta})$ interchangeably.

identity

$$\nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} \left[f(\boldsymbol{\theta}, \boldsymbol{\lambda}) \right] = \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} \left[f(\boldsymbol{\theta}, \boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}} \ln q_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) \right] \tag{2.77}$$

$$\approx \frac{1}{L} \sum_{\ell} f(\boldsymbol{\theta}^{(\ell)}, \boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}} \ln q_{\boldsymbol{\lambda}}(\boldsymbol{\theta}^{(\ell)}) \tag{2.78}$$

$$\triangleq \hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}^{(sf)} \tag{2.79}$$

The score function estimator is quite general, though often exhibits high variance.

Alternatively, the *reparameterization gradient estimator* estimates the expectation by differentiating through the random sampling procedure. To compute the this estimator, first we define a sampling procedure that separates a random seed from $\boldsymbol{\lambda}$. For a normal distribution, $q_{\boldsymbol{\lambda}}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{m}_{\boldsymbol{\lambda}}, \boldsymbol{s}_{\boldsymbol{\lambda}}^2)$, one such transformation is

$$\epsilon \sim \mathcal{N}(0, I) \tag{2.80}$$

$$\boldsymbol{\theta}(\epsilon, \boldsymbol{\lambda}) = \boldsymbol{m}_{\boldsymbol{\lambda}} + \boldsymbol{s}_{\boldsymbol{\lambda}} \cdot \epsilon \tag{2.81}$$

where we overload $\boldsymbol{\theta}(\cdot, \cdot)$ to be a deterministic transformation that takes spherical noise ϵ and parameters $\boldsymbol{\lambda}$ and produces a random variate with the distribution $\mathcal{N}(\boldsymbol{m}_{\boldsymbol{\lambda}}, \boldsymbol{s}_{\boldsymbol{\lambda}}^2)$.

Given this transformation, the reparameterization gradient estimator can then be computed

$$\nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} \left[f(\boldsymbol{\theta}, \boldsymbol{\lambda}) \right] = \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\epsilon \sim q_0} \left[f(\boldsymbol{\theta}(\epsilon, \boldsymbol{\lambda}), \boldsymbol{\lambda}) \right] \tag{2.82}$$

$$= \mathbb{E}_{\epsilon \sim q_0} \left[\nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\theta}(\epsilon, \boldsymbol{\lambda}), \boldsymbol{\lambda}) \right] \tag{2.83}$$

$$\approx \frac{1}{L} \sum_{\ell} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\theta}(\epsilon^{(\ell)}, \boldsymbol{\lambda}), \boldsymbol{\lambda}) \tag{2.84}$$

$$\triangleq \hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}^{(rg)} \tag{2.85}$$

The reparameterization gradient estimator is limited to continuous-valued random variables $\boldsymbol{\theta}$, as it requires differentiating through the sampling procedure. Though less general, it is often more efficient than the score function estimator. In Chapter 5, we improve upon the reparameterization gradient estimator by constructing a lower variance estimator that exploits model structure.

2.8 Model Evidence Estimation

Given alternative models for data, $p_a(\mathcal{D}|\boldsymbol{\theta}^{(a)})$ and $p_b(\mathcal{D}|\boldsymbol{\theta}^{(b)})$, the question often arises, "which model is better?" Within the Bayesian paradigm, one can directly compare the likelihood of each model. For instance, denoting model a as $\mathcal{M}_a$, we can compute the *model evidence*, or the marginal likelihood of the observed data, integrating out all model parameters

$$p(\mathcal{D}|\mathcal{M}_a) = \int p_a(\mathcal{D}|\boldsymbol{\theta}^{(a)}) p(\boldsymbol{\theta}^{(a)}) d\boldsymbol{\theta}^{(a)} \,. \tag{2.86}$$

The marginal likelihood for each model can be compared—both are simply probability densities of the data. Even when $\boldsymbol{\theta}^{(a)}$ and $\boldsymbol{\theta}^{(b)}$ have different dimension or constraints, the coherent use of probability over these parameters makes the marginal likelihood a fair comparison between models. These marginal likelihoods can then be combined with priors over which model is more likely to be true a priori, e.g. $Pr(\mathcal{M}_a)$ vs $Pr(\mathcal{M}_b)$ to compute a posterior over models.

Computing the integral in Equation 2.86 can be intractable in general. Methods for efficiently estimating model evidence is an active area of research. Annealed importance sampling [Neal, 2001], bridge and path sampling [Gelman and Meng, 1998], and nested sampling [Skilling et al., 2006] are examples of such methods that rely on generating a Monte Carlo estimator of the model evidence.

Even if we're not using Bayesian model evidence to decide between models, model evidence estimation can be important. A common practice in machine learning is to compare the *out-of-sample* data likelihood between two models to see which generalizes better. For certain complex latent variables models, just computing the out-of-sample likelihood for each observation is itself a (conditional) model evidence problem that requires marginal likelihood estimation. This appears when comparing the test log-likelihood of variational autoencoders, which can be a fraught exercise when ELBO estimates are not a good approximation of marginal likelihood estimates (or even their ordering) [Wu et al., 2016].

Discussion Given the building blocks of probabilistic modeling, this next chapter will describe a few applied probabilistic models in a range of areas.

3

Probabilistic Models for Scientific Discovery

Big, heterogeneous data require expressive models to explain their many sources of variation. This chapter describes probabilistic modeling case studies in three applied domains: astronomy, healthcare, and sports analytics. While the goals of each model are application specific, the challenges of specifying a useful probabilistic model and inferring its latent structure are shared.

In each of these applied projects, we model a high-dimensional spatial, temporal, or spatiotemporal signal with unobserved structure—spectroscopic and photometric measurements of quasars, electrocardiogram recordings of the cardiac cycle, and spatiotemporal trajectories of basketball player movement and decision-making throughout live game-play. Our goal is to describe this unobserved structure with an accurate and interpretable probabilistic model—this will allow us to make predictions about new observations, visualize the structure discovered in our data, and explore the data to obtain new insights about the underlying phenomenon.

We also want to incorporate existing knowledge into our probabilistic model. For example, we understand the physical relationship between redshift and rest-frame spectral energy distributions. This information is invariant to statistical sampling, and its incorporation into our model will make its inferences more *portable*—structure that is not tied to a particular statistical sample, but able to make reliable predictions in different contexts. We will see a concrete example of this portability in the astronomy modeling project.

In each case, the probabilistic model itself is not the end goal—it will be used as a tool to aid decision-making. We want our model to be able to report what it can not know from the data its observed. It is necessary for our model to quantify these sources of uncertainty.

We must also navigate the trade-off between flexibility and interpretability. The underlying phenomena of each are understood by domain experts at a particular level of abstraction, and building models that reveal new knowledge about the underlying phenomenon requires that we operate (at least partially) at this level of abstraction. In each model, we use a flexible (and difficult-to-interpret) model component to capture important difficult-to-prescribe structure in the data—a latent Gaussian process factor component to capture structure in spectral energy distributions; a deep generative model to describe the variation in cardiac morphology; a mixture of Bezier curves to describe the discrete actions of basketball players.

The following applied probabilistic modeling projects are described in the next three sections within this chapter.

Photometric redshift In Section 3.1, we describe a novel probabilistic model of two different types of astronomical data that measure the same underlying phenomenon, and apply the idea to a particular type of source, quasi-stellar radio objects or quasars. We use a Gaussian process latent factor model to capture complex variation in the underlying latent variables, and a physically motivated data generating process that allows us to accurately measure a source's *redshift*, a particular physical quantity of interest, from a low-information data source. This section is based on previously published work in Miller et al. [2015].

Electrocardiogram modeling In Section 3.2 we develop a new probabilistic generative model of electrocardiogram (EKG) tracings. This model describes multiple sources of variation in EKGs, including patient-specific cardiac cycle morphology and between-cycle variation that leads to quasi-periodicity. We use a deep generative network as a flexible model component to describe factors of variation in beat-specific morphology. We apply our model to a set of 549 EKG records, including over 4,600 unique beats, and show that it is able to discover interpretable dimensions of variation, such as patient similarity and meaningful physiological features (e.g., T wave inversion). This section is based on previously published work in Miller et al. [2017d].

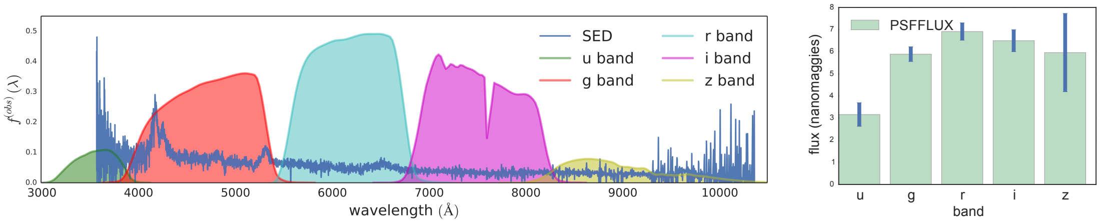
Unsupervised structure discovery in basketball possessions In Section 3.3, we present a new model for representing variation in basketball possessions, with the goal of organizing and exploring a massive database of basketball player-tracks. This model describes player-specific movements and team interaction that compose the offensive structure in a basketball possession. We show that our model is able to group together possessions with similar offensive structure, allowing efficient search and exploration of the entire database of player-tracking data. We show that our model finds repeated offensive structure in teams (e.g. strategies), providing a much more sophisticated, yet interpretable lens into basketball player-tracking data. This section is based on work previously published in Miller and Bornn [2017].

3.1 Application: Photometric Redshift

Enormous amounts of astronomical data are collected by a range of instruments at multiple spectral resolutions, providing information about billions of sources of light in the observable universe [Alam et al., 2015, Martin et al., 2005]. Among these data are measurements of the spectral energy distributions (SEDs) of sources of light (e.g. stars, galaxies, and quasars). The SED describes the distribution of energy radiated by a source over the spectrum of wavelengths or photon energy levels. The SED is of interest because it conveys information about a source's physical properties, including type, chemical composition, and redshift, which will be an estimand of interest in this model.

The SED can be thought of as a latent function of which we can only obtain noisy measurements. Measurements of SEDs, however, are produced by instruments at widely varying spectral resolutions—some instruments measure many wavelengths simultaneously (e.g. spectroscopy), while others average over large swaths of the energy spectrum and report a low dimensional summary (e.g. photometry). Spectroscopic data describe a source's SED in finer detail than broadband photometric data. For example, the Baryonic Oscillation Spectroscopic Survey [Dawson et al., 2013] measures SED samples at over four thousand wavelengths between 3,500 and 10,500 Å. In contrast, the Sloan Digital Sky Survey (SDSS) [Alam et al., 2015] collects spectral information in only 5 broad spectral bins by using broadband filters (called u, g, r, i, and z), but at a much higher *spatial* resolution. Photometric preprocessing models can then aggregate pixel information into five band-specific fluxes and their uncertainties [Stoughton et al., 2002], reflecting the weighted average response over a large range of the wavelength spectrum. The two methods of spectral information collection are graphically compared in Figure 3.1.

Despite carrying less spectral information, broadband photometry is more widely



(a) **(b)**

Figure 3.1: Left: example of a BOSS-measured quasar SED with SDSS band filters, $S_b(\lambda)$, $b \in \{u, g, r, i, z\}$, overlaid. Right: the same quasar's photometrically measured band fluxes. Spectroscopic measurements include noisy samples at thousands of wavelengths, whereas SDSS photometric fluxes reflect the (weighted) average response over a large range of wavelengths. Compared to spectroscopy, photometric data convey much less information about the underlying spectral energy distribution.

available and exists for a larger number of sources than spectroscopic measurements. This work develops a method for inferring physical properties sources by jointly modeling spectroscopic and photometric data. One use of our model is to measure the redshift of quasars for which we only have photometric observations. Redshift is a phenomenon in which the observed SED of a source of light is stretched toward longer (redder) wavelengths. This effect is due to a combination of radial velocity with respect to the observer and the expansion of the universe (termed *cosmological redshift*) [Hogg, 1999, Harrison, 1993]. Quasars, or quasi-stellar radio sources, are extremely distant and energetic sources of electromagnetic radiation that can exhibit high redshift [Silk and Rees, 1998]. Accurate estimates and uncertainties of redshift measurements from photometry have the potential to guide the use of higher spectral resolution instruments to study sources of interest. Furthermore, accurate photometric models can aid the automation of identifying source types and estimating physical characteristics of faintly observed sources in large photometric surveys [Regier et al., 2015].

We directly model a quasar's latent SED and the process by which it generates spectroscopic and photometric observations to jointly describe both resolutions of data. Representing a quasar's SED as a latent random measure, we describe a Bayesian inference procedure to compute the marginal probability distribution of a quasar's redshift given observed photometric fluxes and their uncertainties. The following section provides relevant application and statistical background. Section 3.1.2 describes our probabilistic model of SEDs and broadband photometric measurements. Section 3.1.3 outlines our MCMC-based inference method for efficiently computing statistics of the posterior distribution. Section 5.4 presents redshift and SED predictions from photometric measurements, among other model summaries, and a quantitative comparison between our method and two existing "photo-z". We conclude with a discussion of directions for future work.

3.1.1 Background

The SEDs of most stars are roughly approximated by Planck's law for black body radiators and stellar atmosphere models [Gray et al., 2001]. Quasars, on the other hand, have complicated SEDs characterized by some salient features, such as the Lyman-α forest that indicates the absorption of light at many wavelengths from neutral hydrogen gas between the earth and the quasar [Weinberg et al., 2003]. One of the most interesting properties of quasars (and galaxies) conveyed by the SED is redshift, which gives us insight into an object's distance and age. Redshift affects our observation of SEDs by "stretching" the wavelengths, $\lambda \in \Lambda$, of the quasar's *rest frame* SED, skewing toward longer (redder) wavelengths. Denoting the *rest frame* SED of a quasar n as a function, $f_n^{(\text{rest})} : \Lambda \to \mathbb{R}_+$, the effect of redshift with value z_n (typically between 0 and 7) on the *observation-frame* SED is described by the relationship

$$f_n^{(\text{obs})}(\lambda) = f_n^{(\text{rest})}\left(\frac{\lambda}{1+z_n}\right) . \tag{3.1}$$

Some observed quasar spectra and their "de-redshifted" rest frame spectra are depicted in Figure 3.2. The BOSS spectra are stored in units $10^{-17} \cdot \text{erg} \cdot \text{cm}^{-2} \cdot \text{s}^{-1} \cdot \text{Å}^{-1}$.

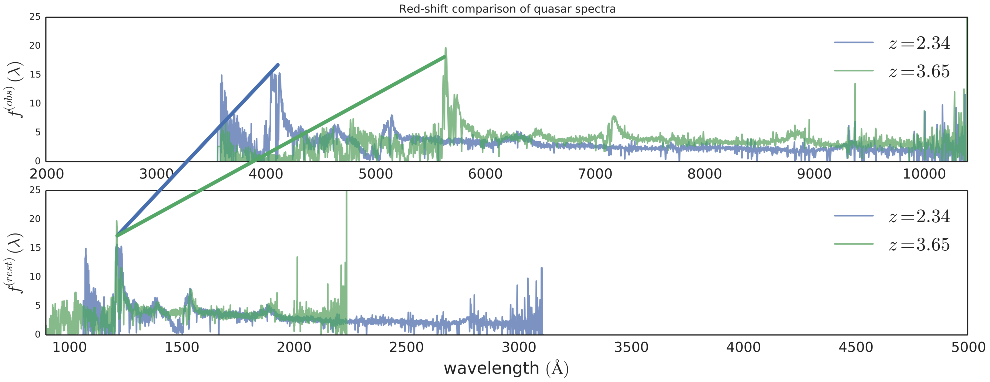


Figure 3.2: Spectroscopic measurements of multiple quasars at different redshifts, z. The upper graph depicts the sample spectrograph in the observation frame, intuitively thought of as "stretched" by a factor $(1 + z)$. The lower figure depicts the "de-redshifted" (rest frame) version of the same quasar spectra, The two lines show the corresponding locations of the characteristic peak in each reference frame. Note that the x-axis has been changed to ease the visualization - the transformation is much more dramatic. The appearance of translation is due to missing data; we don't observe SED samples outside the range 3,500-10,500 Å.

3.1.2 Model

This section describes our probabilistic model of spectroscopic and photometric observations.

Spectroscopic flux model The SED of a quasar is a non-negative function $f : \Lambda \rightarrow \mathbb{R}_{+}$, where Λ denotes the range of wavelengths and $\mathbb{R}_{+}$ are non-negative real numbers representing flux density. Our model specifies a quasar's *rest frame* SED as a latent random function. Quasar SEDs are highly structured, and we model this structure by imposing the assumption that each SED is a convex mixture of K latent, positive basis functions. The model assumes there are a small number (K) of latent features or characteristics and that each quasar can be described by a short vector of mixing weights over these features.

We place a normalized log-Gaussian process prior on each of these basis functions (described in supplementary material). The generative procedure for quasar spectra begins with a shared basis

$$\beta_k(\cdot) \overset{\text{iid}}{\sim} \mathcal{GP}(0, K_\theta),\ k = 1, \ldots, K, \qquad B_k(\cdot) = \frac{\exp(\beta_k(\cdot))}{\int_\Lambda \exp(\beta_k(\lambda))\, d\lambda}\,, \tag{3.2}$$

where K_θ is the kernel and B_k is the exponentiated and normalized version of β_k. For each quasar n,

$$\mathbf{w}_n \sim p(\mathbf{w})\,,\ \text{s.t.}\ \sum_{w_k} w_k = 1, \qquad m_n \sim p(m)\,,\ \text{s.t.}\ m_n > 0, \qquad z_n \sim p(z), \tag{3.3}$$

where $\mathbf{w}_n$ mixes over the latent types, m_n is the apparent brightness, z_n is the quasar's redshift, and distributions $p(\mathbf{w})$, $p(m)$, and $p(z)$ are priors to be specified later. As each positive SED basis function, B_k, is normalized to integrate to one, and each quasar's weight vector $\mathbf{w}_n$ also sums to one, the latent *normalized* SED is then constructed as

$$f_n^{(\text{rest})}(\cdot) = \sum_k w_{n,k} B_k(\cdot) \tag{3.4}$$

and we define the unnormalized SED $\tilde{f}_n^{(\text{rest})}(\cdot) \equiv m_n \cdot f_n^{(\text{rest})}(\cdot)$. This parameterization admits the interpretation of $f_n^{(\text{rest})}(\cdot)$ as a probability density scaled by m_n. This interpretation allows us to separate out the apparent brightness, which is a function of distance and overall luminosity, from the SED itself, which carries information pertinent to the estimand of interest, redshift.

For each quasar with spectroscopic data, we observe noisy samples of the redshifted and scaled spectral energy distribution at a grid of P wavelengths $\lambda \in \{\lambda_1, \ldots, \lambda_P\}$. For quasar n, our *observation frame* samples are conditionally distributed as

$$x_{n,\lambda} | z_n, \mathbf{w}_n, \{B_k\} \overset{\text{ind}}{\sim} \mathcal{N}\left(\tilde{f}_n^{(\text{rest})}\left(\frac{\lambda}{1+z_n}\right), \sigma^2_{n,\lambda}\right) \tag{3.5}$$

where $\sigma^2_{n,\lambda}$ is known measurement variance from the instruments used to make the observations.

Due to the complicated shape of quasar SEDs, we use a Gaussian process (GP) prior to flexibly encode our prior beliefs about their structure and shape. Refer to Section 2.2.2 for a brief review of Gaussian processes.

Photometric flux model Photometric data summarize the amount of energy observed over a large swath of the wavelength spectrum. Roughly, a photometric flux measures (proportionally) the number of photons recorded by the instrument over the duration of an exposure, filtered by a band-specific sensitivity curve. We express flux in nanomaggies [SDSSIII, 2013]. Photometric fluxes and measurement error derived from broadband imagery have been computed directly from pixels [Stoughton et al., 2002]. For each quasar n, SDSS photometric data are measured in five bands, $b \in \{u, g, r, i, z\}$, yielding a vector of five flux values and their variances, $\mathbf{y}_n$ and $\tau^2_{n,b}$. Each band, b, measures photon observations at each wavelength in proportion to a known filter sensitivity, $S_b(\lambda)$. The filter sensitivities for the SDSS *ugriz* bands are depicted in Figure 3.1, with an example observation frame quasar SED overlaid. The actual measured fluxes can be computed by integrating the full object's spectrum, $m_n \cdot f_n^{(\mathrm{obs})}(\lambda)$ against the filters. For a band $b \in \{u, g, r, i, z\}$

$$\mu_b(f_n^{(\mathrm{rest})}, z_n) = \int f_n^{(\mathrm{obs})}(\lambda)\, S_b(\lambda)\, C(\lambda)\, d\lambda\,, \tag{3.6}$$

where $C(\lambda)$ is a conversion factor to go from the units of $f_n(\lambda)$ to nanomaggies (details of this conversion are available in the supplementary material). The function μ_b takes in a rest frame SED, a redshift (z) and maps it to the observed b-band specific flux. The results of this projection onto SDSS bands are modeled as independent Gaussian random variables with known variance

$$y_{n,b} \,|\, f_n^{(\mathrm{rest})}, z_n \overset{\mathrm{ind}}{\sim} \mathcal{N}(\mu_b(f_n^{(\mathrm{rest})}, z_n), \tau^2_{n,b})\,. \tag{3.7}$$

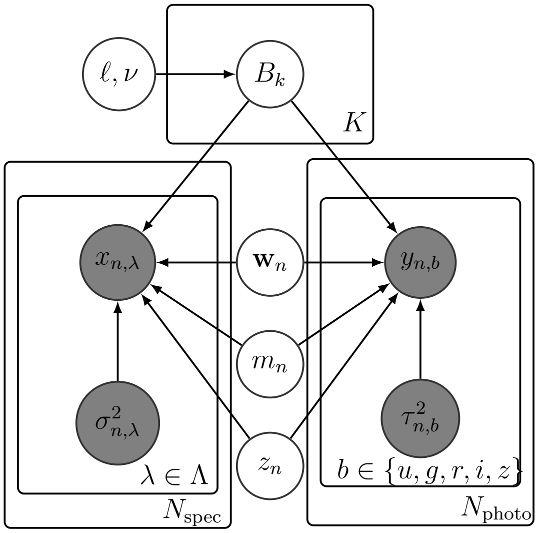


Figure 3.3: Graphical model representation of the joint photometry and spectroscopy model. The left shaded variables represent spectroscopically measured samples and their variances. The right shaded variables represent photometrically measured fluxes and their variances. The upper box represents the latent basis, with GP prior parameters ℓ and ν. Note that $N_{\text{spec}} + N_{\text{photo}}$ replicates of $\mathbf{w}_n, m_n$ and z_n are instantiated.

Conditioned on the basis, $B = \{B_k\}$, we can represent $f_n^{(\text{rest})}$ with a low-dimensional vector. Note that $f_n^{(\text{rest})}$ is a function of $\mathbf{w}_n, z_n, m_n$, and B (see Equation 3.4), so we can think of μ_b as a function of $\mathbf{w}_n, z_n, m_n$, and B. We overload notation, and re-write the conditional likelihood of photometric observations as

$$y_{n,b} \,|\, \mathbf{w}_n, z_n, m_n, B \sim \mathcal{N}(\mu_b(\mathbf{w}_n, z_n, m_n, B), \tau^2_{n,b})\,. \tag{3.8}$$

Intuitively, what gives us statistical traction in inferring the posterior distribution over z_n is the structure learned in the latent basis, B, and weights w, i.e., the features that correspond to distinguishing bumps and dips in the SED.

Note on priors For photometric weight and redshift inference, we use a flat prior on $z_n \in [0, 8]$, and empirically derived priors for m_n and w_n, from the sample of spectroscopically measured sources. Choice of priors is described in the supplementary material.

3.1.3 Inference

Basis estimation For computational tractability, we first compute a maximum a posteriori (MAP) estimate of the basis, B_{map} to condition on. Using the spectroscopic data, $\{x_{n,\lambda}, \sigma^2_{n,\lambda}, z_n\}$, we compute a discretized MAP estimate of $\{B_k\}$ by directly optimizing the unnormalized (log) posterior implied by the likelihood in Equation 3.5, the GP prior over B, and diffuse priors over $\mathbf{w}_n$ and m_n,

$$\begin{aligned} &p\left(\{\mathbf{w}_n, m_n\}, \{B_k\} | \{x_{n,\lambda}, \sigma^2_{n,\lambda}, z_n\}\right) && (3.9) \\ &\propto \prod_{n=1}^{N} p(x_{n,\lambda}|z_n, \mathbf{w}_n, m_n, \{B_k\}) p(\{B_k\}) p(\mathbf{w}_n) p(m_n)\,. && (3.10) \end{aligned}$$

We use gradient descent with momentum and LBFGS [Nocedal, 1980] directly on the parameters $\beta_k, \omega_{n,k}$, and $\log(m_n)$ for the N_{spec} spectroscopically measured quasars. Gradients were automatically computed using `autograd` [Maclaurin et al., 2015a]. Following [Walcher et al., 2011], we first resample the observed spectra into a common rest frame grid, $\lambda_0 = (\lambda_{0,1}, \ldots, \lambda_{0,V})$, easing computation of the likelihood. We note that although our model places a full distribution over B_k, efficiently integrating out those parameters is left for future work.

Sampling $\mathbf{w}_n, m_n$, and z_n The Bayesian “photo-z” task requires that we compute posterior marginal distributions of z, integrating out $\mathbf{w}$, and m. To compute these distributions, we construct a Markov chain over the state space including z, $\mathbf{w}$, and m that leaves the target posterior distribution invariant. We treat the inference problem for each photometrically measured quasar, $\mathbf{y}_n$, independently. Conditioned on a basis $B_k, k = 1, \ldots, K$, our goal is to draw posterior samples of $\mathbf{w}_n$, m_n and z_n for each

n. The unnormalized posterior can be expressed

$$p(\mathbf{w}_n, m_n, z_n | \mathbf{y}_n, B) \propto p(\mathbf{y}_n | \mathbf{w}_n, m_n, z_n, B) p(\mathbf{w}_n, m_n, z_n) \tag{3.11}$$

where the left likelihood term is defined in Equation 3.8. Note that due to analytic intractability, we numerically integrate expressions involving $\int_\Lambda f_n^{(obs)}(\lambda) d\lambda$ and $S_b(\lambda)$. Because the observation $\mathbf{y}_n$ can often be well explained by various redshifts and weight settings, the resulting marginal posterior, $p(z_n | \mathbf{X}, \mathbf{y}_n, B)$, is often multi-modal, with regions of near zero probability between modes. Intuitively, this is due to the information loss in the SED-to-photometric flux integration step.

This multi-modal property is problematic for many standard MCMC techniques. Single chain MCMC methods have to jump between modes or travel through a region of near-zero probability, resulting in slow mixing. To combat this effect, we use parallel tempering [Brooks et al., 2011], a method that is well-suited to constructing Markov chains on multi-modal distributions. Parallel tempering instantiates C independent chains, each sampling from the target distribution raised to an inverse temperature. Given a target distribution, $\pi(x)$, the constructed chains sample $\pi_c(x) \propto \pi(x)^{1/T_c}$, where T_c controls how "hot" (i.e., how close to uniform) each chain is. At each iteration, swaps between chains are proposed and accepted with a standard Metropolis-Hastings acceptance probability

$$\Pr(\text{accept swap } c, c') = \frac{\pi_c(x_{c'}) \pi_{c'}(x_c)}{\pi_c(x_c) \pi_{c'}(x_{c'})} \,. \tag{3.12}$$

Within each chain, we use component-wise slice sampling [Neal, 2003] to generate samples that leave each chain's distribution invariant. Slice sampling is an auxiliary variable MCMC algorithm that instantiates a Markov chain based on draws from the uniform distribution "under the probability density curve." Slice-sampling is a (relatively) tuning-free MCMC method, a convenient property when sampling from thousands of independent posteriors. We found parallel tempering to be essential to accurate posterior simulations and slice-sampling an easy-to-use MCMC transition.

3.1.4 Experiments and Results

We conduct three experiments to test our model, where each experiment measures redshift predictive accuracy for a different train/test split of spectroscopically measured quasars from the DR10QSO dataset [Pâris et al., 2014] with confirmed redshifts in the range $z \in (.01, 5.85)$. Our experiments split train/test in the following ways: (i) randomly, (ii) by r-band fluxes, (iii) by redshift values. In split (ii), we train on the brightest 90% of quasars, and test on a subset of the remaining. Split (iii) takes the lowest 85% of quasars as training data, and a subset of the brightest 15% as test cases. Splits (ii) and (iii) are intended to test the method's robustness to different training and testing distributions, mimicking the discovery of fainter and farther sources. For each split, we find a MAP estimate of the basis, $B_1, \ldots, B_K$, and weights, $\mathbf{w}_n$ to use as a prior for photometric inference. For computational purposes, we limit our training sample to a random subsample of 2,000 quasars. The following sections outline the resulting model fit and inferred SEDs and redshifts.

Basis validation We examined multiple choices of K using out of sample likelihood on a validation set. In the following experiments we set $K = 4$, which balances generalizability and computational tradeoffs. Discussion of this validation is provided in the supplementary material. Following [Budavari et al., 2001] we set $K = 4$, and note that this is also the number of PCA components that have been shown to carry over 90% of the variation of quasar spectroscopy [Suzuki, 2006]. This value could also be fit using model-checking methods for latent factorization models, which we do not address for computational reasons.

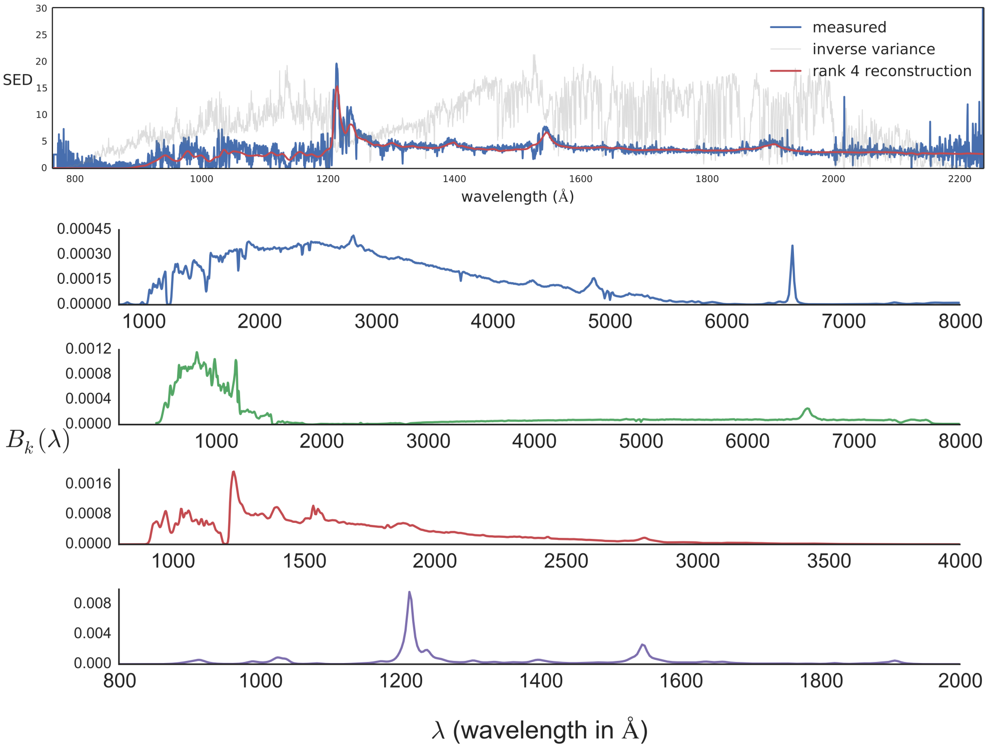


Figure 3.4: Top: MAP estimate of the latent bases $B = \{B_k\}_{k=1}^K$. Note the different ranges of the x-axis (wavelength). Each basis function distributes its mass across different regions of the spectrum to explain different salient features of quasar spectra in the rest frame. Bottom: model reconstruction of a training-sample SED.

SED Basis We depict a MAP estimate of $B_1, \ldots, B_K$ in Figure 3.4. Our basis decomposition enjoys the benefit of physical interpretability due to our density-estimate formulation of the problem. Basis B_4 places mass on the Lyman-α peak around 1,216 Å, allowing the model to capture the co-occurrence of more peaked SEDs with a bump around 1,550 Å. Basis B_1 captures the H-α emission line at around 6,500 Å. Because of the flexible nonparametric priors on B_k our model is able to automatically learn these features from data. The positivity of the basis and weights distinguishes our model from PCA-based methods, which sacrifice physical interpretability.

Photometric measurements For each test quasar, we construct an 8-chain parallel tempering sampler and run for 8,000 iterations, and discard the first 4,000 samples as burn-in. Given posterior samples of z_n, we take the posterior mean as a point estimate. Figure 3.5 compares the posterior mean to spectroscopic measurements (for three different data-split experiments), where the gray lines denote posterior sample quantiles. In general there is a strong correspondence between spectroscopically measured redshift and our posterior estimate. In cases where the posterior mean is off, our distribution often covers the spectroscopically confirmed value with probability mass. This is clear upon inspection of posterior marginal distributions that exhibit extreme multi-modal behavior. To combat this multi-modality, it is necessary to inject the model with more information to eliminate plausible hypotheses; this information could come from another measurement (e.g., a new photometric band), or from structured prior knowledge over the relationship between $z_n, \mathbf{w}_n$, and m_n. Our method simply fits a mixture of Gaussians to the spectroscopically measured $\mathbf{w}_n, m_n$ sample to formulate a prior distribution. However, incorporating the statistical relationship between z_n, $\mathbf{w}_n$ and m_n, similar to the XDQSOz technique, will be incorporated in future work.

Comparisons We compare the performance of our redshift estimator with two recent photometric redshift estimators, XDQSOz [Bovy et al., 2012] and a neural network [Brescia et al., 2013]. The method in [Bovy et al., 2012] is a conditional density estimator that discretizes the range of one flux band (the i-band) and fits a mixture of Gaussians to the joint distribution over the remaining fluxes and redshifts. One disadvantage to this approach is there there is no physical significance to the mixture of Gaussians, and no model of the latent SED. Furthermore, the original method trains and tests the model on a pre-specified range of i-magnitudes, which is problematic when predicting redshifts on much brighter or dimmer stars. The regression approach from [Brescia et al., 2013] employs a neural network with two hidden layers, and the SDSS fluxes as inputs. More features (e.g., more photometric bands) can be incorporated into all models, but we

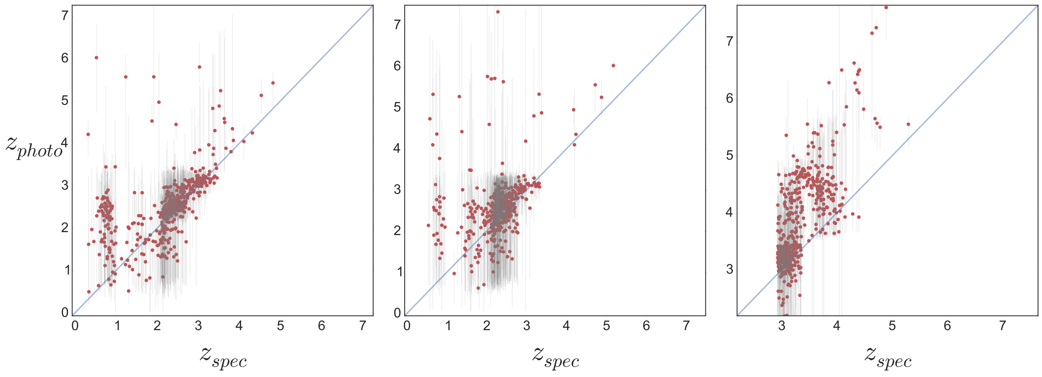


Figure 3.5: Comparison of spectroscopically (x-axis) and photometrically (y-axis) measured redshifts from the SED model for three different data splits. The left reflects a random selection of 4,000 quasars from the DR10QSO dataset. The right graph reflects a selection of 4,000 test quasars from the upper 15% ($z_{cutoff} \approx 2.7$), where all training was done on lower redshifts. The red estimates are posterior means.

limit our experiments to the five SDSS bands for the sake of comparison. Further detail on these two methods and a broader review of "photo-z" approaches are available in the supplementary material.

Average error and test distribution We compute mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE) to measure predictive performance. Table 3.1 compares prediction errors for the three different approaches (XD, NN, Spec). Our experiments show that accurate redshift measurements are attainable even when the distribution of training set is different from test set by directly modeling the SED itself. Our method dramatically outperforms [Bovy et al., 2012] and [Brescia et al., 2013] in split (iii), particularly for very high redshift fluxes. We also note that our training set is derived from only 2,000 examples, whereas the training set for XDQSOz and the neural network were ≈ 80,000 quasars and 50,000 quasars, respectively. This shortcoming can be overcome with more sophisticated inference techniques for the non-negative basis. Despite this, the SED-based predictions are comparable. Additionally, because we are directly modeling the latent SED, our

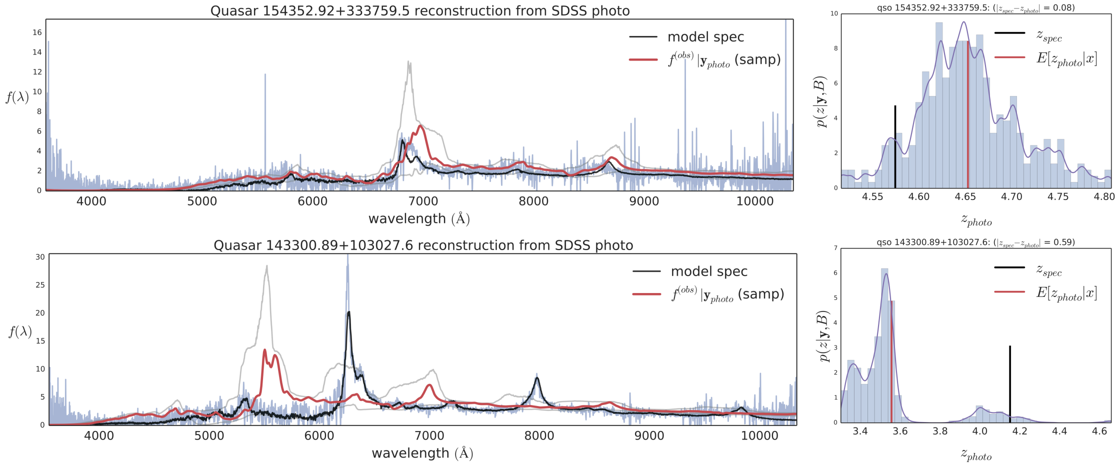


Figure 3.6: Left: inferred SEDs from photometric data. The black line is a smoothed approximation to the "true" SED using information from the full spectral data. The red line is a sample from the posterior, $f_n^{(obs)}(\lambda)|\mathbf{X}, \mathbf{y}_n, B$, which imputes the entire SED from only five flux measurements. Note that the bottom sample is from the left mode, which under-predicts redshift. Right: corresponding posterior predictive distributions, $p(z_n|\mathbf{X}, \mathbf{y}_n, B)$. The black line marks the spectroscopically confirmed redshift; the red line marks the posterior mean. Note the difference in scale of the x-axis.

method admits a posterior estimate of the entire SED. Figure 3.6 displays posterior SED samples and their corresponding redshift marginals for test-set quasars inferred from only SDSS photometric measurements.

3.1.5 Discussion

We have presented a generative model of two sources of information at very different spectral resolutions to form an estimate of the latent spectral energy distribution of quasars. We also described an efficient MCMC-based inference algorithm for computing posterior statistics given photometric observations. Our model accurately predicts and characterizes uncertainty about redshifts from only photometric observations and a small number of separate spectroscopic examples. Moreover, we showed that we can make reasonable estimates of the unobserved SED itself, from which we can make inferences about other physical properties informed by the full SED.

Table 3.1: Prediction error for three train-test splits, (i) random, (ii) flux-based, (iii) redshift-based, corresponding to XDQSOz [Bovy et al., 2012] (XD), the neural network approach [Brescia et al., 2013] (NN), our SED-based model (Spec). The middle and lowest sections correspond to test redshifts in the upper 50% and 10%, respectively. The XDQSOz and NN models were trained on (roughly) 80,000 and 50,000 example quasars, respectively, while the Spec models were trained on 2,000.

	MAE			MAPE			RMSE		
split	XD	NN	Spec	XD	NN	Spec	XD	NN	Spec
random (all)	**0.359**	0.773	0.485	**0.293**	0.533	0.430	**0.519**	0.974	0.808
flux (all)	**0.308**	0.483	0.497	**0.188**	0.283	0.339	**0.461**	0.660	0.886
redshift (all)	0.841	0.736	**0.619**	0.237	0.214	**0.183**	1.189	0.923	**0.831**
random ($z > 2.35$)	**0.247**	0.530	0.255	**0.091**	0.183	0.092	**0.347**	0.673	0.421
flux ($z > 2.33$)	**0.292**	0.399	0.326	**0.108**	0.143	0.124	**0.421**	0.550	0.531
redshift ($z > 3.20$)	1.327	1.149	**0.806**	0.357	0.317	**0.226**	1.623	1.306	**0.997**
random ($z > 3.11$)	**0.171**	0.418	0.289	**0.050**	0.117	0.082	**0.278**	0.540	0.529
flux ($z > 2.86$)	0.373	0.493	**0.334**	0.112	0.144	**0.103**	**0.606**	0.693	0.643
redshift ($z > 3.80$)	2.389	2.348	**0.829**	0.582	0.569	**0.198**	2.504	2.405	**1.108**

We see multiple avenues of future work. Firstly, we can extend the model of SEDs to incorporate more expert knowledge. One such augmentation would include a fixed collection of features, curated by an expert, corresponding to physical properties already known about a class of sources. Furthermore, we can also extend our model to directly incorporate photometric pixel observations, as opposed to preprocessed flux measurements. Secondly, we note that our method is more more computationally burdensome than XDQSOz and the neural network approach. Another avenue of future work is to find accurate approximations of these posterior distributions that are cheaper to compute. Lastly, we can extend our methodology to galaxies, whose SEDs can be quite complicated. Galaxy observations have spatial extent, complicating their SEDs. The combination of SED and spatial appearance modeling and computationally efficient inference procedures is a promising route toward the automatic characterization of millions of sources from the enormous amounts of data available in massive photometric surveys.

This next section describes a probabilistic modeling project in the area of healthcare. In this project, our goal is to describe the factors of variation of electrocardiogram data. This is expanded upon research that is based on previously published work [Miller et al., 2017d].

3.2 Application: Electrocardiogram Tracings

An electrocardiogram (EKG) is a common non-invasive medical test that measures the electrical activity of a patient's heart by recording the time-varying potential difference between electrodes placed on the surface of the skin. The resulting data is a multivariate time-series that reflects the depolarization and repolarization of the heart muscle that occurs during each heartbeat. These raw waveforms are then inspected by a physician to detect irregular patterns that are evidence of an underlying physiological problem.

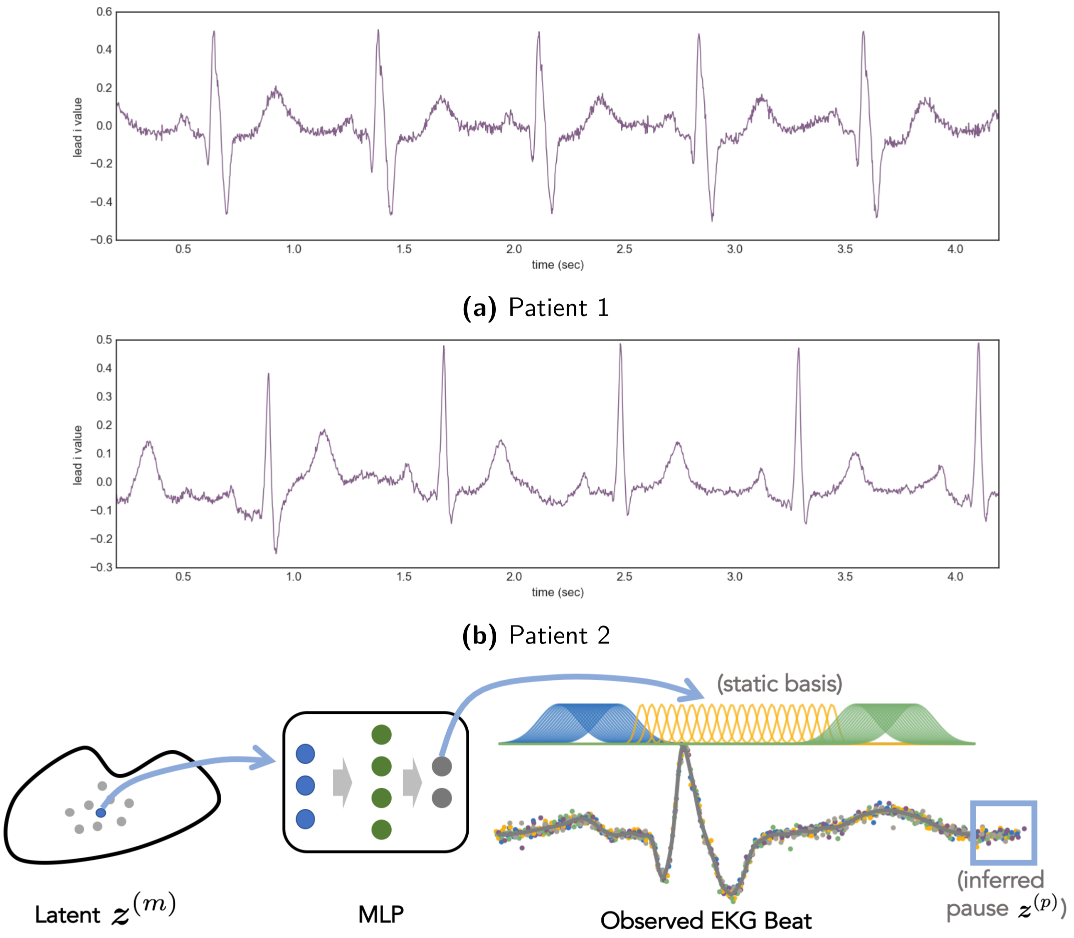
In this paper, we build a hierarchical generative model of electrocardiogram signals that disentangles sources of variation. We are motivated to build a generative model of EKGs for multiple reasons. First, we would like our inferences to properly cope with nuisance variation present in EKG signals (e.g. variation in cardiac cycle with breathing and inadvertent movement). Second, generative models can be used for semi-supervised tasks — not all EKG observations are paired with test results or diagnoses. Semi-supervised modeling allows us to leverage a large amount of unlabeled EKG data to improve predictions when training with a smaller labeled dataset (under an appropriate model of censoring or missingness). Third, in medical diagnoses, correlating model features with underlying physiological realities is important for several reasons, including model checking and interpretability. A generative model provides an intuitive mechanism to examine the inner workings of the model — one can always draw a sample from the inferred generative distribution to reveal what the latent features themselves represent in terms of observed data features. Finally, a statistical model can be much more sensitive and robust to difficult-to-detect patterns.

3.2.1 Modeling Electrocardiograms

Looking at an EKG, a few features of the tracing stand out. First, there are individual heartbeats—discrete periods of active contraction of the heart muscle, that are responsible for pumping blood to the lungs and the rest of the body. These are the bursts of activity in the tracing. Small parts of the signal in this area are carefully scrutinized by physicians for signs of disturbance in the heart's electrical conduction system, or heart attack. Second, there are the periods between heartbeats, when the muscle is resting as the heart fills with blood. Since there is little electrical activity during this period, the electrodes record it as a flat line. The time between beats can vary as a function of idiosyncratic aspects of the patient's conduction system, or with variations in blood flow to the heart driven by the respiratory cycle. Third, patient movement or artifacts in the recording equipment (conductance of the electrodes, etc) can introduce arbitrary changes in the signal.

Our generative model tackles these multiple sources of variation in electrocardiogram data directly. We model: (i) the morphology of an individual beat; (ii) the variation in that morphology from beat to beat; (iii) the variability in the periodicity of the beat; and (iv) nuisance variability in the measurement process (e.g. overall drift due to movement). The output of this model will be a set of features that explain these sources of variability, which can be useful in exploratory and predictive tasks.

Our data are multi-dimensional temporal observations $\boldsymbol{Y} = \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$ where $\boldsymbol{y}_t \in \mathbb{R}^D$ are sampled on a regular time grid, $\boldsymbol{t}^{(obs)} = t_1, \ldots, t_T$ (given in seconds). Our model separately parameterizes the morphology of the cardiac cycle, its duration, and the length of time between cycles. We express these sources of variation as a hierarchical



(a) Patient 1

(b) Patient 2

(c) Generative model

Figure 3.7: Top: Example EKG tracings (single lead) from two patients. Bottom: The generative procedure — each beat is represented in a low-dimensional latent space (left). To generate a beat, this vector is up-sampled through a multi-layer perceptron (center), resulting in a set of coefficients (depicted as grey dots above). These coefficients are used with an over-complete set of fixed radial basis functions (top right) to describe the raw EKG signal, excluding the inferred pause duration.

probabilistic model

$$\boldsymbol{z}^{(m)}, \boldsymbol{z}^{(p)} \sim p(\boldsymbol{z}^{(m)}, \boldsymbol{z}^{(p)}; \boldsymbol{\theta})$$

$$\boldsymbol{y}_t \mid \boldsymbol{z}^{(m)}, \boldsymbol{z}^{(p)} \ldots = f(\boldsymbol{z}^{(m)}, \boldsymbol{z}^{(p)}; \boldsymbol{\theta}) + \epsilon_t \ , \epsilon_t \sim \mathcal{N}(\boldsymbol{m}_t, \sigma^2)$$

where the variables are

- $\boldsymbol{z}^{(m)}$: the morphology of a patient's beat in a low-dimensional (D) latent space.
- $\boldsymbol{z}^{(p)}$: the pause between cardiac cycles and the length of cycle (in seconds).
- $\boldsymbol{\theta}$: global parameters, including the morphology basis parameters, and prior parameters.
- $\boldsymbol{y}_t$: observed voltage, conditionally Gaussian given parameters and link function $f(\cdot)$.

The pause variable $\boldsymbol{z}^{(p)}$ measures the amount of time on each side of the cardiac cycle that can be explained by a constant offset. The morphology variable $\boldsymbol{z}^{(m)}$ explains the shape of the cardiac cycle measured by the EKG.

Generative model of beat morphology The cardiac cycle exhibits difficult-to-prescribe variation from patient to patient and beat to beat. To address this, we represent a beat's shape with a low-dimensional latent variable that is passed through a set of non-linear basis functions (i.e. a deep neural network), parameterized by $\boldsymbol{\theta}$. The output of the deep generative model is a set of regression coefficients, applied to a fixed, temporally separated and over-complete basis. These two components model the de-noised EKG tracing; the output basis is held fixed to maintain a degree of interpretability. For inference, we use a variational autoencoder-style inference network within a variational inference framework to maximize a lower bound to the marginal likelihood of the data as a function of $\boldsymbol{\theta}$ and recognition network parameters $\boldsymbol{\phi}$ [Kingma and Welling, 2013]. The generative procedure is illustrated in Figure 3.7.

Our inference procedure is two step — we first maximize the data likelihood with respect to latent variables $\boldsymbol{z}^{(p)}$, which finds the per-beat pause and EKG cardiac cycle length for each heartbeat. We do this by using the closed-form posterior mean solution for the regression coefficients with respect to the fixed basis. We can think of this as a sort of alignment procedure — the part of the waveform corresponding to the P-wave, QRS complex, and T-waves can now be jointly modeled.

We then fix this alignment, and fit the deep generative network that produces regression coefficients for the same fixed output basis. For this, we use a multi-layer perceptron with two hidden layers, each with 50 units

$$\boldsymbol{\beta} = \texttt{MLP}(\boldsymbol{z}^{(m)}; \boldsymbol{\theta}) \,. \tag{3.13}$$

Our model of beat morphology is essentially a deep generative regression model. Given generative parameters $\boldsymbol{\theta}$, and a fixed observation basis, $B_1(\cdot), \dots, B_K(\cdot)$, where $B_k(\cdot) : [0, 2\pi] \mapsto \mathbb{R}$ are von Mise-like radial basis functions

$$B_k(\omega; \mu_k, \kappa_k) = \exp\left(\kappa_k \mathrm{cos}(\omega - \mu_k) - \kappa_k\right) \,. \tag{3.14}$$

The data are then generated using $\boldsymbol{\beta}$ and the static basis $B_1, \dots, B_K$ at the points where the EKG tracings are observed. For instance, if we observe samples for a single beat at time points $t_1 \dots, t_T$, and we have inferred the start time and duration of the cardiac cycle, $z^{(p)} = (t^{(start)}, t^{(dur)})$, then the observation can be split into three segments — the pause before the cycle, the cycle itself, and the pause observed after the cycle. For observations within the cycle, we simply transform them to “canonical time” to align with the von-Mise basis

$$\tau_i = (t_i - t^{(start)})/t^{(dur)} \cdot 2\pi \,. \tag{3.15}$$

If there are $T^{(cycle)}$ cardiac cycle samples, then the cycle portion of each beat thus has a corresponding “design matrix” of size $T^{(cycle)} \times K$

$$\boldsymbol{X}_{i,k} = B_k(\tau_i; \mu_k, \kappa_k) \,. \tag{3.16}$$

Conditioned on this design matrix, the observed data are normal with a small error term

$$\boldsymbol{y}_i = \mathcal{N}(\boldsymbol{\beta}^\intercal \boldsymbol{X}_i, \sigma^2)\,. \tag{3.17}$$

The inferential task is to estimate the posterior distribution over $\boldsymbol{z}^{(m)}$ given observations. For this, we use an inference network, which is another multi-layer perceptron (that mirrors the generative network)

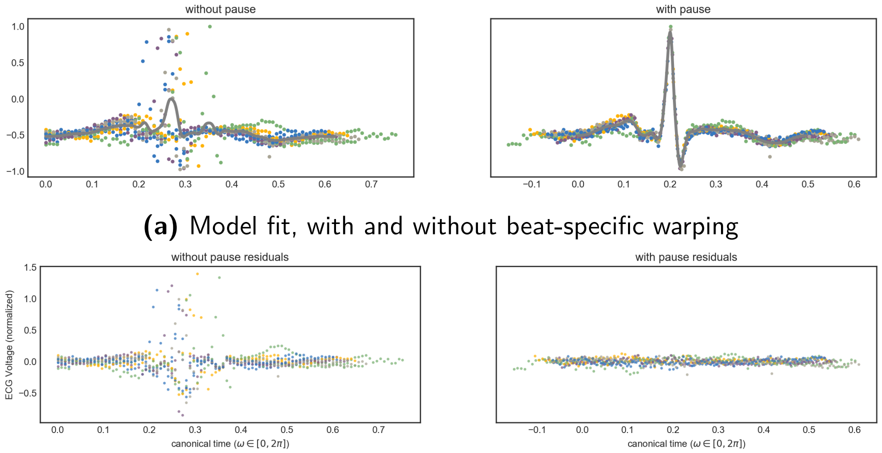
$$\mu_z, \sigma_z = \texttt{MLP}(\boldsymbol{\beta}^{(ols)}; \boldsymbol{\phi}) \tag{3.18}$$

which outputs an estimate of the posterior mean and variance for the latent morphology parameter. The variational objective is now defined with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z} \sim q(\cdot; \boldsymbol{\phi}, \boldsymbol{\beta}^{(ols)})} \left[\ln p(\boldsymbol{Y} | \boldsymbol{z}) p(\boldsymbol{z}) - \ln q(\boldsymbol{z}; \boldsymbol{\phi}, \boldsymbol{\beta}^{(ols)} \right] \tag{3.19}$$

The “data” we use for the inference network is not the observation vector $\boldsymbol{Y}$, but the posterior mean solution for $\boldsymbol{\beta}$ given the fixed basis, $B_1, \dots, B_K$. For our experiments, we fix a basis of size $K = 60$, spatially spread over the interval $[0, 2\pi]$.

Related Work [McSharry et al., 2003] describe a generative model of EKG records defined ordinary differential equations. This model similarly includes a periodic basis, and instantiates an angular velocity to model the quasi-periodicity of the signal. However, inference for datasets of EKG records is not discussed. [Oster et al., 2015] describe a switching Kalman filter approach to EKG modeling, using discrete latent states to cluster similar beat types, in contrast to our continuous latent-space description of EKG beat morphology. This approach uses discrete clusters, while our approach is more similar to a non-linear factor model. [Chia and Syed, 2014] align EKG beats using dynamic time warping, while our approach directly models the biologically plausible sources of temporal variation, the cycle length and pause duration.



(a) Model fit, with and without beat-specific warping

(b) Model residuals, with and without beat-specific warping

Figure 3.8: Top: comparison of model fit without pauses between beats (left) and with pause latent variables (right). The model average beat is shown in grey. Bottom: comparison of model residuals without (left) and with (right) pause latent variables. By modeling the pause duration, the salient features of the cardiac cycle (e.g. the P wave, QRS complex, and T wave) are aligned. Without this alignment, these features can be washed out by inappropriately averaging misaligned features within the cycle.

3.2.2 Empirical Evaluation

We fit our model to the PhysioNet PTB Diagnostic EKG database [Bousseljot et al., 1995, Kreiseler and Bousseliot, 1995]. This dataset contains 549 EKG records from 290 subjects with over 4,600 individual beats. We first look at consequence of inferring beat-specific pauses, and see that it offers a sort of alignment that enables coherent modeling of the morphology. We then examine the latent space inferred by the beat morphology model.

Pause model checks Inferring pause parameters provides an effective way to align the salient features (e.g. P wave, QRS complex, T wave) in a way that can be coherently modeled. Without the pause latent variable, the temporal variability exhibited by the patient washes out features of the beat. With it, the temporal variability is appropriately separated from morphological features, which are modeled with a basis function

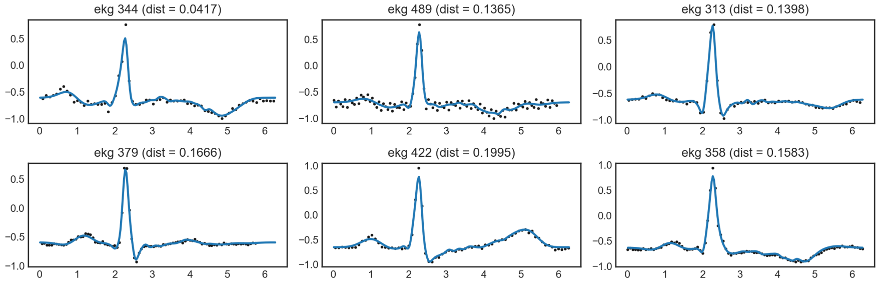
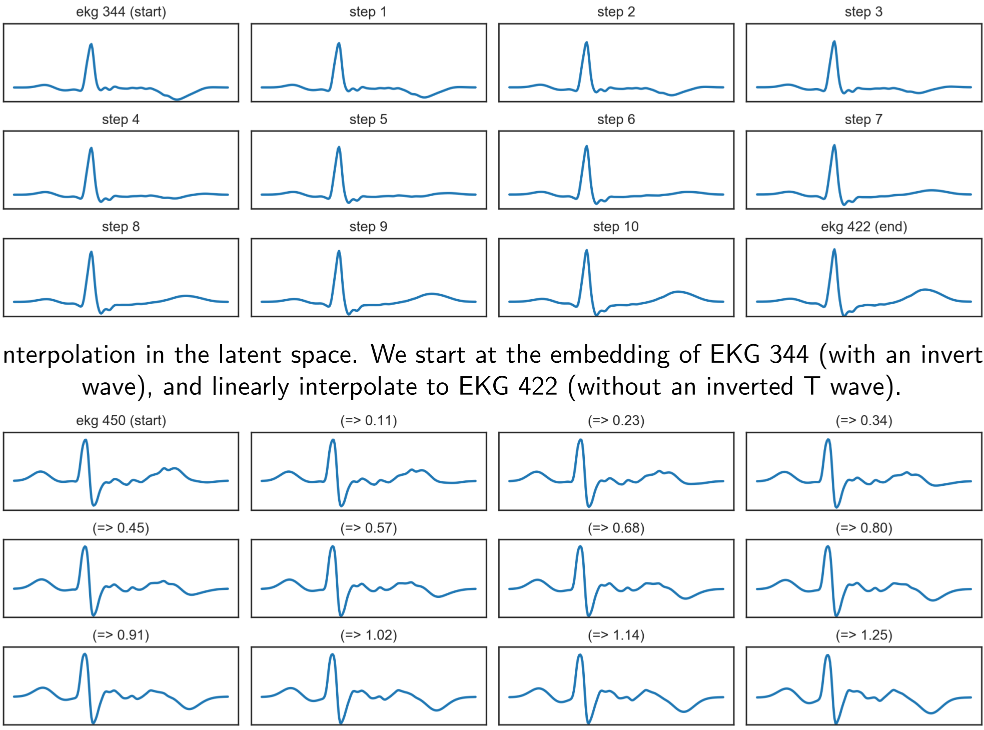


Figure 3.9: Nearest record examples. The source record is in the upper left (EKG 344). The following five are example beats from the nearest neighbors in latent $\boldsymbol{z}^{(m)}$ space. The solid line is the generative model reconstruction.

regression model. In Figure 3.8 we see that residuals around the QRS complex decrease significantly when we incorporate a model of the pause between beats.

Nearest Neighbor Evaluation We examine the nearest neighbors (in different EKG records) in the latent morphology space, $\boldsymbol{z}^{(mor)}$. In Figure 3.9, we show a source beat and depict the five nearest neighbors in the latent morphology space. We see that the similarity in distance for this example corresponds in part to similarity in T-wave direction, while the rest of the beat remains unchanged.

Interpolation To illustrate the generative capacity of our model, we visualize the latent path between two beats. In Figure 3.10a, we start out at the $z^{(m)}$ value of an observed beat, and linearly interpolate to the $z^{(m)}$ value of another observed beat. Each transition beat is generated from our model — we see that following this direction corresponds to inverting the T wave. To further explore this concept, we take a beat with a different morphology (and standard T wave) and follow the direction "T wave inversion" direction found in the previous example. Similar to neural word embeddings [Mikolov et al., 2013], we find that following this direction inverts the T wave while leaving other features of the morphology relatively unchanged, shown in Figure 3.10b.



(a) Interpolation in the latent space. We start at the embedding of EKG 344 (with an inverted T wave), and linearly interpolate to EKG 422 (without an inverted T wave).

(b) Generated beats along the T wave inversion direction determined by EKG 422 to EKG 344 (above), starting from very different EKG 450. We see that this direction does correspond to T wave inversion, leaving other features relatively unchanged.

Figure 3.10: Exploring the cardiac cycle morphology latent space.

Identifying Patients A subset of patients have multiple EKG records in the PTB dataset. We expect a good representation of EKG beats to cluster together the same patients across different records. To test this, we compare the average distance of $\boldsymbol{z}^{(m)}$ between the same patient across different EKG records ($d^{(same)}$) to the average distance of $\boldsymbol{z}^{(m)}$ to some random EKG record ($d^{(diff)}$). We compare this difference to the same value measured using PCA, and find that our model measures the same patient to be significantly closer.

We measure the average distance of $\boldsymbol{z}^{(m)}$ between the same patient across different EKG records ($d^{(same)}$) to the average distance of $\boldsymbol{z}^{(m)}$ to some random EKG record ($d^{(diff)}$). We compare this difference to the same value measured using PCA on the least squares regression coefficients for the fixed final layer, with the same number of latent dimensions. We find that our model measures the same patient to be significantly closer on average.

For latent dimension $D = 10$, we measure the two averages as

$$\mathbb{E}[d^{(same)} - d^{(diff)}] = -.037 \in [-0.049, -0.025] \qquad \text{PCA} \tag{3.20}$$

$$\mathbb{E}[d^{(same)} - d^{(diff)}] = -.100 \in [-0.118, -0.083] \qquad \text{VAE} \tag{3.21}$$

indicating that the VAE significantly improves over PCA. We draw the conclusion that a strictly linear model of EKG beats (given our fixed observation basis) may not be expressive enough to carry patient-specific information, compared to a non-linear latent factor model.

3.2.3 Discussion

We developed a latent variable model for large datasets of electrocardiogram records using a flexible deep neural network component and an interpretable pause and beat duration component. In preliminary model exploration, we show that the latent morphology space encodes information about patient similarity and physiological features that correlate to biological processes. We plan to further criticize our model and hope to predict patient outcomes that are less easily observed.

This next probabilistic modeling project finds latent structure in the team play of professional basketball players, given observations of their trajectories. This section is based on previously published work [Miller and Bornn, 2017].

3.3 Application: Trajectory Modeling

Player-tracking data present a unique challenge for basketball analytics. It is widely believed that a windfall of quantitative insight is hidden in these data, in spatiotemporal patterns that coaches and analysts typically process with human intuition. While there has been work toward quantifying player ability [Franks et al., 2015a], possession value [Cervone et al., 2014a,b], and play classification based on small sets of labeled plays [Wang and Zemel, 2016], methods for *automatically* organizing, summarizing, and interpreting basketball possessions have yet to be fully developed.

As an example, consider the following use case for defensive scouting: an analyst is tasked with finding all possessions in which James Harden drives to the basket and passes the ball to a teammate for a right corner three-point attempt. Simple engineering solutions for this scenario are easy to imagine: first sub-select Rockets possessions with a right corner three-point attempt and then look for passes from Harden that originate in the paint. However, adding search criteria quickly renders this ad hoc solution intractable: find sequences where Harden uses a high screen before driving to the basket and then passes to the corner for a three-point attempt; find sequences where Harden uses a high screen, drives to the basket, passes to the corner and that teammate drives to the basket; find sequences where *any Rocket* uses a high screen, drives to the basket, etc. The landscape of relevant basketball scenarios is far too vast and complex for ad hoc search solutions.

Furthermore, this type of sequential query is only one approach to gaining insight from player-tracking data. We can imagine starting a research project by simply asking — what leads to a corner three? What sort of patterns are employed by different offenses in order to get an open three-point attempt? What sorts of actions do specific players tend to do in order to generate an open three-point attempt? Existing methodology falls short of supporting this kind of exploratory analysis with player-tracking data.

In this work, we bridge this gap by formulating a novel machine learning method to describe an entire database of player-tracks. Our method uncovers characteristic patterns of offense in a way that is searchable and interpretable. We first describe individual player's actions by building a data-driven dictionary of *action templates* derived from a statistical model. We then construct a model of possessions that describes patterns in these action templates—common co-occurrences that create a signature of offensive strategy. For each play, this yields a *possession sketch*, a concise summary of the offense's actions in a basketball possession. We model this structure at multiple levels—in dynamic actions taken by individual players, as well as collective actions present in each possession.

Importantly, we construct our model out of interpretable pieces—each *action template* can be interpreted as a type of on- or off-ball cut. Further, pairs of actions are also interpretable — some correspond to on-and off-ball screens, others correspond to drives and passes to various wings. Our use of probabilistic graphical models on an interpretable representation of the data allows for easier-to-understand model output and inferences than recent deep learning approaches [Wang and Zemel, 2016].

In the following section we describe the components of our method that generate *action templates* and *possession sketches*. After describing our method, we explore the structure it reveals by looking at three of the different organizational tools it makes possible:

- *team possession maps*: low-dimensional visualization of all of the offensive possessions of a team—exploring this map reveals different set calls used by a team.
- *shot possession maps*: low-dimensional visualization of possessions that led to a particular type of shot—we examine the different types of actions that lead to corner threes.
- *possession basis*: common and repeated actions discovered by the model — this establishes the types of player interaction that make up the "vocabulary" of a basketball possession.

By integrating scalable probabilistic modeling and visualization, this work shows that we can organize and systematically explore NBA possessions, allowing us to derive useful basketball intelligence from the NBA's vast and growing store of player-tracking data.

3.3.1 Methods

This section details the machine learning model we construct to recognize patterns at two resolutions: spatiotemporal patterns in individual player trajectories (*action templates*), and co-occurrence of actions in each possession (*possession sketches*).

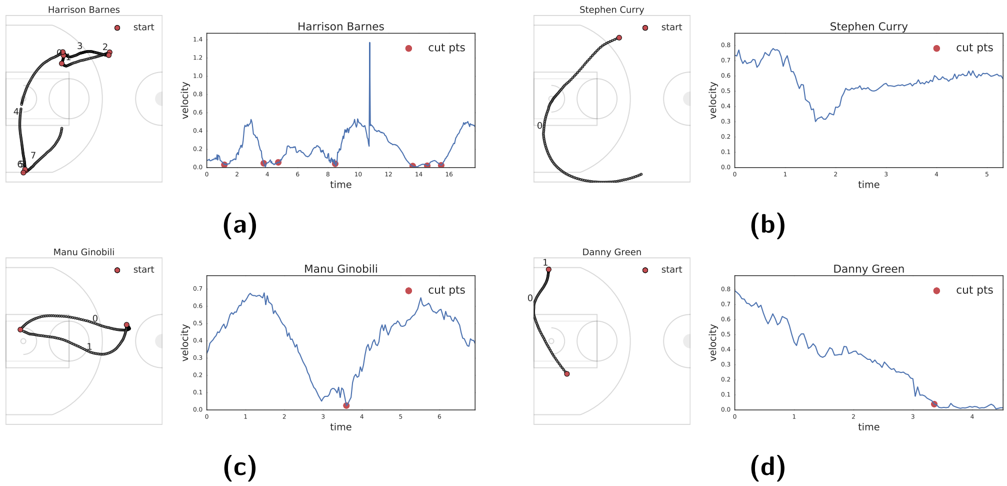
Before we go into further detail, the overall procedure behind our method can be decomposed into the following steps:

- *Segmentation*: We cut possession-length (e.g. 5-24 second) player trajectories into shorter, more manageable segments based on moments of sustained low-velocity.
- *Learning action templates*: We formulate a novel statistical clustering algorithm to learn which action is represented by each short segment.
- *Possession modeling*: We represent each possession as a "bag" of pair-actions, and fit a possession-level hierarchical model inspired by the document modeling and natural language processing literature.

The following subsections describe the process of applying the above steps to a large data set of basketball player-tracks.

3.3.2 Data and preprocessing

We analyze a database of player-tracks from the 2014-2015 season of the NBA. The data are organized into over $N = 190{,}000$ possessions (and possessions into quarters and games). For each possession (indexed by n), we model the trajectories of players on offense. For each player (indexed by j) in possession n, we cut their trajectory (denoted $\boldsymbol{x}_j^{(n)}$) into short segments at locations of sustained low-velocity. To do this, we first detect moments of low velocity by inspecting the smoothed first difference of the trajectory. At sustained moments of low velocity ($>$ .25 seconds below a threshold of



(a) **(b)** **(c)** **(d)**

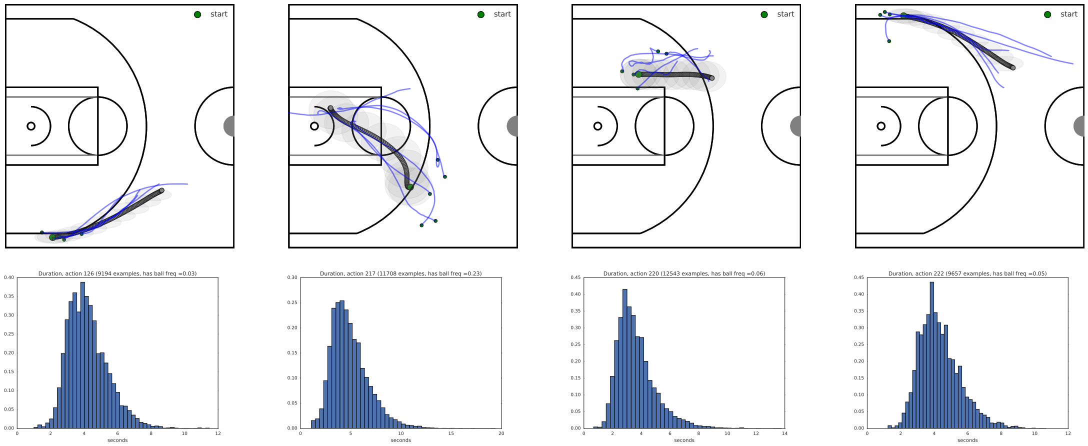
Figure 3.11: Examples of trajectory segments resulting from the "sustained-low-velocity-moment" finding algorithm. In each example, the left plot depicts the spatial trajectory with cut points denoted by the red dots (with the order of the cuts labeled). The right plot depicts the approximate magnitude of the velocity at each moment during the possession, with the corresponding cut points.

.1 feet per second), we cut the possession, resulting in a collection of shorter segments. Figure 3.11 depicts four example trajectories, cut into various number of segments.

We refer to these shorter trajectory segments as $\boldsymbol{x}_{j_1}^{(n)}, \ldots, \boldsymbol{x}_{j_S}^{(n)}$, where it is understood that the number of segments, S, varies for each possession-player pair. The resulting short segments are on average 2.25 seconds (the interior 95 percentiles range from 0.6 to 7.96 seconds). Applying this preprocessing step to the full 2014-2015 regular season creates a data set of roughly 4.5 million segment observations.

3.3.3 Action Templates: Segment Clustering

Our method assumes that each short trajectory segment represents some discrete *action*, and each player performs a series of actions throughout the course of a possession. For instance, a player might (i) make a cut along the baseline and then (ii) camp out in the corner. Alternatively, a player can (i) make a cut along the 3-point line, (ii) stand at the break, then (iii) cut toward the basket. In order to decompose a player's trajectory



(a) Action 126 **(b)** Action 217 **(c)** Action 220 **(d)** Action 222

Figure 3.12: A sampling of *action templates*. Our method automatically builds a taxonomy of commonly repeated movements shared among all players (i.e. *actions*). In each column, the top plot depicts the spatial trajectory for a single action template. The light blue lines are real segment trajectories that fall in that cluster. Below each action plot is a histogram of segment lengths (in seconds) for all segments that fall into that cluster — some actions are shorter or longer (on average) than others. For a more dynamic picture of an action template, please view this animated figure: `https://youtu.be/-a6_0t6etmk`

into a set of actions, we must first infer a meaningful set of actions that all players share. We use a data-driven approach to infer this set of actions, each action's structure, and the action label for each trajectory segment.

To accomplish this, we construct a probabilistic clustering algorithm tailored for functional data (i.e. continuous trajectories). Our model posits that each trajectory segment represents one of V discrete actions, where each action is characterized by a *template*. Each template can be thought of as a cluster center—each observed trajectory segment is centered around a template with some deviation. We specify each template as a *Bezier curve*—a tool commonly used to model movement in the computer graphics community – which specifies a function $B(t)$ that maps time to a two-dimensional point, $B : [0, 1] \mapsto \mathbb{R}^2$. This maps out a dynamic curve through space, which describes the movement of each action.

The clustering model specifies V Bezier curve components, $B_v : [0,1] \mapsto \mathbb{R}^2$, each parameterized by $\theta_v \in \mathbb{R}^{P\times 2}$, where P is the number of *control points* used to characterize the curve.[1]

$$B_v(t; \theta_v) = \theta_v^\intercal D_P(t) \tag{3.22}$$

$$D_P(t) = \binom{P}{p} t^p (1-t)^{P-p} \qquad \text{for } p = 0, \dots, P-1. \tag{3.23}$$

Importantly, each curve can be specified as a linear function with respect to parameters θ_v, with a non-linear (but fixed) basis in time, $D_P(t)$. Bezier curves are a natural choice for these data — they are flexible, concisely parameterized, and easy to fit. The non-linear basis in time allows for a wide variety of template shapes.

The complete functional clustering model is specified as

$$z_{j_s}^{(n)} \sim Pr(action|\pi) \qquad \text{action type} \tag{3.24}$$

$$\boldsymbol{x}_{j_s,t}^{(n)} \sim \mathcal{N}(B_v(t, \theta_v), \Sigma_v) \qquad \text{location at moment } t \tag{3.25}$$

We use maximum likelihood to learn parameters θ_v, π, and Σ_v (and therefore each action) directly from the data set of 4.5 million trajectory segments. To do so efficiently, we devise expectation maximization [Dempster et al., 1977] updates that exploit the linear structure of Bezier curves—each maximization step can be computed using weighted least squares. Furthermore, each expectation step can operate on each segment in parallel, allowing us to scale our method up to the 4.5 million trajectory segments.

Figure 3.12 depicts a sampling of learned templates resulting from fitting a mixture of $V = 250$ Bezier curves to the processed trajectory segments. The output of this model allows us to succinctly represent each trajectory segment as a single integer, $v = 1, \dots, 250$. We view these actions as a kind of *vocabulary*—each possession com-

[1] More control points allow for more flexibility in fitting shapes — we use 10 control points in our experiments

bines words in the vocabulary to describe structured interactions that characterize the possession. Following this thread, we turn to statistical methods originally devised for modeling documents, and adapt them to basketball sequences.

3.3.4 Possession Model

Offensive possessions are highly structured. When James Harden drives toward the basket, drawing defender attention, his teammates are not distributed randomly on the floor — it is likely that at least one teammate is in the corner waiting for a pass; it is likely that other teammates vacate the paint, and begin jockeying for rebounding position. The structure of an offensive possession is created by the individual actions that each player performs throughout the possession. Which actions tend to simultaneously co-occur? Which actions tend to precede or follow other actions? Our possession model seeks to answer these questions by first observing that these actions are a lot like words. Words are interwoven sequentially to express a coherent idea; player actions are interwoven sequentially to implement a coherent strategy. We run with this analogy by adapting *topic models* [Blei, 2012] to describe sequences of actions in basketball possessions.

We use Latent Dirichlet Allocation (LDA) [Blei et al., 2003], a topic model for unsupervised structure discovery in a corpus of text documents. LDA is a latent factor model, similar to factor analysis or principal components analysis. In document modeling, LDA describes each document as a mixture of *topics*, where each topic is a distribution over the entire vocabulary of words. As a concrete example, LDA applied to a corpus of *Science* articles finds topics corresponding to *cancer* (e.g. probable words are "tumor", "cell", "cancer", etc.), and *neuroscience* ("synaptic", "neurons", "hippocampal", etc.), among many others (see [Griffiths and Steyvers, 2004, Blei, 2012]).

Conceptually, LDA defines K *topics*, ϕ_k, each a distribution over *actions*. Each observed possession is characterized by some latent distribution over *topics*, $\pi^{(n)}$, which describes the probability that a particular topic is expressed in possession n. These two distributions — possession-specific proportions and global topics — determine the probability of observing any particular action in possession n. LDA posits the following data generating process to give rise to the matrix of counts

$$\phi_k \sim \texttt{Dir}_V(\alpha_0) \text{ for } k = 1, \ldots, K \tag{3.26}$$

$$\pi^{(n)} \sim \texttt{Dir}_K(\alpha) \text{ for } n = 1, \ldots, N \tag{3.27}$$

$$Y_{n,:} \sim \texttt{Mult}\left(M_n, p = \sum_k \pi_k^{(n)} \phi_k\right) \tag{3.28}$$

where M_n is the total number of actions present in possession n (a fixed constant). We use statistical inference techniques to infer both the global topics, Φ, and the possession-specific proportions, $\pi^{(n)}$ for all possessions. Due to the size of the dataset, we use stochastic variational inference [Hoffman et al., 2013], a scalable method for Bayesian inference in hierarchical models.

In our application, rather than topics, we represent each possession as a mixture of *strategies*, where each strategy is a distribution over co-occurring actions that are frequently observed in the data. We then use LDA to infer the strategies employed in each offensive possession (as well as the set of strategies themselves). LDA requires that we represent each possession as a "bag of words" — a vector where each entry corresponds to a unique word and represents the number of times that word occurs in the possession. To do this, we need to first establish a vocabulary.[2] Our first approach was to simply count the number of each $v = 1, \ldots, V$ actions in each possession. This approach is appealing in its simplicity, and does reveal interesting structure. However, this representation ignores interactions between players and temporal structure.

In this work we use a vocabulary of *pair-actions*, where each "word" in the vocabulary

[2] In document modeling, the vocabulary is typically the vocabulary of the language itself, with minimal preprocessing. Common sequences of two or three words (bi-grams and tri-grams) are sometimes included in the vocabulary to improve the model.

is a unique pair of the V actions, (v_i, v_j) for $v_i, v_j \in \{1, \ldots, V\}$ and $v_i \neq v_j$. We then represent each possession as a "bag of simultaneous pair-actions", mapping the "bag of words" concept from topic models to basketball interactions. For each possession, we simply count the number of times each unique pair of actions simultaneously occur. We string these counts into a single vector, which represent possession n

$$Y_{n,d} = \#\text{ times action action pair } d = (v_1, v_2) \text{ appears in possession } n. \tag{3.29}$$

Figure 3.13 illustrates the construction of our pair-action vocabulary that we use to succinctly represent each possession. To incorporate ball possession information, we define each action as "with" or "without" the ball[3], resulting in $2 \cdot V$ total player actions. We include pair-actions that appear in at least 100 possessions, resulting in about 25,000 unique pair-actions in our vocabulary. This representation allows us to apply LDA to basketball possessions. To fit this model to the over 190,000 possessions in the season, we use a recently developed scalable Bayesian inference technique [Hoffman et al., 2013].

This model yields a low-dimensional embedding of every NBA play that allows us to quickly assess similarities between possessions and explore the space of team offensive strategies. We can create interactive graphics (a dynamic version of Figure 3.15a), where each point in space represents a full possession and nearby points indicate "similar" possessions — possessions that share the same pattern of actions. The following section dives deeper into this exploration tool, and what it can afford an analyst. The *topics* themselves encode strategic co-occurrences of actions, and using these topics we can shed light on the fundamental building blocks of collective action on the basketball court. Inspecting these topics can help us quantify what exactly makes a unique offense unique.

[3] An action is considered "with" ball if the player possesses the ball for the majority of the segment.

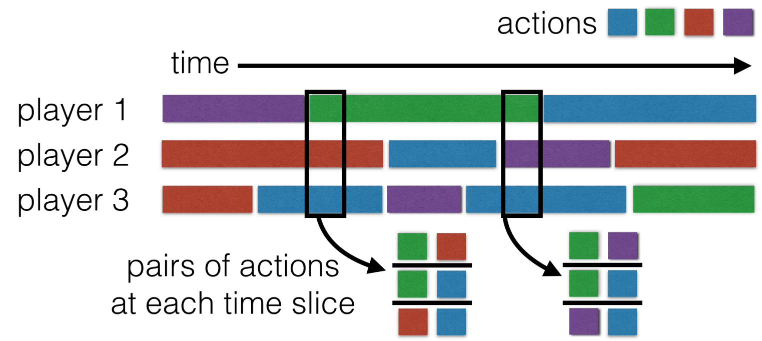
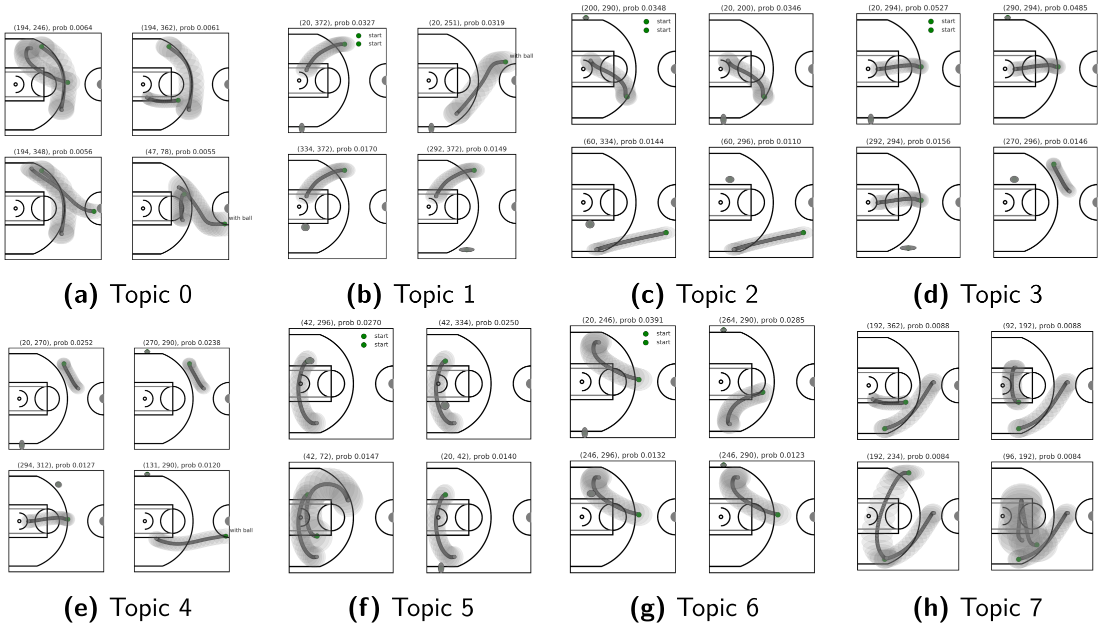


Figure 3.13: The "bag of words" construction of each possession. Each "word" represents two actions that occur simultaneously throughout the course of the possession, where actions are inferred with the *action template* model presented in Section 3.3.3. In the toy example depicted, we have three players, each performing a sequence of actions (corresponding to the four colors). At each moment in time, we enumerate all unique pairs of actions. We represent the entire possession as a bag of these pair-action counts.



(a) Topic 0 **(b)** Topic 1 **(c)** Topic 2 **(d)** Topic 3

(e) Topic 4 **(f)** Topic 5 **(g)** Topic 6 **(h)** Topic 7

Figure 3.14: The result of fitting a $K = 100$ topic possession model. A "topic" in our framework corresponds to a distribution over *pairs* of actions. Above, we show common *pairs* of actions from 8 of the 100 topics. We observe that topics tend to pick up on combinations of actions that include common actions. For instance, the top two pair-actions in topic 0 includes a cut along the 3-point line while a teammate cuts nearby (perhaps setting an off-ball screen).

3.3.5 Analysis

In this section we explore the output of the possession level model to see which patterns are represented. We focus on the following aspects of model output

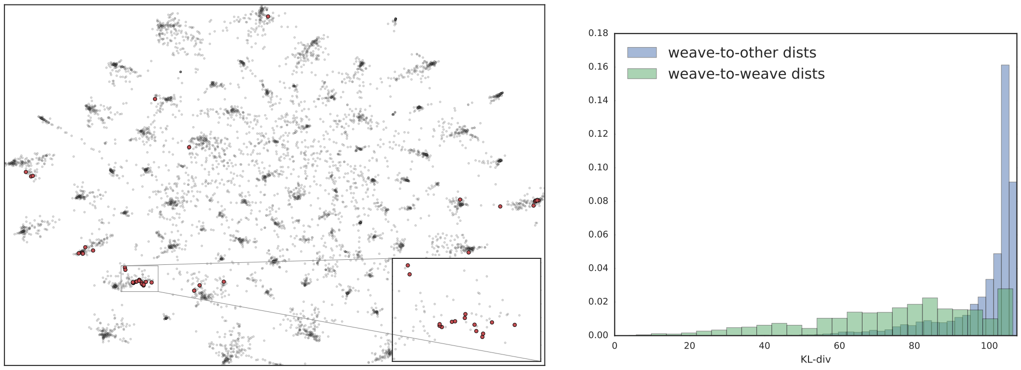
- *basketball topics*: we see which *pair-actions* are represented by each of the $K = 100$ topics. This tells us not only which pair-actions occur frequently, but which pair-actions *co-occur* in possessions, revealing fundamental patterns of basketball offenses.
- *possession sketch*: each possession is characterized as a distribution over topics (or *strategies*), and "similarity" between possessions can be measured using this distribution. We explore what our model describes as similar, and we empirically test this notion of similarity by measuring distances between sets of plays we previously inspected and labeled as similar.

In the following sections we explore the above concepts by visualizing and exploring possessions in ways newly afforded by our framework.

Basketball Topics Figure 3.14 graphically depicts a small sampling of *basketball topics* (i.e. strategies) discovered by the possession model. The topics reveal which pair-actions are most common in our data set, and we do see patterns emerge. As a concrete example, if a particular possession "loads" onto topic 3[4] then that possession is more likely to include the pair-actions depicted in Figure 3.14d—a cut to the basket while a teammate is standing in either of the two corners. Topic 5 prominently includes possessions with a baseline cut from the right block to the left break. Note that there are *many more* pair-actions with significant probability than the ones depicted, and there are many more topics than we depict.

We also notice that each possession topic vector is quite sparse—on average only 8 of the 100 entries are non-zero. This is expected and desired behavior—each possession can only include a small number of offensive patterns from the wide array of available tactics.

[4] i.e. the possession sketch vector is large along the dimension corresponding to topic 3



(a) Warriors Possessions (t-SNE map)

(b) Weave-to-weave and weave-to-other distances

Figure 3.15: Left: map of 2014-2015 Warriors possessions, with a small set of known "weave" plays highlighted in red. The weave plays tend to cluster together in this visualization. We verify this by computing the average distance between two weave possessions and between a weave and a random Warriors possession of a similar length. This indicates that our topic-model-based representation is picking up on patterns that are able to (mostly) distinguish between semantically different plays.

Possession Map Exploration Each possession has an associated *possession sketch*—a per-topic vector that describes how much of each of the *basketball topics* (a subset illustrated in Figure 3.14) are featured in that possession. We can use these possession sketches to reason about large sets of basketball possessions. In this section we select the offensive possessions of the 2014-2015 Golden State Warriors (over 6,000 possessions). With each possession succinctly described by a (sparse) 100-dimensional topic vector, we use the dimensionality reduction technique t-SNE [**?**] to visualize these vectors in 2-dimensions. This method finds a 2-dimensional representation of each 100-dimensional vector such that the distance in 2-d is similar to the distance in 100-d (emphasizing the preservation of local distances).[5] Figure 3.15 visualizes all Warriors possessions in 2014-2015.

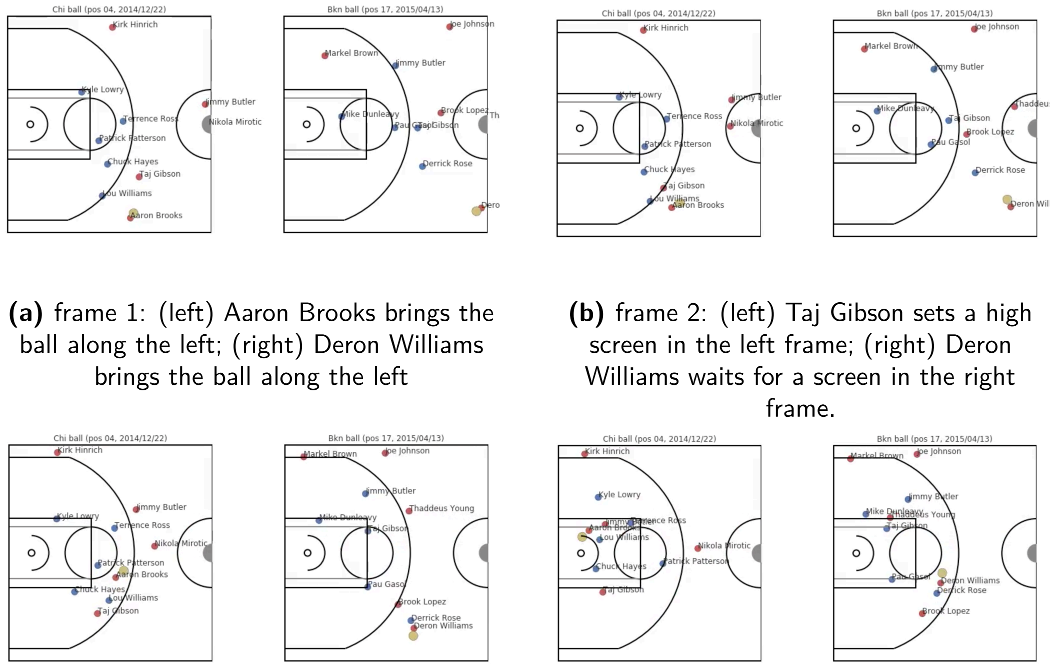
We test the notion of "similarity" in topic space by examining a group of hand-labeled

[5] For intuition, t-SNE tends to yield a visualization where locally clustered points are close in distance in the full, 100-dimensional topic space; points that are farther away from each other tend to be far, but could also be close.

set plays, a “weave play”. We animate two examples of the weave play in this animated figure: `https://youtu.be/KRDsTLMm7FY`. We hand-label 40 weave plays in the 2014-2015 season, and visualize them in the t-SNE Warriors map (Figure 3.15a, in red). We can visually verify in Figure 3.15 that the possession sketch preserves this notion of similarity—weave plays tend to cluster around other weave plays.

We can further measure this clustering by comparing two distributions of possession sketch distances: (i) the distribution of distances between two weave plays, and (ii) the distribution of distances between one weave, and one non-weave play. Figure 3.15b illustrates these two distributions. The average distance between the known weave plays is much smaller than the average distance between weave and non-weave plays. In fact, the nearest neighbor of each weave play is most often itself a weave play, highlighting the potential of our technique to quickly find a collection of plays similar to a chosen play.

Between Team Nearest Neighbors Our method also identifies similar possession structure between different teams. To highlight this we select a play at random, and search through the entire database of 190,000 possession sketches to find the most similar play. The resulting two possessions are compared in Figure 3.16. Chicago is on offense in our first possession, and Brooklyn is on offense in the nearest-neighbor possession. Examining these two possessions, we see a few salient similarities that shed light on what patterns our method is detecting: (i) the point guard brings the ball up the left side of the floor in each possession; (ii) a player sets a high screen on the left side, and the point guard curls around the screen toward the middle with the ball; (iii) through both possessions a player camps out in the weak-side corner three; (iv) the point guard attacks through the middle of the paint. The possession sketch contains this information — and we can further inspect the particular basketball topics for this possession to see how this information is summarized in our model.



(a) frame 1: (left) Aaron Brooks brings the ball along the left; (right) Deron Williams brings the ball along the left

(b) frame 2: (left) Taj Gibson sets a high screen in the left frame; (right) Deron Williams waits for a screen in the right frame.

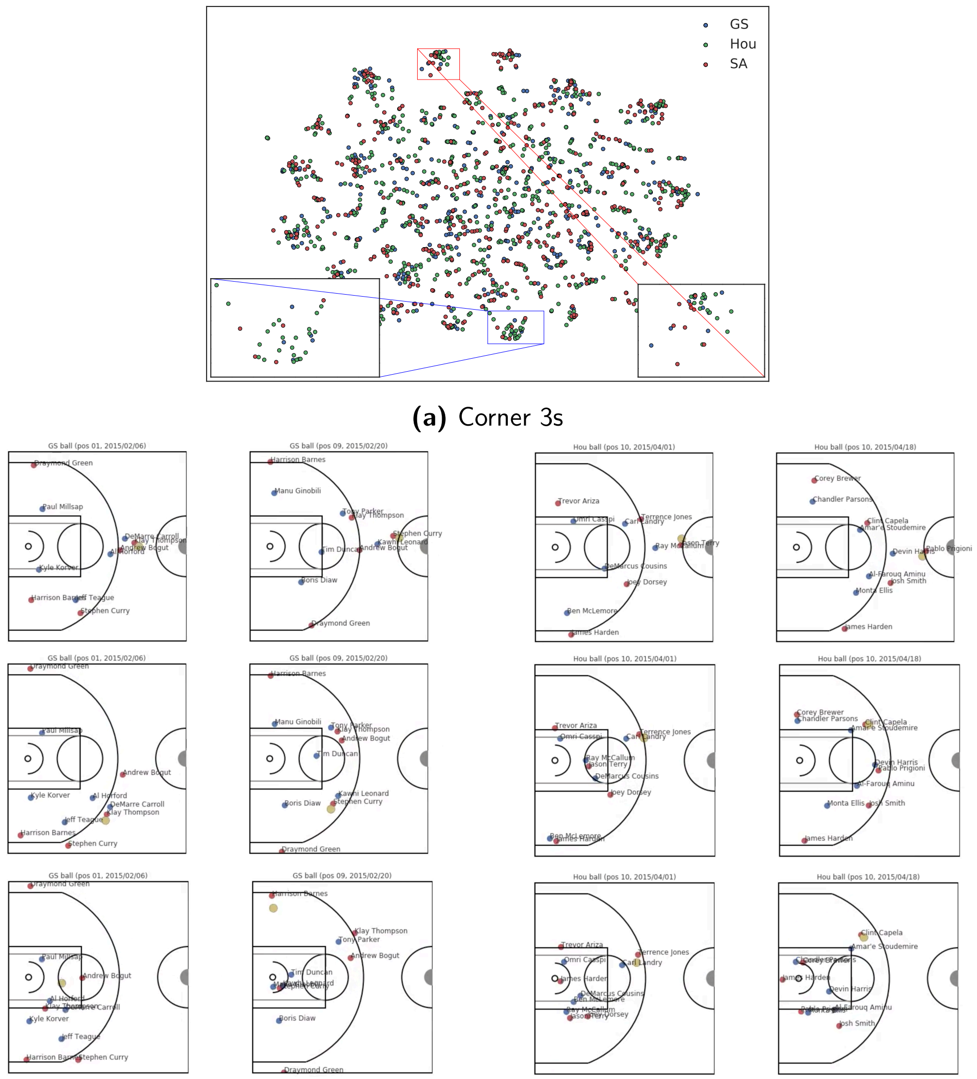
(c) frame 3: (left) Aaron Brooks curls around the screen and drives; (right) Brook Lopez sets a high screen for Deron Williams

(d) frame 4: (left) Aaron Brooks attacks the basket; (right) Deron Williams curls around the screen and drives

Figure 3.16: An example of two very similar possessions: each sub-figure displays key frames from two possessions — one where Chicago has the ball and one where Brooklyn has the ball. These frames highlight similar features between the two possessions. For a clearer picture of "possession similarity", please navigate to https://youtu.be/0Jlj6xekxeI to see these plays animated.

Corner Threes In this section we explore possession sketch similarity in the context of a particular type of shot—a corner three. We first sub-select the 2014-2015 data to possessions that include corner three-point shots for three teams: the Warriors, the Rockets, and the Spurs. We then apply t-SNE to visualize the sketch for each possession in Figure 3.17a. We immediately notice that the possession sketches that lead to corner threes overlap significantly between teams, however there are some regions of the space in which the Rockets are more likely to inhabit than the Spurs.

We examine the structure of the possession-map clusters by zooming in on two groups



(a) Corner 3s

(b) Left cluster example: key frames

(c) Right cluster example: key frames

Figure 3.17: Corner Three. The left pane cluster examples are similar in that they include a drive to the basket, and a pass to a teammate camping out in the corner. The right pane cluster examples are similar in that they include a baseline cut toward the corner in which the shot is taken. Please see the animated figures at https://youtu.be/hUuPkE06rX4 (left), and https://youtu.be/mMcWuqgrj1w (right).

on the opposite side of the map. Figure 3.17 compares two possessions in the cluster in the left-pane to two possessions in the cluster in the right pane. An immediate difference between the two clusters is that the right pane includes a baseline cut toward the corner in which the shot is taken, whereas the left pane includes a drive into the middle, and a pass out to a player camping out in the corner.[6] Indeed, these are two very different ways of ending up with a corner three point attempt, and our method identifies this and allows us to efficiently explore this structural variation.

3.3.6 Discussion

Related work This paper develops a framework for exploring interpretable patterns in player-tracking data—applications of this framework can enhance player evaluation and media consumption. A similar system for measuring play similarity was developed in [Sha et al., 2016], based on point-wise similarities in trajectories. Ours is a more global approach—we fit a probabilistic model to an entire season's worth of player tracking data, directly modeling player interactions. The result is a more interpretable, succinct, and scalable decomposition of possessions.

In [Cervone et al., 2014a,b], the authors propose a stochastic process model to measure the moment-by-moment expected possession value (EPV) of a basketball sequence. They handcraft a set of basketball states that are used in the model. Our approach is more of a data-driven decomposition of basketball states that we use for exploration (but could be used within an EPV model). Other examples that develop data-driven representations from player-tracking data can be found in [Miller et al., 2014, Franks et al., 2015b,a].

Future work and conclusion There are multiple avenues for future work. Firstly, we can improve the *action template* model by also inferring the number of actions using more sophisticated methods, such as Bayesian nonparametrics. The action templates should also have more temporal structure—auto-correlation and dynamic variance. Fur-

[6] Please refer to animated figures https://youtu.be/hUuPkE06rX4 (left pane) and https://youtu.be/mMcWuqgrj1w (right pane).

ther, our possession sketch ignores much of the temporal information in each possession (a trade-off for statistical and computational efficiency). A future project could further describe the time-varying nature of possession strategies, which, for example, would allow us to identify which possessions may have started out in a “weave” set, but broke down into a different sequence.

Insight derived from player-tracking data has been promised more than delivered. We reduce this gap by devising a method that will have a profound impact on the use of player-tracking data for analysis—from summarizing situational statistics (e.g. how often did the “weave” play succeed?), to searching for similar plays (e.g. for post-game analysis), to discovering and quantifying previously unknown habits of interaction between players (e.g. for team-specific scouting).

3.4 Conclusion and Discussion

Despite the differences in application areas and motivation, the utility and challenges of probabilistic models are universal. Having motivated on of expressive probabilistic models, the computational problem of statistical inference becomes the practical limiting factor. It is far easier to imagine a complicated probabilistic model than it is to obtain accurate and computationally tractable estimates of the model structure given observations. In the next few chapters, we develop new methods to address some common shortcomings of approximate Bayesian inference methods.

4

Improving Posterior Approximations: Variational Boosting

Variational inference algorithms typically fix a family of probability distributions as the variational approximating family *a priori*. A particular family is often chosen because it is *tractable*—we can simulate samples from this distribution and evaluate its density point-wise.

In general, the true posterior distribution is not contained within the variational approximating family. This mismatch will lead to a gap between the optimal variational approximation and the true posterior distribution. Consequently, this approximation gap will induce bias in our estimates of posterior expectations of the form in Equation 2.65.

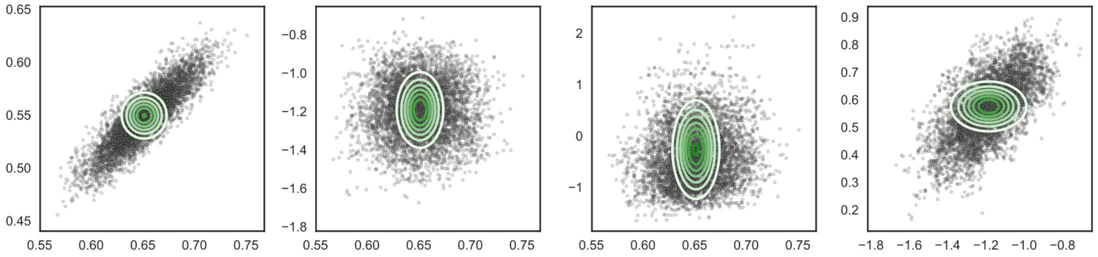


Figure 4.1: Example of under-estimated posterior variances induced by an insufficiently expressive variational approximation, a Gaussian mean-field family. Depicted are four bivariate marginals from the same 37-dimensional posterior distribution. The grey points depict samples simulated by MCMC (and can be thought of as a surrogate for the true posterior). The green contours depict the optimal mean field approximation to the posterior. We can see that the marginal variances are under-estimated in each case because the approximating distribution is unable to capture the correlations present in the posterior. This under-estimation of variance is typical of mean-field approximations.

Perhaps the most common approximating family is a diagonal multivariate normal, or the Gaussian mean-field family. This family imposes strict limitations, including the inability to capture posterior correlations and Gaussianity itself. When this approximating family is used, a common type of bias to see is the (sometimes severe) under-estimation of the marginal posterior variance of each variable. In Figure 4.1 we depict a common example of the biased estimates of posterior variance induced by the mean field assumption. When variables are highly correlated in the posterior, the optimal Gaussian mean-field approximation tends to cover only a small region of the mode (a consequence of the KL-divergence criterion typically used), and reports a much smaller variance for each variable than is present in the true posterior.

In this chapter, we develop a general method for iteratively building more expressive posterior approximations that enables a tradeoff between approximation accuracy and computational cost—similar to the tradeoff that implicitly exists for Markov chain Monte Carlo methods as more correlated samples are simulated. The material in this chapter was published in Miller et al. [2017c].

4.1 Introduction

Variational inference (VI) is a family of methods to approximate an intractable *target* distribution (typically known only up to a constant) with a tractable *surrogate* distribution [Blei et al., 2017a, Jordan et al., 1999, Wainwright and Jordan, 2008]. VI procedures typically minimize the Kullback-Leibler (KL) divergence between the approximation and target distributions by maximizing a tractable lower bound on the marginal likelihood. The approximating family is often fixed, and typically excludes the neighborhood surrounding the target distribution, which prevents the approximation from becoming arbitrarily close to the true posterior. In the context of Bayesian inference, this mismatch between the variational family and the true posterior often manifests as underestimating the posterior variances of the model parameters and the inability to capture posterior correlations [Wainwright and Jordan, 2008].

An alternative approach to posterior inference uses Markov chain Monte Carlo (MCMC) methods that approximate a target distribution with samples drawn from a Markov chain constructed to admit the target distribution as the stationary distribution. MCMC enables a trade-off between computation and accuracy: drawing more samples makes the approximation closer to the target distribution. However, MCMC algorithms typically must be run iteratively and it can be difficult to assess convergence to the true target. Furthermore, correctly specifying MCMC moves can be more algorithmically restrictive than optimization-based approaches.

To alleviate the mismatch between tractable variational approximations and complicated posterior distributions, we propose a variational inference method that *iteratively* allows the approximating family of distributions to become more complex. Under certain conditions, the proposed approximations are eventually expressive enough to represent the true target arbitrarily well (though we do not prove our algorithm attains such a universal approximation here). Thus, the practitioner can trade time fitting a posterior approximation for increased accuracy of posterior estimates. Our algorithm grows the complexity of the approximating class in two ways: 1) incorporating rich covariance structure, and 2) sequentially adding new components to the approximating distribution. Our method builds on black-box variational inference methods using the *re-parameterization trick* by adapting it to be used with mixture distributions. This allows our method to be applied to a variety of target distributions including those arising from non-conjugate model specifications [Kingma and Welling, 2013, Ranganath et al., 2014, Salimans et al., 2013]. We demonstrate empirically that our algorithm improves posterior estimates over other variational methods for several practical Bayesian models.

4.2 Variational Inference

Given a *target distribution* with density[1] $\pi(x)$ for a *continuous* random variable $x \in \mathcal{X} \subseteq \mathbb{R}^D$, variational inference approximates $\pi(x)$ with a tractable distribution, $q(x;\lambda)$, from which we can efficiently draw samples and form sample-based estimates of functions of x. Variational methods minimize the KL-divergence, $\text{KL}(q||\pi)$, between $q(\cdot;\lambda)$ and the true π as a function of *variational parameter* λ [Bishop, 2006]. Although direct minimization of $\text{KL}(q||\pi)$ is often intractable, we can derive a tractable objective based on properties of the KL-divergence. This objective is known as the *evidence lower bound* (ELBO):

$$\begin{aligned}\mathcal{L}(\lambda) &= \mathbb{E}_{q_\lambda}\left[\ln \pi(x) - \ln q(x;\lambda)\right] + \ln \mathcal{C} \\ &= \ln \mathcal{C} - \text{KL}(q_\lambda||\pi) \leq \ln \mathcal{C} = \ln \int \tilde{\pi}(x)dx\end{aligned}$$

which, due to the positivity of $\text{KL}(q||\pi)$, is a lower bound on $\mathcal{C} = \log \pi(x)$, i.e., the marginal likelihood.

Variational methods typically fix a family of distributions $Q = \{q(\cdot;\lambda) : \lambda \in \Lambda\}$ pa-

[1] We assume $\pi(x)$ is known up to a constant, $\tilde{\pi}(x) = \mathcal{C}\pi(x)$.

rameterized by λ, and *maximize* the ELBO with respect to $\lambda \in \Lambda$. Often there exists some (possibly non-unique) $\lambda^* \in \Lambda$ for which $\text{KL}(q||\pi)$ is minimized. However, when the family Q does not include π then $\text{KL}(q_{\lambda^*}||\pi) > 0$ which will result in biased estimates of functions $f(x)$, $\mathbb{E}_{x \sim q_{\lambda^*}}[f(x)] \neq \mathbb{E}_{x \sim \pi}[f(x)]$. An example of this bias, in the form of underestimated marginal variances, is depicted in Figure 4.1.

The primary alternative to variational methods for approximate inference is Markov chain Monte Carlo (MCMC), which constructs a Markov chain such that the target distribution remains invariant. Expectations with respect to the target distribution can be calculated as an average with respect to these correlated samples. MCMC typically enjoys nice asymptotic properties; as the number of samples grows, MCMC samplers represent the true target distribution with increasing fidelity. However, rules for constructing correct Markov steps are restrictive. With a few exceptions, most MCMC algorithms require evaluating a log-likelihood that touches all data at each step in the chain [Maclaurin and Adams, 2014, Welling and Teh, 2011]. This becomes problematic during statistical analyses of large amounts of data — MCMC is often considered unusable because of this computational bottleneck. Notably, variational methods can avoid this bottleneck by sub-sampling the data [Hoffman et al., 2013], as unbiased estimates of the *log-likelihood* can often be straight-forwardly used with optimization methods. As variational methods recast inference as optimization, data sub-sampling can often make already efficient approximation algorithms even more efficient.

In the next section, we propose an algorithm that iteratively grows the approximating class Q and reframes the VI procedure as a series of optimization problems, resulting in a practical inference method that can both represent arbitrarily complex distributions and scale to large data sets.

4.3 Variational Boosting

We define our class of approximating distributions to be mixtures of C simpler component distributions:

$$q^{(C)}(x;\lambda,\rho) = \sum_{c=1}^{C} \rho_c q_c(x;\lambda_c)\,,\ \text{s.t. } \rho_c \geq 0, \sum_c \rho_c = 1,$$

where we denote the full mixture as $q^{(C)}$, mixing proportions $\rho = (\rho_1, \dots, \rho_C)$, and component distributions $q_c(\cdot;\lambda_c)$ parameterized by $\lambda = (\lambda_1, \dots, \lambda_C)$. The component $q_c(\cdot;\lambda_c)$ can be any distribution over $\mathcal{X} \subseteq \mathbb{R}^D$ from which we can efficiently draw samples using a continuous mapping parameterized by λ_c (e.g., multivariate normal [Jaakkola and Jordan, 1998], or a composition of invertible maps [Rezende and Mohamed, 2015]).

When posterior expectations and variances are of interest, mixture distributions provide tractable summaries. Expectations are easily expressed in terms of component expectations:

$$\mathbb{E}_{q^{(C)}}\left[f(x)\right] = \int q^{(C)}(x) f(x) dx = \sum_c \rho_c \mathbb{E}_{q_c}\left[f(x)\right].$$

In the case of multivariate normal components, the mean and covariance of a mixture are easy to compute, as are marginal distributions along any set of dimensions.

Variational boosting (`vboost`) begins with a single mixture component, $q^{(1)}(x;\lambda) = q_1(x;\lambda_1)$ with $C = 1$. We fix $\rho_1 = 1$ and use existing black-box variational inference methods to fit the first component parameter, λ_1. At the next iteration $C = 2$, we fix λ_1 and introduce a new component into the mixture, $q_2(x;\lambda_2)$. We define a new ELBO objective as a function of new component parameters, λ_2, and a new mixture weight, ρ_2. We then optimize this objective with respect to λ_2 and ρ_2 until convergence. At each subsequent round, c, we introduce new component parameters and a mixing weight, (λ_c, ρ_c), which are then optimized according to a new ELBO objective. The name *variational boosting* is inspired by methods that iteratively construct strong learners from ensembles of weak learners. We apply `vboost` to target distributions via black-box variational inference with the *re-parameterization trick* to fit each component and mixture weights [Kingma and Welling, 2013, Ranganath et al., 2014, Salimans et al., 2013]. However, using mixtures as the variational approximation complicates the use of the re-parameterization trick.

4.3.1 The re-parameterization trick and mixtures

The re-parameterization trick is used to compute an unbiased estimate of the gradient of an objective that is expressed as an intractable expectation with respect to a continuous-valued random variable. This situation arises in variational inference when the ELBO cannot be evaluated analytically. We form an unbiased estimate as:

$$\begin{aligned}\mathcal{L}(\lambda) &= \mathbb{E}_q \left[\ln \pi(x) - \ln q(x;\lambda) \right] & (4.1)\\ &\approx \frac{1}{L} \sum_{\ell=1}^{L} \left[\ln \pi(x^{(\ell)}) - \ln q(x^{(\ell)}; \lambda) \right] & (4.2)\end{aligned}$$

where $x^{(\ell)} \sim q(x;\lambda)$. To obtain a Monte Carlo estimate of the gradient of $\mathcal{L}(\lambda)$ using the re-parameterization trick, we first separate the randomness needed to generate $x^{(\ell)}$ from the parameters λ, by defining a deterministic map $x^{(\ell)} \triangleq f_q(\epsilon;\lambda)$ such that $\epsilon \sim p(\epsilon)$ implies $x^{(\ell)} \sim q(x;\lambda)$. Note that $p(\epsilon)$ does not depend on λ. We then differentiate Eq. (4.2) with respect to λ through the map f_q to obtain an estimate of $\nabla_\lambda \mathcal{L}(\lambda)$.

When $q(\cdot;\lambda)$ is a mixture, applying the re-parameterization trick is not straightforward. The typical sampling procedure for a mixture model includes a discrete random variable that indicates a mixture component, which complicates differentiation. We circumvent this by re-writing the variational objective as a weighted combination of expectations with respect to individual mixture components:

$$\begin{aligned}
\mathcal{L}(\lambda, \rho) &= \int \left(\sum_{c=1}^{C} \rho_c q_c(x; \lambda_c) \right) [\ln \pi(x) - \ln q(x; \lambda)] \, dx \\
&= \sum_{c=1}^{C} \rho_c \int q_c(x; \lambda_c) [\ln \pi(x) - \ln q(x; \lambda)] \, dx \\
&= \sum_{c=1}^{C} \rho_c \mathbb{E}_{q_c} [\ln \pi(x) - \ln q(x; \lambda)]
\end{aligned}$$

which is a weighted sum of component-specific ELBOs. If the q_c are continuous and there exists some function $f_c(\epsilon; \lambda)$ such that $x = f_c(\epsilon; \lambda)$ and $x \sim q_c(\cdot; \lambda)$ when $\epsilon \sim p(\epsilon)$, then we can apply the re-parameterization trick to each component to obtain gradients of the ELBO:

$$\begin{aligned}
\nabla_{\lambda_c} \mathcal{L}(\lambda, \rho) &= \nabla_{\lambda_c} \sum_{c=1}^{C} \rho_c \mathbb{E}_{x \sim q(x;\lambda)} [\ln \pi(x) - \ln q(x; \lambda)] \\
&= \sum_{c=1}^{C} \rho_c \mathbb{E}_{\epsilon \sim p(\epsilon)} \left[\nabla_{\lambda_c} \ln \pi(f_c(\epsilon; \lambda_c)) - \nabla_{\lambda_c} \ln q(f_c(\epsilon; \lambda_c); \lambda) \right] .
\end{aligned}$$

This reformulation of $\nabla_{\lambda_c} \mathcal{L}(\lambda, \rho)$ enables the use of the re-parameterization trick in a component-by-component manner. The overall gradient is then a weighted combination of these component-specific gradients.

4.3.2 Incorporating New Components

In this section we present practical details of `vboost`. We first describe how to fit a single component and then the process for incorporating a new component into an existing mixture distribution.

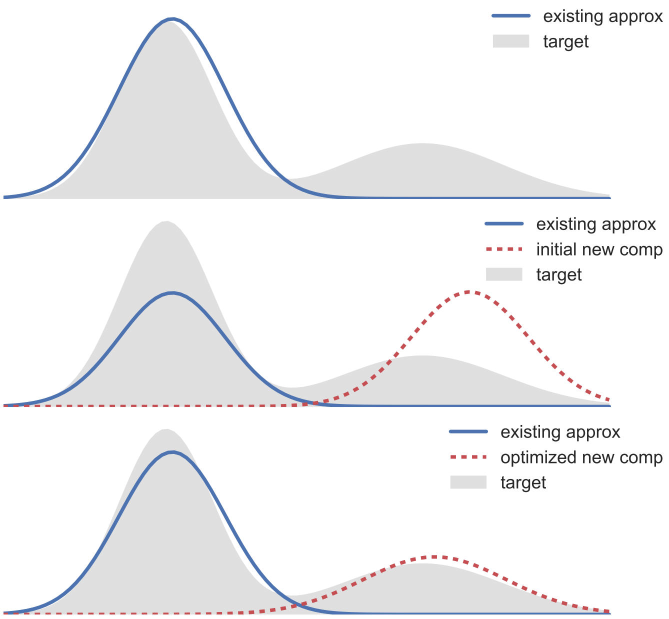


Figure 4.2: One-dimensional illustration of the `vboost` procedure. *Top*: Initial single-component approximation (solid blue). *Middle*: A new component (dotted red) is initialized. *Bottom*: New component parameters and mixing weights are optimized using Monte Carlo gradients of the ELBO. Note that the mass of the existing components can rise and fall, but not shift in space.

The first component `vboost`first fits an approximation to $\pi(x)$ consisting of a single component, q_1. We do this by maximizing the first ELBO objective

$$\mathcal{L}^{(1)}(\lambda_1) = \mathbb{E}_{q_1}\left[\ln \pi(x) - \ln q_1(x;\lambda_1)\right] \tag{4.3}$$

$$\lambda_1^* = \arg\max_{\lambda_1} \mathcal{L}^{(1)}(\lambda_1)\,. \tag{4.4}$$

Depending on the forms of π and q_1, optimizing $\mathcal{L}^{(1)}$ can be accomplished by various methods—an obvious choice being black-box VI with the re-parameterization trick. After convergence we fix λ_1 to be λ_1^*.

Component $C+1$ After iteration C, our current approximation to $\pi(x)$ is a mixture distribution with C components:

$$q^{(C)}(x;\lambda,\rho) = \sum_{c=1}^{C} \rho_c q_c(x;\lambda_c). \tag{4.5}$$

Adding a component to Eq. (4.5) introduces a new component parameter, λ_{C+1}, and a new mixing weight, ρ_{C+1}. In this section, the mixing parameter $\rho_{C+1} \in [0,1]$ mixes between the new component, $q_{C+1}(\cdot;\lambda_{C+1})$ and the existing approximation, $q^{(C)}$. The new approximate distribution is

$$\begin{aligned}
&q^{(C+1)}(x;\lambda,\rho) \\
&= (1-\rho_{C+1})q^{(C)}(x) + \rho_{C+1}q_{C+1}(x;\lambda_{C+1})\,.
\end{aligned}$$

The new ELBO, as a function of ρ_{C+1} and λ_{C+1}, is:

$$\begin{aligned}
&\mathcal{L}^{(C+1)}(\rho_{C+1},\lambda_{C+1}) \\
&= \mathbb{E}_{x\sim q^{(C+1)}}\left[\ln \pi(x) - \ln q^{(C+1)}(x;\lambda_{C+1},\rho_{C+1})\right] \\
&= (1-\rho_{C+1})\mathbb{E}_{q^{(C)}}\left[\ln \pi(x) - \ln q^{(C+1)}(x;\lambda_{C+1},\rho_{C+1})\right] \\
&\quad + \rho_{C+1}\mathbb{E}_{q_{C+1}}\left[\ln \pi(x) - \ln q^{(C+1)}(x;\lambda_{C+1},\rho_{C+1})\right]\,.
\end{aligned}$$

Crucially, we have separated out two expectations: one with respect to the existing approximation, $q^{(C)}$ (which is fixed), and the other with respect to the new component distribution, q_{C+1}. Because we have fixed $q^{(C)}$, we only need to optimize the new component parameters, λ_{C+1} and ρ_{C+1}, allowing us to use the re-parameterization trick to obtain gradients of $\mathcal{L}^{(C+1)}$. Note that evaluating the gradient requires sampling from the existing components which may result in larger variance than typical black-box variational methods. To mitigate the extra variance we use many samples to estimate the gradient and leave variance reduction to future work.

Figure 4.2 illustrates the algorithm on a simple one-dimensional example — the initialization of a new component and the resulting mixture after optimizing the second objective, $\mathcal{L}^{(2)}(\rho_2, \lambda_2)$. Figure 4.3 depicts the result of `vboost` on a two-dimensional, multi-modal target distribution. In both cases, the component distributions are Gaussians with diagonal covariance.

4.3.3 Structured Multivariate Normal Components

Though our method can use any component distribution that can be sampled using a continuous mapping, a sensible choice of component distribution is a multivariate normal

$$\begin{aligned} q(x;\lambda) &= \mathcal{N}(x;\mu_\lambda,\Sigma_\lambda) \\ &= |2\pi\Sigma_\lambda|^{-1/2} \exp\left(-\tfrac{1}{2}(x-\mu_\lambda)^\intercal \Sigma_\lambda^{-1}(x-\mu_\lambda)\right) \end{aligned}$$

where the variational parameter λ is transformed into a mean vector μ_λ and covariance matrix Σ_λ.

Specifying the structure of the covariance matrix is a choice that largely depends on the dimensionality of $\mathcal{X} \subseteq \mathbb{R}^D$ and the correlation structure of the target distribution. A common choice of covariance is a diagonal matrix, $\Sigma_\lambda = \mathtt{diag}(\sigma_1^2, \dots, \sigma_D^2)$, which implies that x is independent across dimensions. When the approximation only consists of one component, this structure is commonly referred to as the *mean field* family. While computationally efficient, mean field approximations cannot model posterior correlations, which often leads to underestimation of marginal variances. Additionally, when diagonal covariances are used as the component distributions in Eq. (4.5) the resulting mixture may require a large number of components to represent the strong correlations (see Fig. 4.3). Furthermore, independence constraints can actually introduce local optima in the variational objective [Wainwright and Jordan, 2008].

On the other end of the spectrum, we can parameterize the entire covariance matrix using the Cholesky decomposition, L, such that $LL^\intercal = \Sigma$. This allows Σ to be any positive semi-definite matrix, enabling q to have the full flexibility of a D-dimensional multivariate normal distribution. However, this introduces $D(D+1)/2$ parameters, which can be computationally cumbersome when D is even moderately large. Furthermore, only a few pairs of variables may exhibit posterior correlations, particularly in multi-level models or neural networks where different parameter types may be nearly independent in the posterior.

As such, we would like to incorporate *some* capacity to capture correlations between dimensions of x without overparameterizing the approximation. The next subsection discusses a covariance specification that provides this tradeoff, while remaining computationally tractable.

Low-rank plus diagonal covariance Black-box variational inference methods with the re-parameterization trick require sampling from the variational distribution and efficiently computing (or approximating) the entropy of the variational distribution. For multivariate normal distributions, the entropy is a function of the determinant of the covariance matrix, Σ, while computing the log likelihood requires computing Σ^{-1}. When the dimensionality of the target, D, is large, computing determinants and inverses will have $O(D^3)$ time complexity and therefore may be prohibitively expensive to compute at every iteration.

However, it may be unnecessary to represent all $D(D-1)/2$ possible correlations in the target distribution, particularly if certain dimensions are close to independent. One way to increase the capacity of $q(x;\lambda)$ is to model the covariance as a *low-rank plus*

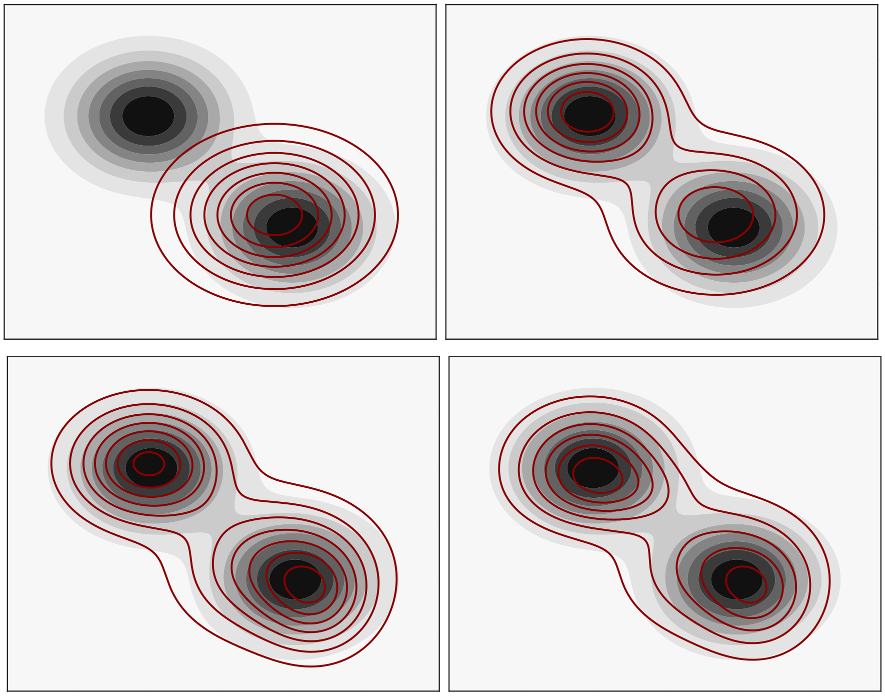


Figure 4.3: Sequence of increasingly complex approximate posteriors, with $C = 1, 2, 3, 4$ isotropic Gaussian components. The background (grey/black) contours depict the target distribution, and the foreground (red) contours depict the approximations.

diagonal (LR+D) matrix

$$\Sigma = FF^{\intercal} + \text{diag}(\exp(v)) \tag{4.6}$$

where $F \in \mathbb{R}^{D \times r}$ is a matrix of off diagonal factors, and $v \in \mathbb{R}^D$ is the log-diagonal component. This is effectively approximating the target via a *factor analysis* model.

The choice of the rank r presents a tradeoff: with a larger rank, the variational approximation can be more flexible; with a lower rank, the computations necessary for fitting the variational approximation are more efficient. As a concrete example, in Section 5.4 we present a $D = 37$ dimensional posterior resulting from a non-conjugate hierarchical model, and we show that a "rank $r = 2$ plus diagonal" covariance does an excellent job capturing all $D(D-1)/2 = 780$ pairwise correlations and D marginal variances. Incorporating more components using the `vboost` framework further improves the approximation of the distribution.

To use the re-parameterization trick with this low rank covariance, we can simulate from q in two steps

$$\begin{aligned} z^{(\text{lo})} &\sim \mathcal{N}(0, I_r) \qquad z^{(\text{hi})} \sim \mathcal{N}(0, I_D) \\ x &= F z^{(\text{lo})} + \mu + \mathcal{I}(v/2) z^{(\text{hi})} \end{aligned}$$

where $z^{(\text{lo})}$ generates the randomness due to the low-rank structure, and $z^{(\text{hi})}$ generates the randomness due to the diagonal structure. We define the function $\mathcal{I}(a) = \text{diag}(\exp(a))$ for notational simplicity. This sampling procedure is differentiable, enabling Monte Carlo estimation of the gradient with respect to F and v suitable for stochastic optimization.

In order to use LR+D covariance structure within `vboost`, we will need to efficiently compute the determinant and inverse of Σ. The matrix determinant lemma expresses the determinant of Σ as the product of two determinants

$$\begin{aligned} |FF^\intercal + \mathcal{I}(v))| &= |\mathcal{I}(v))||I_r + F^\intercal \mathcal{I}(\text{-}v) F| \\ &= \exp\left(\sum_d v_d\right) |I_r + F^\intercal \mathcal{I}(\text{-}v) F| \end{aligned}$$

where the left term is simply the product of the diagonal component, and the right term is the determinant of an $r \times r$ matrix, computable in $O(r^3)$ time [Harville, 1997].

To compute Σ^{-1}, the Woodbury matrix identity states that

$$\begin{aligned} &(FF^\intercal + \mathcal{I}(v))^{-1} \\ &= \mathcal{I}(\text{-}v) - \mathcal{I}(\text{-}v) F (I_r + F^\intercal \mathcal{I}(\text{-}v) F)^{-1} F^\intercal \mathcal{I}(\text{-}v) \end{aligned}$$

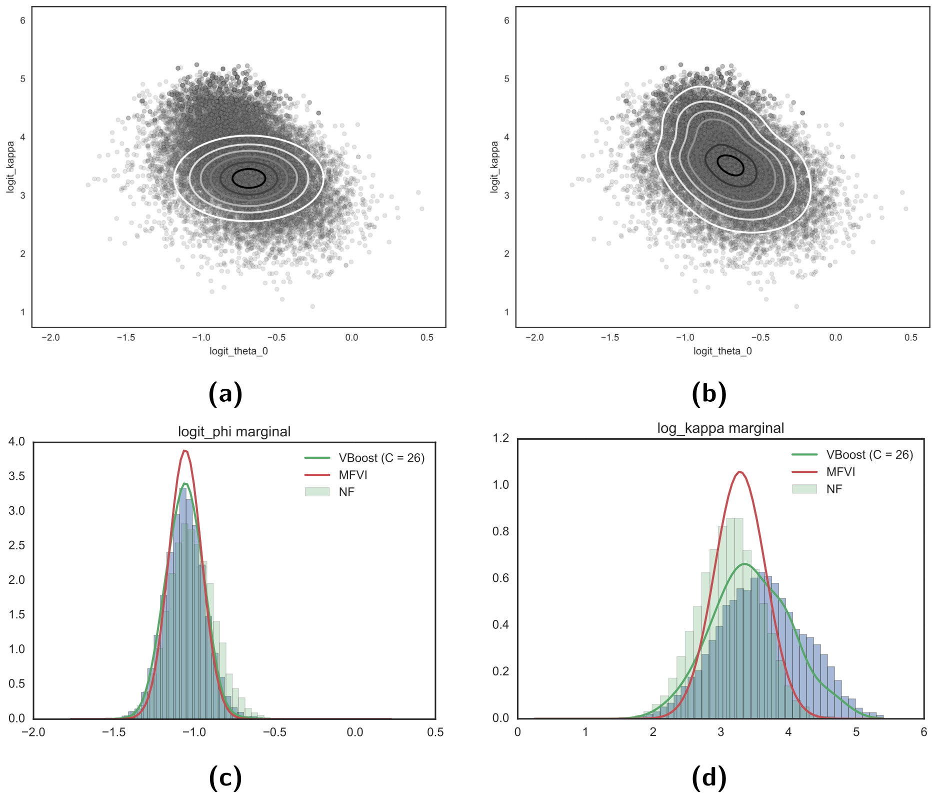
which involves the inversion of a smaller, $r \times r$ matrix and can be done in $O(r^3)$ time [Golub and Van Loan, 2013]. Importantly, for $r \ll D$ the above operations are efficiently differentiable and amenable for use in the BBVI framework.

Fitting the rank To specify the ELBO objective, we need to choose a rank r for the component covariance. There are many ways to decide on the rank of the variational approximation, some more appropriate for certain settings given dimensionality and computational constraints. For instance, we can greedily incorporate new rank components. Alternatively, we can fit a sequence of ranks $r = 1, 2, \dots, r_{\text{max}}$, and choose the best result (in terms of KL). In the Bayesian neural network model, we report results for a fixed schedule of ranks. In the hierarchical Poisson model, we monitor the change in marginal variances to decide the appropriate rank. See Section 4.A2 of the supplement for further discussion.

Initializing new component parameters When we add a new component, we must first initialize the component parameters. We find that the `vboost` optimization procedure can be sensitive to initialization, so we devise a cheap importance sampling-based algorithm to generate a good starting point. This initialization procedure is detailed in Section 4.A1 and Algorithm 1 in this chapter's Appendix.

4.3.4 Related Work

Mixtures of mean field approximations were introduced in Jaakkola and Jordan [1998] where mean field-like updates were developed using a bound on the entropy term and model-specific parameter updates. Nonparametric variational inference is a black-box variational inference algorithm that approximates a target distribution with a mixture of equally-weighted isotropic normals [Gershman et al., 2012]. This is accomplished by using a lower-bound on the entropy term in the ELBO to make the optimization procedure tractable. Similarly, Salimans et al. [2013] present a method for fitting mix-



(a) **(b)** **(c)** **(d)**

Figure 4.4: `baseball` marginals. Comparison of bivariate (top) and univariate (bottom) marginals for the `baseball` model. Histograms/scatterplots depict 20,000 NUTS samples. The top left depicts $(\ln \kappa, \theta_0)$ marginal samples and a mean field approximation (MFVI). The Top Right shows the same bivariate marginal, and the `vboost` approximation with isotropic components. The bottom panels compare NUTS, MFVI, and `vboost` on univariate marginals (ϕ and $\ln \kappa$).

ture distributions as an approximation. However, their method is restricted to mixture component distributions within the exponential family, and a joint optimization procedure. Mixture distributions are a type of hierarchical variational model [Ranganath et al., 2016], where the component identity can be thought of as latent variables in the variational distribution. While in Ranganath et al. [2016], the authors optimize a lower bound on the ELBO to fit general hierarchical variational models, our approach integrates out the discrete latent variables, allowing us to directly optimize the ELBO.

Sequential maximum-likelihood estimation of mixture models has been studied pre-

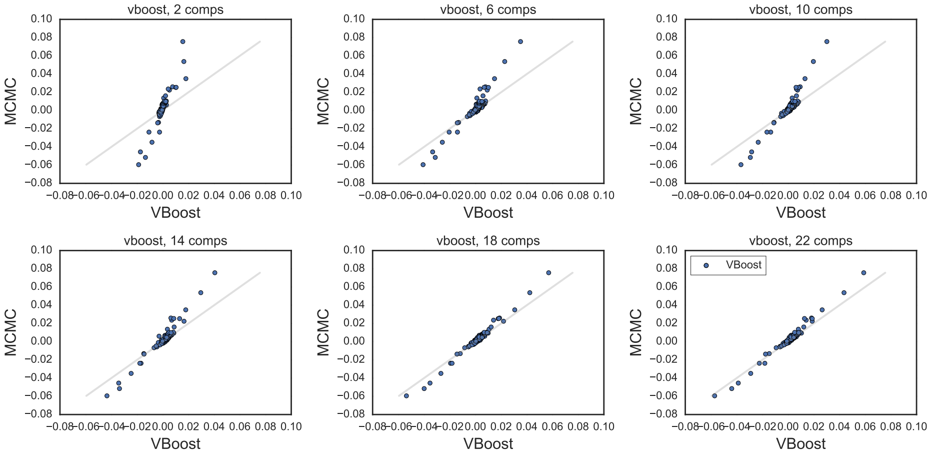


Figure 4.5: Comparison of posterior covariances for the $D = 20$-dimensional `baseball` model. Each plot compares covariance estimates of `vboost` (x-axis) with increasing numbers of components and MCMC samples (y-axis). As more components are added, the `vboost` estimates more closely match the MCMC covariance estimates.

viously where the error between the sequentially learned model and the optimal model where all components and weights are jointly learned is bounded by $O(1/C)$ where C is the number of mixture components [Li and Barron, 1999, Li, 1999, Rakhlin et al., 2006]. A similar bound was proven in Zhang [2003] using arguments from convex analysis. More recently, sequentially constructing a mixture of deep generative models has been shown to achieve the same $O(1/C)$ error bound when trained using an adversarial approach [Tolstikhin et al., 2016]. Though these ideas show promise for deriving error bounds for variational boosting, there are difficulties in applying them.

In concurrent work, Guo et al. [2016] developed a boosting procedure to construct flexible approximations to posterior distributions. In particular, they use gradient-boosting to determine candidate component distributions and then optimize the mixture weight for the new component [Friedman, 2000]. However, Guo et al. [2016] assume that the gradient-boosting procedure is able to find the optimal new component so that the arguments in Zhang [2003] apply, which is not true in general. We note that if we make the similar assumption that at each step of `vboost` the component parameters λ_C^* are found exactly, then the optimization of ρ_C is convex and can be optimized exactly. We can then appeal to the same arguments in Zhang [2003] and obtain an $O(1/C)$ error bound. The work in Guo et al. [2016] provides important first steps in the theoretical development of boosting methods applied to variational inference, however, we note that developing a comprehensive theory that deals with the difficulties of multimodality and the non-joint-convexity of KL divergence in λ and ρ is still needed. Recently, Moore [2016] began to address issues of multimodality from model symmetry in variational inference. However, the question remains whether the entire distribution is being explored.

Seeger [2010] explored the use of low-rank covariance Gaussians as variational approximations using a PCA-like algorithm. Additionally, concurrent work has proposed the use a LR+D matrices as the covariances of Gaussian posterior approximations [Ong et al., 2017]. We have also found that though the LR+D covariance approximation is useful for capturing posterior correlations, combining the idea with boosting new components to capture non-Gaussian posteriors yields superior posterior inferences.

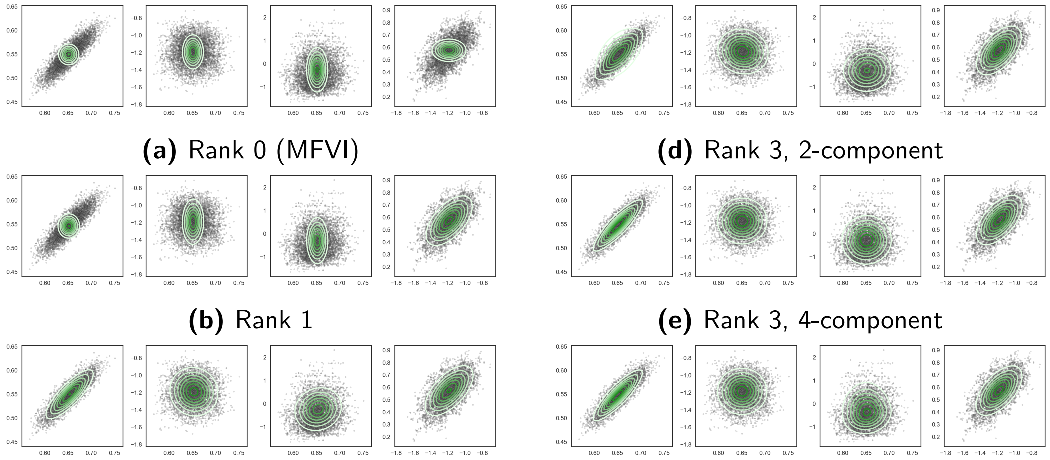
4.4 Experiments and Analysis

To supplement the previous synthetic examples, we use `vboost` to approximate various challenging posterior distributions arising from real statistical models of interest.[2]

Binomial Regression We first apply `vboost` to a non-conjugate hierarchical binomial regression model.[3] The model describes the binomial rates of success (batting averages) of baseball players using a hierarchical model [Efron and Morris, 1975], pa-

[2] Code available at `https://github.com/andymiller/vboost`.

[3] Model and data from the `mc-stan` case studies



(a) Rank 0 (MFVI)

(d) Rank 3, 2-component

(b) Rank 1

(e) Rank 3, 4-component

(c) Rank 2

(f) Rank 3, 8-component

Figure 4.6: *Left*: A sampling of bivariate marginals for a single Gaussian component approximation for the $D = 37$-dimensional `frisk` model. Each row incorporates more covariance structure. Though there are a total of 666 covariances to be approximated, only a few directions in the D-dimensional parameter space exhibit non-trivial correlations. *Right*: The same marginals with a mixture approximation using rank-3 Gaussians at various stages of the `vboost` algorithm. Introducing new mixture components allows the posterior to take a non-Gaussian shape, most exhibited in the third column.

rameterizing the "skill" of each player:

$$
\begin{aligned}
\theta_j &\sim \texttt{Beta}(\phi \cdot \kappa, (1-\phi)\cdot\kappa) && \text{player } j \text{ prior} \\
y_j &\sim \texttt{Binomial}(K_j, \theta_j) && \text{player } j \text{ hits} \enspace ,
\end{aligned}
$$

where y_j is the number of successes (hits) player j has attempted in K_j attempts (at bats). Each player has a latent success rate θ_j, which is governed by two global variables κ and ϕ. We specify the priors $\phi \sim \texttt{Unif}(0,1)$ and $\kappa \sim \texttt{Pareto}(1, 1.5)$. There are 18 players in this example, creating a posterior distribution with $D = 20$ parameters. For each round of `vboost`, we estimate $\nabla_{\lambda,\rho}\mathcal{L}^{(C+1)}$ using 400 samples each for q_{C+1} and q_C. We use 1,000 iterations of `adam` with default parameters to update ρ_{C+1} and λ_{C+1} [Kingma and Ba, 2014].

dataset	pbp	rank 5	vboost 2	vboost 6	vboost 10
wine	-0.990 (± 0.08)	-0.972 (± 0.05)	**-0.971** (± 0.05)	-0.978 (± 0.06)	-0.994 (± 0.06)
boston	-2.902 (± 0.64)	-2.670 (± 0.16)	-2.651 (± 0.16)	**-2.599** (± 0.16)	-2.628 (± 0.16)
concrete	-3.162 (± 0.15)	-3.247 (± 0.06)	-3.228 (± 0.06)	-3.169 (± 0.07)	**-3.134** (± 0.08)
power-plant	-2.798 (± 0.04)	-2.814 (± 0.03)	-2.811 (± 0.03)	-2.800 (± 0.03)	**-2.793** (± 0.03)
yacht	-0.990 (± 0.08)	-0.972 (± 0.05)	**-0.971** (± 0.05)	-0.978 (± 0.06)	-0.994 (± 0.06)
energy-eff.	**-1.971** (± 0.11)	-2.452 (± 0.12)	-2.422 (± 0.11)	-2.345 (± 0.11)	-2.299 (± 0.12)

Table 4.1: Test log probability for PBP and `vboost` with varying number of components (fixed rank of 5). Each entry shows the average predictive performance of the model and the standard deviation across the 20 trials — bold indicates the best average (though not necessarily "statistical significance").

In all experiments, we use `autograd` to obtain gradients with respect to new component parameters [Maclaurin et al., 2015b,a]. To highlight the fidelity of our method, we compare `vboost` with rank-1 components to mean field VI (MFVI) and the No-U-Turn Sampler (NUTS) [Hoffman and Gelman, 2014]. The empirical distribution resulting from 20k NUTS samples is considered the "ground truth" posterior in this example. Figure 4.4 compares a selection of univariate and bivariate posterior marginals. We see that `vboost` is able to closely match the NUTS posteriors, improving upon the MFVI approximation. Figure 4.5 compares the `vboost` covariance estimates to the "ground truth" estimates of MCMC at various stages of the algorithm. We see that `vboost` is able to capture pairwise covariances with increasing accuracy as the number of components increases.

Multi-level Poisson GLM We use `vboost` to approximate the posterior of a hierarchical Poisson GLM, a common non-conjugate Bayesian model. Here, we focus on a specific model that was formulated to measure the relative rates of stop-and-frisk events for different ethnicities and in different precincts [Gelman et al., 2007], and has been used as an illustrative example of multi-level modeling [Gelman and Hill, 2006]. The model uses a precinct and ethnicity effect to describe the relative rate of stop-and-frisk events

$$\begin{aligned}
\alpha_e &\sim \mathcal{N}(0, \sigma_\alpha^2) && \text{ethnicity effect} \\
\beta_p &\sim \mathcal{N}(0, \sigma_\beta^2) && \text{precinct effect} \\
\ln \lambda_{ep} &= \mu + \alpha_e + \beta_p + \ln N_{ep} && \text{log rate} \\
Y_{ep} &\sim \mathcal{P}(\lambda_{ep}) && \text{stop-and-frisk events}
\end{aligned}$$

where Y_{ep} are the number of stop-and-frisk events within ethnicity group e and precinct p over some fixed period of time; N_{ep} is the total number of arrests of ethnicity group e in precinct p over the same period of time; α_e and β_p are the ethnicity and precinct effects. The prior over the mean offset and group variances is given by $\mu, \ln \sigma_\alpha^2, \ln \sigma_\beta^2 \sim \mathcal{N}(0, 10^2)$.

As before, we simulate 20k NUTS samples, and compare various variational approximations. Because of the high posterior correlations present in this example, `vboost` with *diagonal* covariance components is inefficient in its representation of this structure. As such, this example relies on the *low-rank* approximation to shape the posterior. Figure 4.6 shows how posterior accuracy is affected by incorporating covariance structure (left) and adding more components (right).

Figures 4.A3.1 and 4.A3.2 compare `vboost` covariances to MCMC samples, showing that increased posterior rank capacity and number of components yield more accurate marginal variance and covariance estimates. These results indicate that while incorporating covariance structure increases the accuracy of estimating marginal variances, the non-Gaussianity afforded by the use of mixture components allows for a better posterior approximation translating into more accurate moment estimates.

Bayesian Neural Network We apply our method to a Bayesian neural network (BNN) regression model, which admits a high-dimensional, non-Gaussian posterior. We compare predictive performance of `vboost` to Probabilistic Backpropagation (PBP) [Hernández-

Lobato and Adams, 2015]. Mimicking the experimental setup of Hernández-Lobato and Adams [2015], we use a single 50-unit hidden layer, with ReLU activation functions. We place a normal prior over each weight in the neural network, governed by the same variance and an inverse Gamma prior over the observation variance yielding the model:

$$\begin{aligned} w_i &\sim \mathcal{N}(0, 1/\alpha) && \text{weights} \\ y|x, w, \tau &\sim \mathcal{N}(\phi(x, w), 1/\tau) && \text{output distribution} \end{aligned}$$

where $w = \{w_i\}$ is the set of weights, and $\phi(x, w)$ is a multi-layer perceptron that maps input x to output y as a function of parameters w. Both α and τ are given Gamma$(1, .1)$ priors. We denote the set of parameters as $\theta \triangleq (w, \alpha, \tau)$. We approximate the posterior $p(w, \alpha, \tau|\mathcal{D})$, where $\mathcal{D}$ is the training set of $\{x_n, y_n\}_{n=1}^N$ input-output pairs. We then use the posterior predictive distribution to compute the distribution for a new input x^*

$$p(y|x^*, \mathcal{D}) = \int p(y|x^*, \theta)p(\theta|\mathcal{D})d\theta \tag{4.7}$$

$$\approx \frac{1}{L}\sum_{\ell=1}^{L} p(y|x^*, \theta^{(\ell)}) \,, \quad \theta^{(\ell)} \sim p(\theta|\mathcal{D}) \tag{4.8}$$

and report average predictive log probabilities for held out data, $p(Y = y^*|x^*, \mathcal{D})$. For a dataset with input dimension P, the posterior has dimension $D = (P + 2) \cdot 50 + 3$ (between $D = 303$ and $D = 753$ for the data sets considered).

We report held-out predictive performance for different approximate posteriors for six datasets. For each dataset, we perform the following training procedure 20 times. First, we create a random partition into a 90% training set and 10% testing set. We then apply `vboost`, adding rank 5 components. We allow each additional component only 200 iterations. To save time on initialization, we draw 100 samples from the existing approximation, and initialize the new component with the sample with maximum weight. For comparison, Probabilistic back-propagation is given 1000 passes over the training data — empirically, sufficient for the algorithm to converge.

Table 4.A3.1 in the supplement presents out-of-sample log probability for single-component multivariate Gaussian approximations with varying rank structure. Table 4.1 presents out-of-sample log probability for additional rank 5 components added using `vboost`. We note that though we do not see much improvement as rank structure is added, we do see predictive improvement as components are added. Our results suggest that incorporating and adapting new mixture components is a recipe for a more expressive posterior approximation, translating into better predictive results. In fact, for all datasets we see that incorporating a new component improves test log probability, and we see further improvement with additional components for most of the datasets. Furthermore, in five of the datasets we see predictive performance surpass probabilistic back-propagation as new components are added. This highlights `vboost`'s ability to trade computation for improved accuracy. These empirical results suggest that augmenting a Gaussian approximation to include additional capacity can improve predictive performance in a BNN while retaining computational tractability.

4.4.1 Comparison to NPVI

We also compare `vboost` to nonparametric variational inference (NPVI) [Gershman et al., 2012], a similar mixture based black-box variational method. NPVI derives a tractable lower bound to the ELBO which is then approximately maximized. NPVI requires computing the Hessian of the model for the ELBO approximation, so we limit our comparison to the lower dimensional hierarchical models.

We also note that the NPVI ELBO approximation does not fully integrate the $\ln \pi(x)$ term against the variational approximation, $q(x; \lambda)$ when optimizing the mean parameters of the approximation components. When we applied NPVI to the `baseball` model, we discovered an instability in the optimization of these mean parameters (which we verified by finding that maximum a posteriori optimization diverges). Black box VI,

num comps	1	2	5	10	20
VBoost	-702.97	-700.92	-699.69	-699.07	-698.88
NPVI	-718.47	-717.86	-717.09	-716.36	-715.86

Table 4.2: ELBO values for vboost and NPVI (higher is better). Note that vboost with 1 component is MFVI. All ELBO values are computed using a Monte Carlo estimate with $L = 100k$ samples from the variational distribution. In NPVI, each component is a spherical gaussian with a single σ^2 shared across all dimensions — this limits the capacity of the approximation, requiring more components. Note, vboost greedily incorporates components, while NPVI is re-run using a different number of components.

vboost, and MCMC were not susceptible to this pathology. Consequently, we only compare NPVI to vboost on the frisk model. Because NPVI uses diagonal components, we restrict vboost to use purely diagonal components ($r = 0$). In Table 4.2 we show marginal likelihood lower bounds, comparing NPVI to vboost with a varying number of components. Even with a single component, the NPVI objective tends to underperform. The NPVI component variance is spherical, limiting its capacity to represent posterior correlations. Further, NPVI is approximately optimizing a looser lower bound to the marginal likelihood. These two factors explain why NPVI fails to match MFVI and vboost.

4.5 Discussion and Conclusion

We proposed vboost, a practical variational inference method that constructs an increasingly expressive posterior approximation and is applicable to a variety of Bayesian models. We demonstrated the ability of vboost to learn rich representations of complex, high-dimensional posteriors on a variety of real world statistical models. One avenue for future work is incorporating flexible component distributions such as compositions of invertible maps [Rezende and Mohamed, 2015] or auxiliary variable variational models [Maaløe et al., 2016]. We also plan to study approximation guarantees of the vboost method and variance reduction techniques for our reparameterization gradient approach. Also, when optimizing parameters of a variational family, recent work has shown that the natural gradient can be more robust and lead to better optima [Hoffman et al., 2013, Johnson et al., 2016]. Deriving and applying natural gradient updates for mixture approximations could make `vboost` more efficient.

Appendix

4.A1 Initializing Components

Introducing a new component requires initialization of component parameters. When our component distributions are mixtures of Gaussians, we found that the optimization procedure is sensitive to initialization. This section describes an importance-weighting scheme for initialization that produces (empirically) good initial values of component and mixing parameters.

Conceptually, a good initial component is located in a region of the target $\pi(x)$ that is underrepresented by the existing approximation $q^{(C)}$. A good initial weight is close to the proportion of mass in the unexplained region. Following this principle, we construct this component by first drawing importance-weighted samples from our existing approximation

$$x^{(\ell)} \sim q^{(C)}, \quad w^{(\ell)} = \frac{\pi(x^{(\ell)})}{q^{(C)}(x^{(\ell)})} \qquad \text{for } \ell = 1, \dots, L. \tag{4.9}$$

The samples with the largest weights $w^{(\ell)}$ tell us where regions of the target are poorly represented by our approximation. In fact, as L grows, and if $q^{(C)}$ is "close" enough to π, we can interpret $\{x^{(\ell)}, w^{(\ell)}\}$ as a weighted sample from π. Based on this interpretation, we can fit a mixture distribution (or *some components* of a mixture distribution) to this weighted sample using maximum likelihood, and recover a type of target approximation. For mixture distributions, an efficient inference procedure is Expectation-Maximization (EM) Dempster et al. [1977].

This approach, however, presents a few complications. First, we must adapt EM to fit a *weighted* sample. Second, importance weights can suffer from extremely high variance — one or two $w^{(\ell)}$ values may be extremely large compared to all other weights. This destabilizes our new component parameters and mixing weight, particularly the variance of the component. Intuitively, if a single weight $w^{(\ell)}$ is extremely large, this would correspond to many samples being located in a single location, and maximum likelihood with EM would want to shrink the variance of the new component to zero right on that location. To combat this behavior, we use a simple method to break up the big weights using a resampling and re-weighting step before applying weighted EM. Empirically, this improves our new component initializations and subsequent ELBO convergence.

Weighted EM Expectation-maximization is typically used to perform maximum likelihood in latent variable models. Mixture distributions are easily represented with latent variables — a sample's latent variable corresponds to the mixture component that produced it. EM starts with some initialization of model parameters (e.g.,component means, variances and mixing weights). The algorithm then iterates between two steps: 1) the *E-step*, which computes the distribution over the latent variables given the current setting of parameters, and 2) the *M-step*, which maximizes the *expected complete data log-likelihood* with respect to the distributions computed in the E-step.

We suppress details of the general treatment of EM, and focus on EM for mixture models as presented in Bishop [2006]. For mixture distributions, the E-step computes "responsibilities", or the probability that a datapoint came from one of the components. The M-step then computes a weighted maximum likelihood, where the log-likelihood of a datapoint for a particular component is weighted by the associated "responsibility". This weighted maximum likelihood is an easy entry-point for an additional set of weights — weights associated with each datapoint from the importance-weighting.

More concretely, for a sample of data, $x^{(\ell)}$, C mixture components, and current mixture component parameters and weights $\lambda = \{\rho_c, \lambda_c\}_{c=1}^{C}$, the E-step computes the following quantities

$$\gamma_c^{(\ell)} = p(z^{(\ell)} = c | x^{(\ell)}, \lambda) \tag{4.10}$$

$$\propto p(x^{(\ell)} | z^{(\ell), \lambda_c} = c) p(z^{(\ell)} = c) \tag{4.11}$$

where $\gamma_c^{(\ell)}$ is the "responsibility" of cluster c for datapoint ℓ. The M-step then computes component parameters by a weighted maximum likelihood

$$\lambda_c^* = \arg\max_{\lambda} \sum_{\ell=1}^{L} \gamma_c^{(\ell)} \cdot \ln p(x^{(\ell)} | z^{(\ell)} = c, \lambda_c) \,. \tag{4.12}$$

To incorporate importance weights $w^{(\ell)}$, we only need to slightly change the M-step:

$$\lambda_c^* = \arg\max_{\lambda} \sum_{\ell=1}^{L} w^{(\ell)} \cdot \gamma_c^{(\ell)} \cdot \ln p(x^{(\ell)} | z^{(\ell)} = c, \lambda_c) \,. \tag{4.13}$$

Because we are adding a new component, we would like our weighted EM routine to leave the remaining components unchanged. For instance, we want $\lambda_1, \ldots, \lambda_{C-1}$ to be fixed, while λ_C is free to explain the weighted sample. This can be accomplished in a straightforward manner by simply clamping the first $C - 1$ parameters during the M-step.

Resampling importance weights If our current approximation $q^{(C)}$ is sufficiently different in certain regions of the posterior, then some weights $w^{(\ell)}$ will end up being large compared to other weights. For instance, the objective KL$(q||p)$ tends to under-cover regions of the posterior, allowing $\pi(x)$ to be much larger than $q^{(c)}(x)$, meaning the weight associated with x will be large. This will create instability in the weighted EM approximation — likelihood maximization will want to put a zero-variance component on the single highest-weighted sample, which does not accurately reflect the local

Algorithm 1 Importance-weighted initialization of new components. This algorithm takes in the target distribution, $\pi(x)$, the current approximate distribution $q^{(C)}(x)$, and a number of samples L. This returns an initial value of new component parameters, λ_{C+1} and a new mixing weight ρ_{C+1}.

```
procedure InitComp($\pi, q^{(C)}, L$)
    $x^{(\ell)} \sim q^{(C)}$ for $\ell = 1, \ldots, L$                          ▷ sample from existing approx
    $w^{(\ell)} \leftarrow \frac{\pi(x^{(\ell)})}{q^{(C)}(x^{(\ell)})}$                                  ▷ set importance weights
    $\mathcal{O} \leftarrow$ outlier-weights($\{w^{(\ell)}\}$)
    $q^{(IW)} \leftarrow$ make-mixture($\mathcal{O}, \{w^{(\ell)}, x^{(\ell)}\}, q^{(C)}$)            ▷ break up big weights
    $x_r^{(\ell)} \sim q^{(IW)}$ for $\ell = 1, \ldots, L$                       ▷ sample from new mixture
    $w_r^{(\ell)} \leftarrow \frac{\pi(x_r^{(\ell)})}{q^{(IW)}(x^{(\ell)})}$                        ▷ re-sampled importance weights
    $\lambda_{C+1}, \rho_{C+1} \leftarrow$ weighted-em($\{x_r^{(\ell)}, w_r^{(\ell)}\}$)              ▷ fit new component
    return $\lambda_{C+1}, \rho_{C+1}$
```

curvature of $\pi(x)$. To combat this, we construct a slightly more complicated proposal distribution. Conceptually, we first create this naïve importance-weighted sample, and then find samples with outlier weights, and break those samples up. We do this by constructing a new proposal distribution that mixes the existing proposal, $q^{(C)}$, and component means located at the outlier samples. We define this proposal to be

$$q^{(IW)}(x) \propto p_0 q^{(C)}(x) + \sum_{\ell \in \mathcal{O}} w^{(\ell)} \mathcal{N}(x | x^{(\ell)}, \Sigma^{(\ell)}) \tag{4.14}$$

where $\ell \in \mathcal{O}$ denote the set of outlier samples from our original sample, and $p_0 = 1 - \sum_{\ell \in \mathcal{O}} w^{(\ell)}$ is the mass not placed on outlier samples. The variance of each outlier component, $\Sigma^{(\ell)}$ is set to some heuristic value — we typically use the diagonal of the covariance of $q^{(C)}$ as a good-enough guess.

We then create a new importance-weighted sample, using $q^{(IW)}$ and $\pi(x)$ just as we did before. By placing new components (with some non-zero variance) on the outlier samples, which are known to be in a region of high target probability and low approximate probability, we assume that there is more local probability around that region that needs to be explored. This allows us to inflate the local variance of the samples in this region — the region that weighted EM will place a component. Algorithm 1 unites the components from above sections into our final initialization procedure.

4.A2 Fitting the Rank

To specify the ELBO objective, we need to choose a rank r for the component covariance. There are a many ways to decide on the rank of the variational approximation, some more appropriate for certain settings given dimensionality and computation constraints. For instance, we can greedily incorporate new rank components. Alternatively, we can fit a sequence of components $r = 1, 2, \ldots, r_{max}$, and choose the best result (in terms of KL). In the Bayesian neural network model, we report results for a fixed schedule of ranks. In the hierarchical Poisson model, we monitor the change in marginal variances to decide the appropriate rank. In both cases, we require a stopping criterion. For a single Gaussian, one such criterion is the average change in marginal variances — if the marginal variation along each dimension remains the same from rank r to $r+1$, then the new covariance component is not incorporating explanatory power, particularly if marginal variances are of interest. As the $KL(q||\pi)$ objective tends to underestimate variances when restricted to a particular model class, we observe that the marginal variances grow as new covariance rank components are added. When fitting rank $r+1$, we can monitor the average absolute change in marginal variance (or standard deviation) as more covariance structure is incorporated. Figure 4.A2.1 in this supplement depicts this measurement for the $D = 37$-dimensional ‘frisk‘ posterior.

To justify sequentially adding ranks to mixture components we consider the KL-divergence between a rank-r Gaussian approximation to a full covariance Gaussian, $\text{KL}(q_r||p)$, where $q_r(\theta) = \mathcal{N}(0, \mathcal{I}(v) + \sum_{l=1}^{r} f_k f_k^\intercal)$ and $p(\theta) = \mathcal{N}(0, \Sigma)$. For simplicity, we assume both distributions have zero mean. If the true posterior is non-Gaussian we will attempt to approximate the best full-rank Gaussian with a low-rank Gaussian thus suffering an unrepresentable KL-divergence between the family of Gaussians and the true posterior. We also assume that the diagonal component, $\mathcal{I}(v)$, and the first $r-1$ columns of $F=[f_1,\ldots,f_r]$ are held fixed. Then we have

$$\begin{aligned}
&\mathrm{KL}(q_r||p)\\
&=\frac{1}{2}\Bigg(\mathrm{tr}\left(\Sigma^{-1}\left(\mathcal{I}(v)+\sum_{l=1}^{r} f_l f_l^{\intercal}\right)\right)\\
&\quad -k+\log\det\Sigma\\
&\quad -\log\det\left(\mathcal{I}(v)+\sum_{l=1}^{r} f_l f_l^{\intercal}\right)\Bigg)
\end{aligned}$$

which we differentiate with respect to v_r, remove terms that do not depend on v_r, and set to zero, yielding

$$\begin{aligned}
&\frac{\partial}{\partial v_r}\mathrm{KL}(q_r||p)\\
&=\frac{1}{2}\left[\Sigma^{-1}v_r-\left(\mathcal{I}(v)+\sum_{l=1}^{r} f_l f_l^{\intercal}\right)^{-1}v_r\right]=0\\
\to \Sigma^{-1}v_r &=\left(\underbrace{\mathcal{I}(v)+\sum_{l=1}^{r-1} f_l f_l^{\intercal}}_{C}+f_r f_r^{\intercal}\right)^{-1}v_r\\
&=\left(C^{-1}-\frac{C^{-1}f_r f_r^{\intercal}C^{-1}}{1+f_r^{\intercal}C^{-1}f_r}\right)f_r.
\end{aligned}$$

We can thus determine the optimal f_r from the following equation

$$\left(\Sigma^{-1}-C^{-1}\right)f_r=\left(-\frac{C^{-1}f_r f_r^{\intercal}C^{-1}}{1+||f_r||_C^2}\right)f_r \tag{4.15}$$

where we have defined $f_r^{\intercal}C^{-1}f_r=||f_r||_C^2$. Eq. (4.15) is reminiscent of an eigenvalue

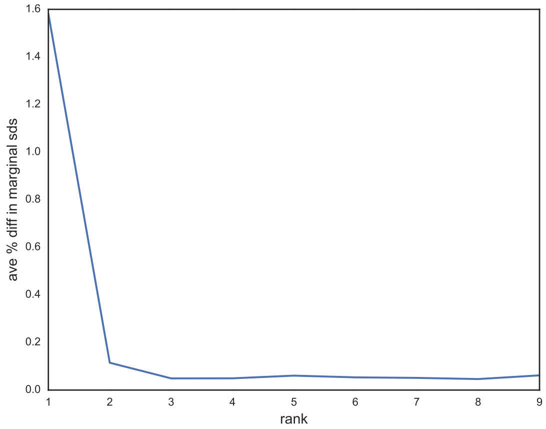


Figure 4.A2.1: Mean percent change in marginal variances for the Poisson GLM. After rank 5, the average percent change is less than 5%—this estimate is slightly noisy due to the stochastic optimization procedure.

problem indicating that the optimal solution for f_r should maximally explain $\Sigma^{-1} - C^{-1}$, i.e. the parameter space not already explained by $C = \mathcal{I}(v) + \sum_{l=1}^{r-1} f_l f_l^{\intercal}$. This provides justification for the previously proposed stopping criterion that monitors the increase in marginal variances since incorporating a new vector into the low-rank approximation should grow the marginal variances if extra correlations are captured. This is due to minimizing $\mathrm{KL}(q_r || p)$ which underestimates the variances when dependencies between parameters are broken.

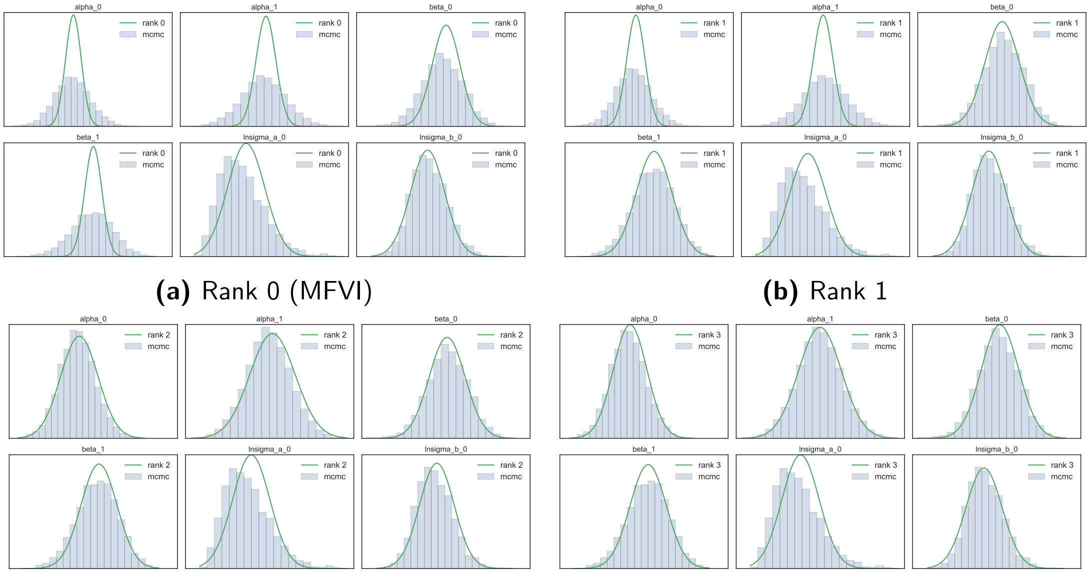
4.A3 Experiment Figures

4.A3.1 Frisk Model

Figures 4.A3.1 and 4.A3.2 depict `vboost` approximations of the 'frisk' model.

4.A3.2 Bayes Neural Network Results

Table 4.A3.1 depict out of sample log probability results for the Bayesian neural network as ranks vary.



(a) Rank 0 (MFVI)

(b) Rank 1

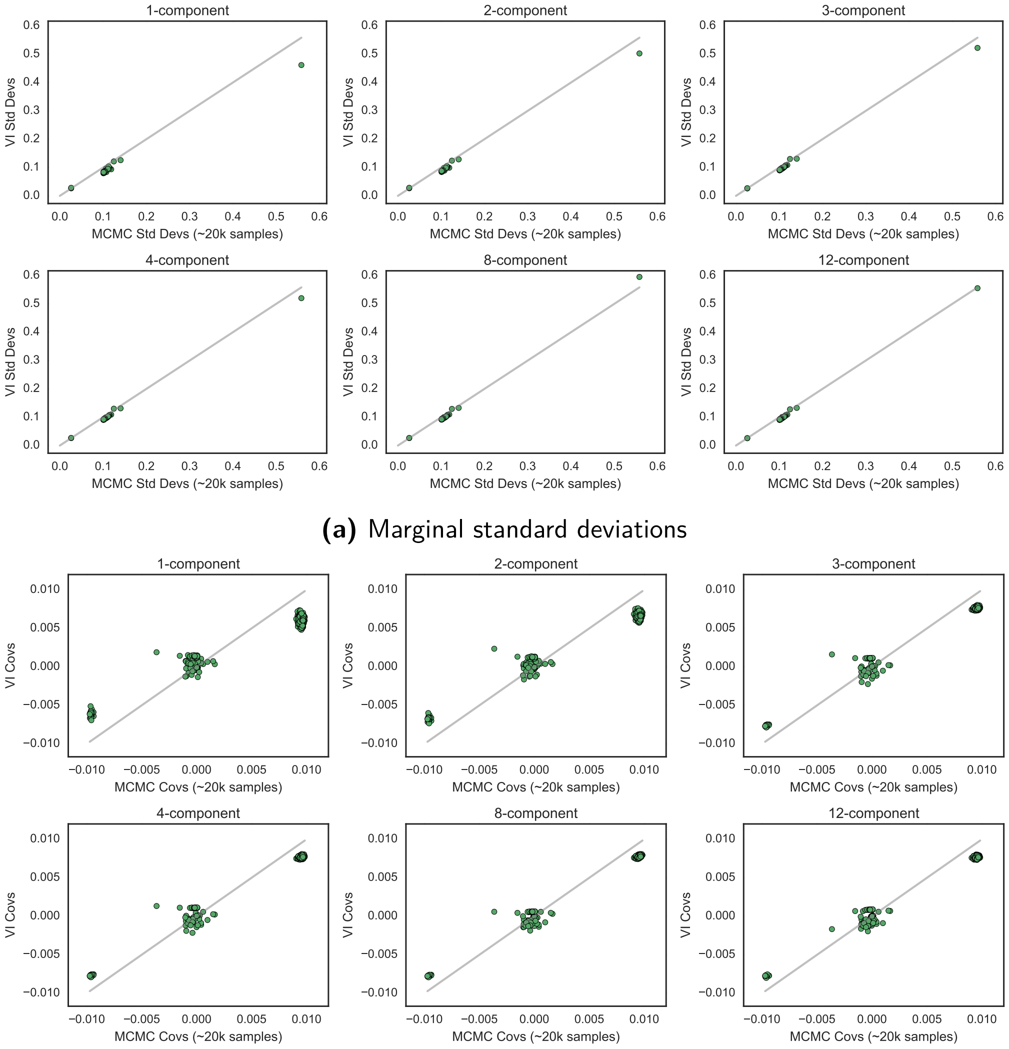
(c) Rank 2

(d) Rank 3

Figure 4.A3.1: Comparison of single Gaussian component marginals by rank for a 37-dimensional Poisson GLM posterior. The top left plot is a diagonal Gaussian approximation. The next plots show the how the marginal variances inflate as the covariance is allotted more capacity.

	pbp	mfvi	rank 5	rank 10	rank 15
wine	-0.990 (± 0.08)	-0.973 (± 0.05)	-0.972 (± 0.05)	**-0.972** (± 0.05)	-0.973 (± 0.05)
boston	-2.902 (± 0.64)	**-2.658** (± 0.18)	-2.670 (± 0.16)	-2.696 (± 0.14)	-2.743 (± 0.12)
concrete	**-3.162** (± 0.15)	-3.248 (± 0.07)	-3.247 (± 0.06)	-3.261 (± 0.06)	-3.286 (± 0.05)
power-plant	**-2.798** (± 0.04)	-2.812 (± 0.03)	-2.814 (± 0.03)	-2.838 (± 0.03)	-2.867 (± 0.02)
yacht	-0.990 (± 0.08)	-0.973 (± 0.05)	-0.972 (± 0.05)	**-0.972** (± 0.05)	-0.973 (± 0.05)
energy-efficiency	**-1.971** (± 0.11)	-2.451 (± 0.12)	-2.452 (± 0.12)	-2.469 (± 0.11)	-2.502 (± 0.09)

Table 4.A3.1: Comparison of test log probability for PBP Hernández-Lobato and Adams [2015] to Variational Inference with various ranks. Each entry shows the average predictive performance of the model on a specific dataset and the standard deviation across the 20 trials — bold indicates the best average (though not necessarily statistical significance).



(a) Marginal standard deviations

(b) Pairwise covariances

Figure 4.A3.2: A comparison of standard deviations and covariances for the `frisk` model. The MCMC-inferred values are along the horizontal axis, with the variational boosting values along the vertical axis. While the rank 3 plus diagonal covariance structure is able to account for most of the marginal variances, the largest one is still underestimated. Incorporating more rank 3 components allows the approximation to account for this variance. Similarly, the non-zero covariance measurements improve as more components are added.

5

Reducing Reparameterization Gradient Variance

Variational inference optimization objectives tend to be expectations—the objective we use in this thesis is the ELBO defined in 5.1, an expectation with respect to the variational approximation. The eventual goal of variational inference is to approximate expectations agains the *true posterior* distribution. These approximations are typically computed with samples from a surrogate distribution, the optimal variational approximation. This optimal approximation is the result of optimizing the ELBO, itself an expectation (albeit a more tractable one). In this way, a variational inference algorithm approximates a difficult expectation by computing a series of easy expectations—i.e. the ELBO and its gradient—throughout the optimization procedure.

However, for many interesting models, the ELBO itself is an intractable expectation. To do variational inference in this setting, we can either optimize a surrogate objective as in Gershman et al. [2012], or we can use a noisy estimate of the ELBO and its gradient within an optimization setting [Paisley et al., 2012, Ranganath et al., 2014]. Two commonly used estimators are defined in Section 2.7: the *score function estimator* (Equation 2.79) and the *reparameterization gradient estimator* (Equation 2.85). In this chapter, we focus on improving the efficiency of the reparameterization gradient estimator.

Ideally our gradient-based optimization algorithms would have access to the true gradient of the ELBO objective—this would eliminate noise in the optimization path and result in more efficient optimization. However, when this true gradient is intractable to compute, a low-variance Monte Carlo estimate can be used; the lower variance the better. We can always incorporate more independent samples to reduce the variance of the estimator, but this comes at a computational cost. Can we do better than the standard Monte Carlo estimator?

In this chapter, we present a method that exploits the structure of both the variational distribution and the smoothness of the log-posterior density. We show that we can compute gradient estimates that are orders of magnitude lower-variance at a modest computational cost. We empirically show that these gradient estimators can lead to faster, more robust optimization of variational parameters. The material from this chapter was previously published as Miller et al. [2017a].

5.1 Introduction

Representing massive datasets with flexible probabilistic models has been central to the success of many statistics and machine learning applications, but the computational burden of fitting these models is a major hurdle. For optimization-based fitting methods, a central approach to this problem has been replacing expensive evaluations of the gradient of the objective function with cheap, unbiased, stochastic estimates of the gradient. For example, stochastic gradient descent using small mini-batches of (conditionally) i.i.d. data to estimate the gradient at each iteration is a popular approach with massive data sets. Alternatively, some learning methods sample directly from a generative model or approximating distribution to estimate the gradients of interest, for example, in learning algorithms for implicit models [Mohamed and Lakshminarayanan, 2016, Tran et al., 2017] and generative adversarial networks [Arjovsky et al., 2017, Goodfellow et al., 2014].

Approximate Bayesian inference using variational techniques (variational inference, or VI) has also motivated the development of new stochastic gradient estimators, as the variational approach reframes the integration problem of inference as an optimization problem [Blei et al., 2017a]. VI approaches seek out the distribution from a well-understood variational family of distributions that best approximates an intractable posterior distribution. The VI objective function itself is often intractable, but recent work has shown that it can be optimized with stochastic gradient methods that use Monte Carlo estimates of the gradient [Paisley et al., 2012, Kingma and Welling, 2013, Ranganath et al., 2014, Rezende et al., 2014], which we call Monte Carlo variational inference (MCVI). In MCVI, generating samples from an approximate posterior distribution is the source of gradient stochasticity. Alternatively, *stochastic variational inference* (SVI) [Hoffman et al., 2013] and other stochastic optimization procedures induce stochasticity through data subsampling; MCVI can also be augmented with data subsampling to accelerate computation for large data sets.

The two commonly used MCVI gradient estimators are the *score function gradient* [Paisley et al., 2012, Ranganath et al., 2014] and the *reparameterization gradient* [Kingma and Welling, 2013, Rezende et al., 2014, Titsias and Lázaro-Gredilla, 2014, Glasserman, 2013]. Broadly speaking, score function estimates can be applied to both discrete and continuous variables, but often have high variance and thus are frequently used in conjunction with variance reduction techniques. On the other hand, the reparameterization gradient often has lower variance, but is restricted to continuous random variables. See Ruiz et al. [2016] for a unifying perspective on these two estimators. Like other stochastic gradient methods, the success of MCVI depends on controlling the variance of the stochastic gradient estimator.

In this work, we present a novel approach to controlling the variance of the *reparameterization gradient estimator* in MCVI. Existing MCVI methods control this variance naïvely by averaging several gradient estimates, which becomes expensive for large data sets and complex models, with error that only diminishes as $O(1/\sqrt{N})$. Our approach exploits the fact that, in MCVI, the randomness in the gradient estimator is completely determined by a known Monte Carlo generating process; this allows us to leverage knowledge about this generative procedure to de-noise the gradient estimator. In particular, we construct a computationally cheap control variate based on an analytical linear approximation to the gradient estimator. Taking a linear combination of a naïve gradient estimate with this control variate yields a new estimator for the gradient that remains unbiased but has lower variance. Applying the idea to Gaussian approximating families, we observe a 20-2,000× reduction in variance of the gradient norm under various conditions, and faster convergence and more stable behavior of optimization traces.

5.2 Background

Variational Inference Recapitulating some of Section 2.7, we consider models of the form $p(\boldsymbol{z}, \mathcal{D}) = p(\mathcal{D}|\boldsymbol{z})p(\boldsymbol{z})$ for data $\mathcal{D}$ with parameters and latent variables $\boldsymbol{z}$. VI approximates the posterior distribution $p(\boldsymbol{z}|\mathcal{D})$ with one from a simpler family, $\mathcal{Q} = \{q(\boldsymbol{z}; \boldsymbol{\lambda}), \boldsymbol{\lambda} \in \boldsymbol{\Lambda}\}$, parameterized by *variational parameters* $\boldsymbol{\lambda}$. The task is to find a setting of $\boldsymbol{\lambda}$ that makes $q(\boldsymbol{z}; \boldsymbol{\lambda})$ close to the posterior $p(\boldsymbol{z}|\mathcal{D})$ in KL divergence. Directly computing the KL divergence requires evaluating the posterior itself; therefore, VI procedures use the *evidence* *lower bound* (ELBO) as the optimization objective

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\lambda}}} \left[\ln p(\boldsymbol{z}, \mathcal{D}) - \ln q(\boldsymbol{z}; \boldsymbol{\lambda}) \right], \tag{5.1}$$

which, when maximized, minimizes the KL divergence between $q(\boldsymbol{z}; \boldsymbol{\lambda})$ and $p(\boldsymbol{z}|\mathcal{D})$. In special cases, parts of the ELBO can be expressed analytically (e.g. the entropy form or KL-to-prior form [Hoffman and Johnson, 2016]) — we focus on the general form in Equation 5.1.

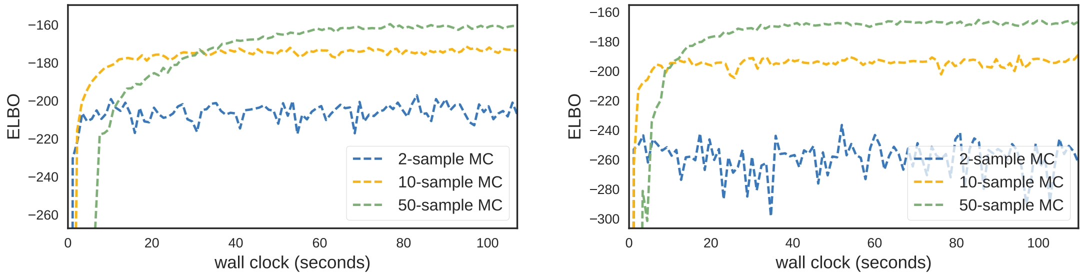
To maximize the ELBO with gradient methods, we need to compute the gradient of Eq. (5.1), $\partial \mathcal{L} / \partial \boldsymbol{\lambda} \triangleq \boldsymbol{g}_{\boldsymbol{\lambda}}$. The gradient inherits the ELBO's form as an expectation, which is in general an intractable quantity to compute. In this work, we focus on *reparameterization gradient estimators* (RGEs) computed using the *reparameterization trick*. The reparameterization trick exploits the structure of the *variational data generating procedure* — the mechanism by which $\boldsymbol{z}$ is simulated from $q_{\boldsymbol{\lambda}}(\boldsymbol{z})$. To compute the RGE, we first express the sampling procedure from $q_{\boldsymbol{\lambda}}(\boldsymbol{z})$ as a differentiable map applied to exogenous randomness

$$\epsilon \sim q_0(\epsilon) \qquad \text{independent of } \boldsymbol{\lambda} \tag{5.2}$$

$$\boldsymbol{z} = \mathcal{T}(\epsilon; \boldsymbol{\lambda}) \qquad \text{differentiable map,} \tag{5.3}$$

where the initial distribution q_0 and $\mathcal{T}$ are jointly defined such that $\boldsymbol{z} \sim q(\boldsymbol{z}; \boldsymbol{\lambda})$ has the desired distribution. As a simple concrete example, if we set $q(\boldsymbol{z}; \boldsymbol{\lambda})$ to be a diagonal Gaussian, $\mathcal{N}(\boldsymbol{m}_{\boldsymbol{\lambda}}, \text{diag}(\boldsymbol{s}_{\boldsymbol{\lambda}}^2))$, with $\boldsymbol{\lambda} = [\boldsymbol{m}_{\boldsymbol{\lambda}}, \boldsymbol{s}_{\boldsymbol{\lambda}}]$, $\boldsymbol{m}_{\boldsymbol{\lambda}} \in \mathbb{R}^D$, and $\boldsymbol{s}_{\boldsymbol{\lambda}} \in \mathbb{R}_+^D$ the mean and variance. The sampling procedure could then be defined as

$$\epsilon \sim \mathcal{N}(0, I_D), \qquad \boldsymbol{z} = \mathcal{T}(\epsilon; \boldsymbol{\lambda}) = \boldsymbol{m}_{\boldsymbol{\lambda}} + \boldsymbol{s}_{\boldsymbol{\lambda}} \odot \epsilon, \tag{5.4}$$



(a) step size = .01

(b) step size = .1

Figure 5.1: Optimization traces for MCVI applied to a Bayesian neural network with various hyperparameter settings. Each trace is running `adam` [Kingma and Ba, 2014]. The three lines in each plot correspond to three different numbers of samples, L, used to estimate the gradient at each step. (Left) small stepsize; (Right) stepsize 10 times larger. Large step sizes allow for quicker progress, however noisier (i.e., small L) gradients combined with large step sizes result in chaotic optimization dynamics. The converging traces reach different ELBOs due to the illustrative constant learning rates; in practice, one decreases the step size over time to satisfy the convergence criteria in Robbins and Monro [1951].

where $\boldsymbol{s} \odot \epsilon$ denotes an element-wise product.[1] Given this map, the reparameterization gradient estimator is simply the gradient of a Monte Carlo ELBO estimate with respect to $\boldsymbol{\lambda}$. For a single sample, this is

$$\epsilon \sim q_0(\epsilon)\,, \qquad \hat{\boldsymbol{g}}_{\boldsymbol{\lambda}} \triangleq \nabla_{\boldsymbol{\lambda}} \left[\ln p(\mathcal{T}(\epsilon;\boldsymbol{\lambda}), \mathcal{D}) - \ln q(\mathcal{T}(\epsilon;\boldsymbol{\lambda});\boldsymbol{\lambda})\right]$$

and similarly the L-sample approximation can be computed by averaging the single-sample estimator over the individual samples

$$\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}^{(L)} = \frac{1}{L}\sum_{\ell=1}^{L} \hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}(\epsilon^{\ell}). \tag{5.5}$$

Crucially, the reparameterization gradient is unbiased, $\mathbb{E}[\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}] = \nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda})$, guaranteeing the convergence of stochastic gradient optimization procedures that use it [Robbins and Monro, 1951].

[1] We will also use x/y and x^2 to denote pointwise division and squaring, respectively.

Gradient Variance and Convergence The efficiency of Monte Carlo variational inference hinges on the magnitude of gradient noise and the step size chosen for the optimization procedure. When the gradient noise is large, smaller gradient steps must be taken to avoid unstable dynamics of the iterates. However, a smaller step size increases the number of iterations that must be performed to reach convergence.

We illustrate this trade-off in Figure 5.1, which shows realizations of an optimization procedure applied to a Bayesian neural network using reparameterization gradients. The posterior is over $D = 653$ parameters that we approximate with a diagonal Gaussian (see Appendix 5.A3.2). We compare the progress of the `adam` algorithm using various numbers of samples [Kingma and Ba, 2014], fixing the learning rate. The noise present in the single-sample estimator causes extremely slow convergence, whereas the lower noise 50-sample estimator quickly converges, albeit at 50 times the cost.

The upshot is that with low noise gradients we are able to safely take larger steps, enabling faster convergence to a local optimum. A natural question is, how can we reduce the variance of gradient estimates without introducing too much extra computation? Our approach is to use information about the variational model, $q(\cdot; \boldsymbol{\lambda})$, and carefully construct a control variate to the gradient.

Control Variates Control variates are random quantities that are used to reduce the variance of a statistical estimator without introducing any bias by incorporating additional information into the estimator [Glasserman, 2004]. Given an unbiased estimator $\hat{\boldsymbol{g}}$ such that $\mathbb{E}[\hat{\boldsymbol{g}}] = \boldsymbol{g}$ (the quantity of interest), our goal is to construct another unbiased estimator with lower variance. We can do this by defining a *control variate* $\tilde{\boldsymbol{g}}$ with *known expectation* $\tilde{\boldsymbol{m}}$ and can write the new estimator as

$$\boldsymbol{g}^{(cv)} = \hat{\boldsymbol{g}} - \boldsymbol{C}(\tilde{\boldsymbol{g}} - \tilde{\boldsymbol{m}})\,. \tag{5.6}$$

where $\boldsymbol{C} \in \mathbb{R}^{D \times D}$ for D-dimensional $\hat{\boldsymbol{g}}$. Clearly the new estimator has the same expectation as the original estimator, but has a different variance. We can attain optimal variance reduction by appropriately setting $\boldsymbol{C}$. Intuitively, the optimal $\boldsymbol{C}$ is very similar to a regression coefficient — it is related to the *covariance* between the control variate and the original estimator. See Appendix 5.A1 for further details on optimally setting $\boldsymbol{C}$.

5.3 Method: Modeling Reparameterization Gradients

In this section we develop our main contribution, a new gradient estimator that can dramatically reduce reparameterization gradient variance. In MCVI, the reparameterization gradient estimator (RGE) is a Monte Carlo estimator of the true gradient — the estimator itself is a random variable. This random variable is generated using the "reparameterization trick" — we first generate some randomness ϵ and then compute the gradient of the ELBO with respect to $\boldsymbol{\lambda}$ holding ϵ fixed. This results in a complex distribution from which we can generate samples, but in general cannot characterize due to the complexity of the term arising from the gradient of the model term.

However, we do have a lot of information about the sampling procedure — we know the variational distribution $\ln q(\boldsymbol{z}; \boldsymbol{\lambda})$, the transformation $\mathcal{T}$, and we can evaluate the model joint density $\ln p(\boldsymbol{z}, \mathcal{D})$ pointwise. Furthermore, with automatic differentiation, it is often straightforward to obtain gradients and Hessian-vector products of our model $\ln p(\boldsymbol{z}, \mathcal{D})$. We propose a scheme that uses the structure of $q_{\boldsymbol{\lambda}}$ and curvature of $\ln p(\boldsymbol{z}, \mathcal{D})$ to construct a tractable approximation of the distribution of the RGE.[2] This approximation has a known mean and is correlated with the RGE distribution, allowing us to use it as a control variate to reduce the RGE variance.

Given a variational family parameterized by $\boldsymbol{\lambda}$, we can decompose the ELBO gradient

[2] We require the model $\ln p(\boldsymbol{z}, \mathcal{D})$ to be twice differentiable.

into a few terms that reveal its "data generating procedure"

$$\epsilon \sim q_0 \,, \quad \boldsymbol{z} = \mathcal{T}(\epsilon; \boldsymbol{\lambda}) \tag{5.7}$$

$$\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}} \triangleq \hat{\boldsymbol{g}}(\boldsymbol{z}; \boldsymbol{\lambda}) = \underbrace{\frac{\partial \ln p(\boldsymbol{z}, \mathcal{D})}{\partial \boldsymbol{z}}}_{\textit{data term}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\lambda}} - \underbrace{\frac{\partial \ln q_{\boldsymbol{\lambda}}(\boldsymbol{z})}{\partial \boldsymbol{z}}}_{\textit{pathwise score}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\lambda}} - \underbrace{\frac{\partial \ln q_{\boldsymbol{\lambda}}(\boldsymbol{z})}{\partial \boldsymbol{\lambda}}}_{\textit{parameter score}} \,. \tag{5.8}$$

Certain terms in Eq. (5.8) have tractable distributions. The Jacobian of $\mathcal{T}(\cdot; \boldsymbol{\lambda})$, given by $\partial \boldsymbol{z} / \partial \boldsymbol{\lambda}$, is defined by our choice of $q(\boldsymbol{z}; \boldsymbol{\lambda})$. For some transformations $\mathcal{T}$ we can exactly compute the distribution of the Jacobian given the distribution of ϵ. The *pathwise* and *parameter score* terms are gradients of our approximate distribution with respect to $\boldsymbol{\lambda}$ (via $\boldsymbol{z}$ or directly). If our approximation is tractable (e.g., a multivariate Gaussian), we can exactly characterize the distribution for these components.[3]

However, the *data term* in Eq. (5.8) involves a potentially complicated function of the latent variable $\boldsymbol{z}$ (and therefore a complicated function of ϵ), resulting in a difficult-to-characterize distribution. Our goal is to construct an approximation to the distribution of $\partial \ln p(\boldsymbol{z}, \mathcal{D}) / \partial \boldsymbol{z}$ and its interaction with $\partial \boldsymbol{z} / \partial \boldsymbol{\lambda}$ given a fixed distribution over ϵ. If the approximation yields random variables that are highly correlated with $\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}$, then we can use it to reduce the variance of that RGE sample.

Linearizing the data term To simplify notation, we write the data term of the gradient as

$$\boldsymbol{f}(\boldsymbol{z}') \triangleq \left. \frac{\partial \ln p(\boldsymbol{z}, \mathcal{D})}{\partial \boldsymbol{z}} \right|_{\boldsymbol{z} = \boldsymbol{z}'} \quad , \tag{5.9}$$

[3] In fact, we know that the expectation of the *parameter score* term is zero, and removing that term altogether can sometimes be a source of variance reduction that we do not explore here [Roeder et al., 2017].

where $\boldsymbol{f} : \mathbb{R}^D \mapsto \mathbb{R}^D$ since $\boldsymbol{z} \in \mathbb{R}^D$. We then linearize $\boldsymbol{f}$ about some value $\boldsymbol{z}_0$

$$\tilde{\boldsymbol{f}}(\boldsymbol{z}) = \boldsymbol{f}(\boldsymbol{z}_0) + \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}(\boldsymbol{z}_0)\right](\boldsymbol{z} - \boldsymbol{z}_0) = \boldsymbol{f}(\boldsymbol{z}_0) + \boldsymbol{H}(\boldsymbol{z}_0)(\boldsymbol{z} - \boldsymbol{z}_0), \tag{5.10}$$

where $\boldsymbol{H}(\boldsymbol{z}_0)$ is the Hessian of the model, $\ln p(\boldsymbol{z}, \mathcal{D})$, with respect to $\boldsymbol{z}$ evaluated at $\boldsymbol{z}_0$,

$$\boldsymbol{H}(\boldsymbol{z}_0) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}(\boldsymbol{z}_0) = \frac{\partial^2 \ln p(\boldsymbol{z}, \mathcal{D})}{\partial \boldsymbol{z}^2}(\boldsymbol{z}_0) \tag{5.11}$$

Note that even though this uses second-order information about the model, it is a first-order approximation of the gradient. We also view this as a transformation of the random ϵ for a fixed $\boldsymbol{\lambda}$

$$\tilde{\boldsymbol{f}}_{\boldsymbol{\lambda}}(\epsilon) = \boldsymbol{f}(\boldsymbol{z}_0) + \boldsymbol{H}(\boldsymbol{z}_0)(\mathcal{T}(\epsilon, \boldsymbol{\lambda}) - \boldsymbol{z}_0)\,, \tag{5.12}$$

which is linear in $\boldsymbol{z} = \mathcal{T}(\epsilon, \boldsymbol{\lambda})$. For some forms of $\mathcal{T}$ we can analytically derive the distribution of the random variable $\tilde{\boldsymbol{f}}_{\boldsymbol{\lambda}}(\epsilon)$. In Eq. (5.8), the *data term* interacts with the Jacobian of $\mathcal{T}$, given by

$$\boldsymbol{J}_{\boldsymbol{\lambda}'}(\epsilon) \triangleq \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\lambda}} = \frac{\partial \mathcal{T}(\epsilon, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}}\Big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}'}\,, \tag{5.13}$$

which importantly is a function of the same ϵ as in Eq. (5.12). We form our approximation of the first term in Eq. (5.8) by multiplying Eqs. (5.12) and (5.13) yielding

$$\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}}^{(data)}(\epsilon) \triangleq \tilde{\boldsymbol{f}}_{\boldsymbol{\lambda}}(\epsilon)\boldsymbol{J}_{\boldsymbol{\lambda}}(\epsilon)\,. \tag{5.14}$$

The tractability of this approximation hinges on how Eq. (5.14) depends on ϵ. When $q(\boldsymbol{z}; \boldsymbol{\lambda})$ is multivariate normal, we show that this approximation has a computable mean and can be used to reduce variance in MCVI settings. In the following sections we describe and empirically test this variance reduction technique applied to diagonal Gaussian posterior approximations.

5.3.1 Gaussian Variational Families

Perhaps the most common choice of approximating distribution for MCVI is a diagonal Gaussian, parameterized by a mean $\boldsymbol{m}_{\boldsymbol{\lambda}} \in \mathbb{R}^D$ and scales $\boldsymbol{s}_{\boldsymbol{\lambda}} \in \mathbb{R}^D_+$. [4] The log probability density function is

$$\ln q(\boldsymbol{z}; \boldsymbol{m}_{\boldsymbol{\lambda}}, \boldsymbol{s}^2_{\boldsymbol{\lambda}}) = -\frac{1}{2}(\boldsymbol{z} - \boldsymbol{m}_{\boldsymbol{\lambda}})^\intercal \left[\mathrm{diag}(\boldsymbol{s}^2_{\boldsymbol{\lambda}})\right]^{-1} (\boldsymbol{z} - \boldsymbol{m}_{\boldsymbol{\lambda}}) - \frac{1}{2}\sum_d \ln \boldsymbol{s}^2_{\boldsymbol{\lambda},d} - \frac{D}{2}\ln(2\pi)\,. \tag{5.15}$$

To generate a random variate $\boldsymbol{z}$ from this distribution, we use the sampling procedure in Eq. (5.4). We denote the Monte Carlo RGE as $\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}} \triangleq [\hat{\boldsymbol{g}}_{\boldsymbol{m}_{\boldsymbol{\lambda}}}, \hat{\boldsymbol{g}}_{\boldsymbol{s}_{\boldsymbol{\lambda}}}]$. From Eq. (5.15), it is straightforward to derive the distributions of the *pathwise score*, *parameter score*, and *Jacobian* terms in Eq. (5.8).

The *Jacobian* term of the sampling procedure has two straightforward components

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{m}_{\boldsymbol{\lambda}}} = I_D\,, \quad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{s}_{\boldsymbol{\lambda}}} = \mathrm{diag}(\epsilon)\,. \tag{5.16}$$

The *pathwise score* term is the partial derivative of Eq. (5.15) with respect to $\boldsymbol{z}$, ignoring variation due to the variational distribution parameters and noting that $\boldsymbol{z} = \boldsymbol{m}_{\boldsymbol{\lambda}} + \boldsymbol{s}_{\boldsymbol{\lambda}} \odot \epsilon$:

$$\frac{\partial \ln q}{\partial \boldsymbol{z}} = -\mathrm{diag}(\boldsymbol{s}^2_{\boldsymbol{\lambda}})^{-1}(\boldsymbol{z} - \boldsymbol{m}_{\boldsymbol{\lambda}}) = -\epsilon / \boldsymbol{s}_{\boldsymbol{\lambda}}\,. \tag{5.17}$$

The *parameter score* term is the partial derivative of Eq. (5.15) with respect to variational parameters $\boldsymbol{\lambda}$, ignoring variation due to $\boldsymbol{z}$. The $\boldsymbol{m}_{\boldsymbol{\lambda}}$ and $\boldsymbol{s}_{\boldsymbol{\lambda}}$ components are given

[4] For diagonal Gaussian q, we define $\boldsymbol{\lambda} = [\boldsymbol{m}_{\boldsymbol{\lambda}}, \boldsymbol{s}_{\boldsymbol{\lambda}}]$.

by

$$\frac{\partial \ln q}{\partial \boldsymbol{m}_{\boldsymbol{\lambda}}} = (\boldsymbol{z} - \boldsymbol{m}_{\boldsymbol{\lambda}})/\boldsymbol{s}_{\boldsymbol{\lambda}}^2 = \epsilon/\boldsymbol{s}_{\boldsymbol{\lambda}} \tag{5.18}$$

$$\frac{\partial \ln q}{\partial s_{\boldsymbol{\lambda}}} = -1/\boldsymbol{s}_{\boldsymbol{\lambda}} - (\boldsymbol{z} - m_{\boldsymbol{\lambda}})^2/\boldsymbol{s}_{\boldsymbol{\lambda}}^2 = \frac{\epsilon^2 - 1}{\boldsymbol{s}_{\boldsymbol{\lambda}}}. \tag{5.19}$$

The *data term*, $\boldsymbol{f}(\boldsymbol{z})$, multiplied by the Jacobian of $\mathcal{T}$ is all that remains to be approximated in Eq. (5.8). We linearize $\boldsymbol{f}$ around $\boldsymbol{z}_0 = \boldsymbol{m}_{\boldsymbol{\lambda}}$ where the approximation is expected to be accurate

$$\tilde{\boldsymbol{f}}_{\boldsymbol{\lambda}}(\epsilon) = \boldsymbol{f}(\boldsymbol{m}_{\boldsymbol{\lambda}}) + \boldsymbol{H}(\boldsymbol{m}_{\boldsymbol{\lambda}})\left((\boldsymbol{m}_{\boldsymbol{\lambda}} + \boldsymbol{s}_{\boldsymbol{\lambda}} \odot \epsilon) - \boldsymbol{m}_{\boldsymbol{\lambda}}\right) \tag{5.20}$$

$$\sim \mathcal{N}\left(\boldsymbol{f}(\boldsymbol{m}_{\boldsymbol{\lambda}}), \boldsymbol{H}(\boldsymbol{m}_{\boldsymbol{\lambda}})\text{diag}(\boldsymbol{s}_{\boldsymbol{\lambda}}^2)\boldsymbol{H}(\boldsymbol{m}_{\boldsymbol{\lambda}})^{\intercal}\right) . \tag{5.21}$$

Putting It Together: Full RGE Approximation We write the complete approximation of the RGE in Eq. (5.8) by combining Eqs. (5.16), (5.17), (5.18), (5.19), and (5.21) which results in two components that are concatenated, $\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}} = [\tilde{\boldsymbol{g}}_{\boldsymbol{m}_{\boldsymbol{\lambda}}}, \tilde{\boldsymbol{g}}_{\boldsymbol{s}_{\boldsymbol{\lambda}}}]$. Each component is defined as

$$\tilde{\boldsymbol{g}}_{\boldsymbol{m}_{\boldsymbol{\lambda}}} = \tilde{\boldsymbol{f}}_{\boldsymbol{\lambda}}(\epsilon) + \epsilon/\boldsymbol{s}_{\boldsymbol{\lambda}} - \epsilon/\boldsymbol{s}_{\boldsymbol{\lambda}} \tag{5.22}$$

$$= \boldsymbol{f}(\boldsymbol{m}_{\boldsymbol{\lambda}}) + \boldsymbol{H}(\boldsymbol{m}_{\boldsymbol{\lambda}})(\boldsymbol{s}_{\boldsymbol{\lambda}} \odot \epsilon) \tag{5.23}$$

$$\tilde{\boldsymbol{g}}_{\boldsymbol{s}_{\boldsymbol{\lambda}}} = \tilde{\boldsymbol{f}}_{\boldsymbol{\lambda}}(\epsilon) \odot \epsilon + (\epsilon/\boldsymbol{s}_{\boldsymbol{\lambda}}) \odot \epsilon - \frac{\epsilon^2 - 1}{\boldsymbol{s}_{\boldsymbol{\lambda}}} \tag{5.24}$$

$$= (\boldsymbol{f}(\boldsymbol{m}_{\boldsymbol{\lambda}}) + \boldsymbol{H}(\boldsymbol{m}_{\boldsymbol{\lambda}})(\boldsymbol{s}_{\boldsymbol{\lambda}} \odot \epsilon)) \odot \epsilon + \frac{1}{\boldsymbol{s}_{\boldsymbol{\lambda}}} . \tag{5.25}$$

To summarize, we have constructed an approximation, $\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}}$, of the reparameterization gradient, $\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}$, as a function of ϵ. Because both $\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}$ are functions of the same random variable ϵ, and because we have mimicked the random process that generates true gradient samples, the two gradient estimators will be correlated. This approximation yields two tractable distributions — a Gaussian for the mean parameter gradient, $\boldsymbol{g}_{\boldsymbol{m}_{\boldsymbol{\lambda}}}$,

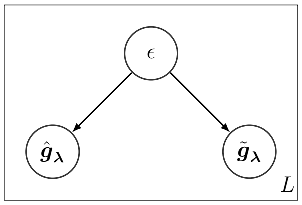

ϵ
$\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}$
$\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}}$
L

Figure 5.1: Relationship between the base randomness ϵ, RGE $\hat{\boldsymbol{g}}$, and approximation $\tilde{\boldsymbol{g}}$. Arrows indicate deterministic functions. Sharing ϵ correlates the random variables. We know the distribution of $\tilde{\boldsymbol{g}}$, which allows us to use it as a control variate for $\hat{\boldsymbol{g}}$.

Algorithm 2 Gradient descent with RV-RGE with a diagonal Gaussian variational family

```
procedure RV-RGE-Optimize($\boldsymbol{\lambda}_1, \ln p(\boldsymbol{z}, \mathcal{D}), L$)
    $\boldsymbol{f}(\boldsymbol{z}) \leftarrow \nabla_{\boldsymbol{z}} \ln p(\boldsymbol{z}, \mathcal{D})$
    $\boldsymbol{H}(\boldsymbol{z}_a, \boldsymbol{z}_b) \leftarrow \left[\nabla^2_{\boldsymbol{z}} \ln p(\boldsymbol{z}_a, \mathcal{D})\right] \boldsymbol{z}_b$    ▷ Define Hessian-vector product function
    for $t = 1, \dots, T$ do
        $\epsilon^{(\ell)} \sim \mathcal{N}(0, I_D)$ for $\ell = 1, \dots, L$    ▷ Base randomness $q_0$
        $\hat{\boldsymbol{g}}^{(\ell)}_{\boldsymbol{\lambda}_t} \leftarrow \nabla_{\boldsymbol{\lambda}} \ln p(\boldsymbol{z}(\epsilon^{(\ell)}, \boldsymbol{\lambda}_t), \mathcal{D})$    ▷ Reparameterization gradients
        $\tilde{\boldsymbol{g}}^{(\ell)}_{\boldsymbol{m}_{\boldsymbol{\lambda}_t}} \leftarrow \boldsymbol{f}(\boldsymbol{m}_{\boldsymbol{\lambda}_t}) + \boldsymbol{H}(\boldsymbol{m}_{\boldsymbol{\lambda}_t}, \boldsymbol{s}_{\boldsymbol{\lambda}_t} \odot \epsilon^{(\ell)})$    ▷ Mean approx
        $\tilde{\boldsymbol{g}}^{(\ell)}_{\boldsymbol{s}_{\boldsymbol{\lambda}_t}} \leftarrow \left(\boldsymbol{f}(\boldsymbol{m}_{\boldsymbol{\lambda}_t}) + \boldsymbol{H}(\boldsymbol{m}_{\boldsymbol{\lambda}_t}, \boldsymbol{s}_{\boldsymbol{\lambda}_t} \odot \epsilon^{(\ell)})\right) \odot \epsilon + \frac{1}{\boldsymbol{s}_{\boldsymbol{\lambda}_t}}$    ▷ Scale approx
        $\mathbb{E}[\tilde{\boldsymbol{g}}_{\boldsymbol{m}_{\boldsymbol{\lambda}_t}}] \leftarrow \boldsymbol{f}(\boldsymbol{m}_{\boldsymbol{\lambda}_t})$    ▷ Mean approx expectation
        $\mathbb{E}[\tilde{\boldsymbol{g}}_{\boldsymbol{s}_{\boldsymbol{\lambda}_t}}] \leftarrow \mathrm{diag}(\boldsymbol{H}(\boldsymbol{m}_{\boldsymbol{\lambda}_t})) \odot \boldsymbol{s}_{\boldsymbol{\lambda}_t} + 1/\boldsymbol{s}_{\boldsymbol{\lambda}_t}$    ▷ Scale approx expectation
        $\hat{\boldsymbol{g}}^{(RV)}_{\boldsymbol{\lambda}_t} = \frac{1}{L} \sum_\ell \hat{\boldsymbol{g}}^\ell_{\boldsymbol{\lambda}_t} - (\tilde{\boldsymbol{g}}^\ell_{\boldsymbol{\lambda}_t} - \mathbb{E}[\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}_t}])$    ▷ Subtract control variate
        $\boldsymbol{\lambda}_{t+1} \leftarrow$ grad-update($\boldsymbol{\lambda}_t, \hat{\boldsymbol{g}}^{(RV)}_{\boldsymbol{\lambda}_t}$)    ▷ Gradient step (sgd, adam, etc.)
    return $\boldsymbol{\lambda}_T$
```

and a location shifted, scaled non-central χ^2 for the scale parameter gradient $\boldsymbol{g}_{\boldsymbol{s}_{\boldsymbol{\lambda}}}$. Importantly, we can compute the mean of each component

$$\mathbb{E}[\tilde{\boldsymbol{g}}_{\boldsymbol{m}_{\boldsymbol{\lambda}}}] = \boldsymbol{f}(\boldsymbol{m}_{\boldsymbol{\lambda}}) \,, \qquad \mathbb{E}[\tilde{\boldsymbol{g}}_{\boldsymbol{s}_{\boldsymbol{\lambda}}}] = \mathrm{diag}(\boldsymbol{H}(\boldsymbol{m}_{\boldsymbol{\lambda}})) \odot \boldsymbol{s}_{\boldsymbol{\lambda}} + 1/\boldsymbol{s}_{\boldsymbol{\lambda}} \,. \tag{5.26}$$

We use $\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}}$ (along with its expectation) as a control variate to reduce the variance of the RGE $\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}$.

5.3.2 Reduced Variance Reparameterization Gradient Estimators

Now that we have constructed a tractable gradient approximation, $\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}}$, with high correlation to the original reparameterization gradient estimator, $\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}$, we can use it as a control variate as in Eq. (5.6)

$$\hat{\boldsymbol{g}}^{(RV)}_{\boldsymbol{\lambda}} = \hat{\boldsymbol{g}}_{\boldsymbol{\lambda}} - \boldsymbol{C}(\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}} - \mathbb{E}[\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}}]). \tag{5.27}$$

The optimal value for $\boldsymbol{C}$ is related to the covariance between $\tilde{\boldsymbol{g}}_{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{g}}_{\boldsymbol{\lambda}}$ (see Appendix 5.A1). We can try to estimate the value of $\boldsymbol{C}$ (or a diagonal approximation to $\boldsymbol{C}$) on the fly, or we can simply fix this value. In our case, because we are using an accurate linear approximation to the transformation of a spherical Gaussian, the optimal value of $\boldsymbol{C}$ will be close to the identity (see Appendix 5.A1.1).

High Dimensional Models For models with high dimensional posteriors, direct manipulation of the Hessian is computationally intractable. However, our approximations in Eqs. (5.23) and (5.25) only require a Hessian-vector product, which can be computed nearly as efficiently as the gradient [Pearlmutter, 1994]. Modern automatic differentiation packages enable easy and efficient implementation of Hessian-vector products for nearly any differentiable model [Abadi et al., 2016, Paszke et al., 2017, Maclaurin et al., 2015b]. We note that the mean of the control variate $\tilde{\boldsymbol{g}}_{\boldsymbol{s}_{\boldsymbol{\lambda}}}$ (Eq. (5.26)), depends on the diagonal of the Hessian matrix. While computing the Hessian diagonal may be tractable in some cases, in general it may cost the time equivalent of D function evaluations to compute [Martens et al., 2012]. Given a high dimensional problem, we can avoid this bottleneck in multiple ways. The first is simply to ignore the random variation in the Jacobian term due to ϵ — if we fix $\boldsymbol{z}$ to be $\boldsymbol{m}_{\boldsymbol{\lambda}}$ (as we do with the data term), the portion of the Jacobian that corresponds to $\boldsymbol{s}_{\boldsymbol{\lambda}}$ will be zero (in Eq. (5.16)). This will result in the same Hessian-vector-product-based estimator for $\tilde{\boldsymbol{g}}_{\boldsymbol{m}_{\boldsymbol{\lambda}}}$ but will set $\tilde{\boldsymbol{g}}_{\boldsymbol{s}_{\boldsymbol{\lambda}}} = 0$, yielding variance reduction for the mean parameter but not the scale.

Alternatively, we can estimate the Hessian diagonal on the fly. If we use $L > 1$ samples at each iteration, we can create a per-sample estimate of the $\boldsymbol{s}_{\boldsymbol{\lambda}}$-scaled diagonal of the Hessian using the other samples [Bekas et al., 2007]. As the scaled diagonal estimator is unbiased, we can construct an unbiased estimate of the control variate mean to use in lieu of the actual mean. We will see that the resulting variance is not much higher than when using full Hessian information, and is computationally tractable to deploy on high-dimensional models. A similar *local baseline* strategy is used for variance reduction in Mnih and Rezende [2016].

RV-RGE Estimators We introduce three different estimators based on variations of the gradient approximation defined in Eqs. (5.23), (5.25), and (5.26), each adressing the Hessian operations differently:

- The *Full Hessian* estimator implements the three equations as written and can be used when it is computationally feasible to use the full Hessian.
- The *Hessian Diagonal* estimator replaces the Hessian in (5.23) with a diagonal approximation, useful for models with a cheap Hessian diagonal.
- The *Hessian-vector product + local approximation* (HVP+Local) uses an efficient Hessian-vector product in Eqs. (5.23) and (5.25), while approximating the diagonal term in Eq. (5.26) using a local baseline. The HVP+Local approximation is geared toward models where Hessian-vector products can be computed, but the exact diagonal of the Hessian cannot.

We detail the RV-RGE procedure in Algorithm 2 and compare properties of these three estimators to the pure Monte Carlo estimator in the following section.

5.3.3 Related Work

Recently, Roeder et al. [2017] introduced a variance reduction technique for reparameterization gradients that ignores the *parameter score* component of the gradient and can be viewed as a type of control variate for the gradient throughout the optimization procedure. This approach is complementary to our method — our approximation is typically more accurate near the beginning of the optimization procedure, whereas the estimator in Roeder et al. [2017] is low-variance near convergence. We hope to incorporate information from both control variates in future work. Per-sample estimators in a multi-sample setting for variational inference were used in Mnih and Rezende [2016]. We employ this technique in a different way; we use it to estimate computationally intractable quantities needed to keep the gradient estimator unbiased. Black box variational inference used control variates and Rao-Blackwellization to reduce the variance of score-function estimators [Ranganath et al., 2014]. Our development of variance reduction for reparameterization gradients complements their work. Other variance reduction techniques for stochastic gradient descent have focused on stochasticity due to data subsampling [Johnson and Zhang, 2013, Wang et al., 2013]. Johnson and Zhang [2013] cache statistics about the entire dataset at each epoch to use as a control variate for noisy mini-batch gradients.

The variance reduction method described in Paisley et al. [2012] is conceptually similar to ours. This method uses first or second order derivative information to reduce the variance of the score function estimator. The score function estimator (and their reduced variance version) often has much higher variance than the reparameterization gradient estimator that we improve upon in this work. Our variance measurement experiments in Table 5.1 includes a comparison to the estimator featured in [Paisley et al., 2012], which we found to be much higher variance than the baseline RGE.

5.4 Experiments and Analysis

In this section we empirically examine the variance properties of RV-RGEs and stochastic optimization for two real-data examples — a hierarchical Poisson GLM and a Bayesian neural network.[5]

- *Hierarchical Poisson GLM*: The `frisk` model is a hierarchical Poisson GLM, described in Appendix 5.A3.1. This non-conjugate model has a $D = 37$ dimensional posterior.
- *Bayesian Neural Network*: The non-conjugate `bnn` model is a Bayesian neural network applied to the `wine` dataset, (see Appendix 5.A3.2) and has a $D = 653$ dimensional posterior.

[5] Code available at https://github.com/andymiller/ReducedVarianceReparamGradients.

Quantifying Gradient Variance Reduction We measure the variance reduction of the RGE observed at various iterates, $\boldsymbol{\lambda}_t$, during execution of gradient descent. Both the gradient magnitude, and the marginal variance of the gradient elements — using a sample of 1000 gradients — are reported. Further, we inspect both the mean, $\boldsymbol{m}_{\boldsymbol{\lambda}}$, and log-scale, $\ln \boldsymbol{s}_{\boldsymbol{\lambda}}$, parameters separately. Table 5.1 compares gradient variances for the `frisk` model for our four estimators: i) pure Monte Carlo (MC), ii) Full Hessian, iii) Hessian Diagonal, and iv) Hessian-vector product + local approximation (HVP+Local). Additionally, we compare our methods to the estimator described in [Paisley et al., 2012], based on the score function estimator and a control variate method. We use a first order delta method approximation of the model term, which admits a closed form control variate term.

Each entry in the table measures the percent of the variance of the pure Monte Carlo estimator. We show the average variance over each component $\text{Ave}\mathbb{V}(\cdot)$, and the variance of the norm $\mathbb{V}(||\cdot||)$. We separate out variance in mean parameters, $\boldsymbol{g}_{\boldsymbol{m}}$, log scale parameters, $\ln \boldsymbol{g}_{\boldsymbol{s}}$, and the entire vector $\boldsymbol{g}_{\boldsymbol{\lambda}}$. The reduction in variance is dramatic. Using HVP+Local, in the norm of the mean parameters we see between a 80× and 3,000× reduction in variance depending on the progress of the optimizer. The importance of the full Hessian-vector product for reducing mean parameter variance is also demonstrated as the Hessian diagonal only reduces mean parameter variance by a factor of 2-5×.

For the variational scale parameters, $\ln \boldsymbol{g}_{\boldsymbol{s}}$, we see that early on the HVP+Local approximation is able to reduce parameter variance by a large factor (≈ 2,000×). However, at later iterates the HVP+Local scale parameter variance is on par with the Monte Carlo estimator, while the full Hessian estimator still enjoys huge variance reduction. This indicates that, by this point, most of the noise is the local Hessian diagonal estimator. We also note that in this problem, most of the estimator variance is in the mean parameters. Because of this, the norm of the entire parameter gradient, $\boldsymbol{g}_{\boldsymbol{\lambda}}$ is reduced

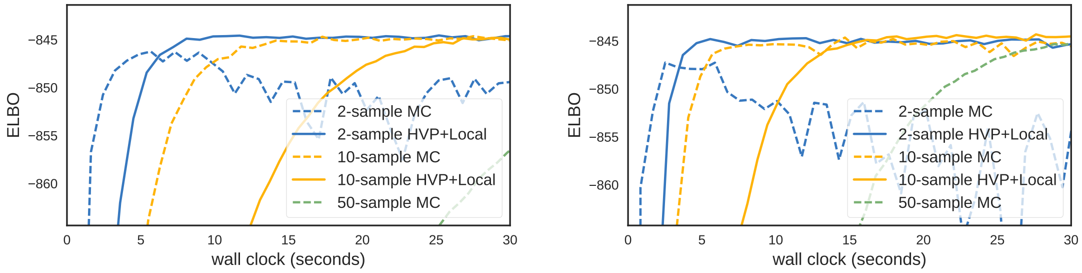
Table 5.1: Comparison of variances for RV-RGEs with $L = 10$-sample estimators. Variance measurements were taken for $\boldsymbol{\lambda}$ values at three points during the optimization algorithm (early, mid, late). The parenthetical rows labeled "MC abs" denote the absolute value of the standard Monte Carlo reparameterization gradient estimator. The other rows compare estimators relative to the pure MC RGE variance — a value of 100 indicates equal variation $L = 10$ samples, a value of 1 indicates a 100-fold decrease in variance (lower is better). Our new estimators (Full Hessian, Hessian Diag, HVP+Local) are described in Section 5.3.2. The Score Delta method is the gradient estimator described in [Paisley et al., 2012]. Additional variance measurement results are in Appendix 5.A4.

		$\boldsymbol{g}_{m_{\boldsymbol{\lambda}}}$		$\ln \boldsymbol{g}_{s_{\boldsymbol{\lambda}}}$		$\boldsymbol{g}_{\boldsymbol{\lambda}}$	
Iteration	Estimator	Ave $\mathbb{V}(\cdot)$	$\mathbb{V}(\lVert\cdot\rVert)$	Ave $\mathbb{V}(\cdot)$	$\mathbb{V}(\lVert\cdot\rVert)$	Ave $\mathbb{V}(\cdot)$	$\mathbb{V}(\lVert\cdot\rVert)$
early	(MC abs.)	(1.7e+02)	(5.4e+03)	(3e+04)	(2e+05)	(1.5e+04)	(5.9e+03)
	MC	100.000	100.000	100.000	100.000	100.000	100.000
	Full Hessian	1.279	1.139	0.001	0.002	0.008	1.039
	Hessian Diag	34.691	23.764	0.003	0.012	0.194	21.684
	HVP + Local	1.279	1.139	0.013	0.039	0.020	1.037
	Score Delta Paisley et al. [2012]	6069.668	718.430	1.395	0.931	34.703	655.105
mid	(MC abs.)	(3.8e+03)	(1.3e+05)	(18)	(3.3e+02)	(1.9e+03)	(1.3e+05)
	MC	100.000	100.000	100.000	100.000	100.000	100.000
	Full Hessian	0.075	0.068	0.113	0.143	0.076	0.068
	Hessian Diag	38.891	21.283	6.295	7.480	38.740	21.260
	HVP + Local	0.075	0.068	30.754	39.156	0.218	0.071
	Score Delta Paisley et al. [2012]	4763.246	523.175	2716.038	700.100	4753.752	523.532
late	(MC abs.)	(1.7e+03)	(1.3e+04)	(1.1)	(19)	(8.3e+02)	(1.3e+04)
	MC	100.000	100.000	100.000	100.000	100.000	100.000
	Full Hessian	0.042	0.030	1.686	0.431	0.043	0.030
	Hessian Diag	40.292	53.922	23.644	28.024	40.281	53.777
	HVP + Local	0.042	0.030	98.523	99.811	0.110	0.022
	Score Delta Paisley et al. [2012]	5183.885	1757.209	17355.120	3084.940	5192.270	1761.317

by $100 - 5{,}000\times$. We found that the score function estimator (with the delta method control variate) is typically much higher variance than the baseline reparameterization gradient estimator (often by a factor of 10-50×). In Appendix 5.A4 we report results for other values of L.

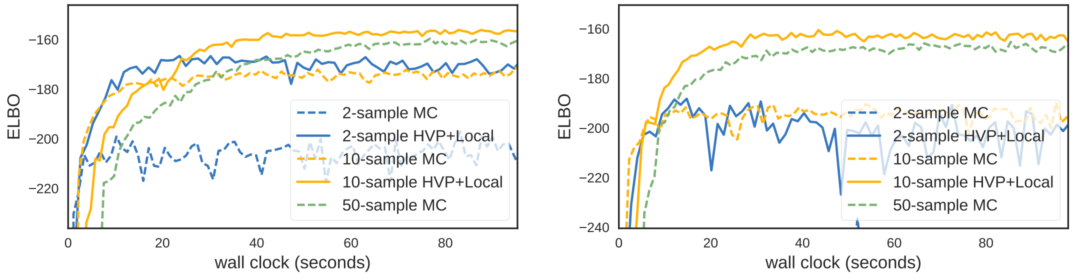
Optimizer Convergence and Stability We compare the optimization traces for the `frisk` and `bnn` model for the MC and the HVP+Local estimators under various conditions. At each iteration we estimate the true ELBO value using 2000 Monte Carlo samples. We optimize the ELBO objective using `adam` [Kingma and Ba, 2014] for two step sizes, each trace starting at the same value of $\boldsymbol{\lambda}_0$.

Figure 5.1 compares ELBO optimization traces for $L = 2$ and $L = 10$ samples and



(a) adam with step size $= 0.05$ **(b)** adam with step size $= .10$

Figure 5.1: MCVI optimization trace applied to the frisk model for two values of L and step size. We run the standard MC gradient estimator (solid line) and the RV-RGE with $L = 2$ and 10 samples.



(a) adam with step size $= 0.05$ **(b)** adam with step size $= 0.10$

Figure 5.2: MCVI optimization for the bnn model applied to the wine data for various L and step sizes. The standard MC gradient estimator (dotted) was run with 2, 10, and 50 samples; RV-RGE (solid) was run with 2 and 10 samples. In 5.2b the 2-sample MC estimator falls below the frame.

step sizes .05 and .1 for the frisk model. We see that the HVP+Local estimators make early progress and converge quickly. We also see that the $L = 2$ pure MC estimator results in noisy optimization paths. Figure 5.2 shows objective value as a function of wall clock time under various settings for the bnn model. The HVP+Local estimator does more work per iteration, however it tends to converge faster. We observe the $L = 10$ HVP+Local outperforming the $L = 50$ MC estimator.

5.5 Conclusion

Variational inference reframes an integration problem as an optimization problem with the caveat that each step of the optimization procedure solves an easier integration problem. For general models, each sub-integration problem is itself intractable, and must be estimated, typically with Monte Carlo samples. Our work has shown that we can use more information about the variational family to create tighter estimators of the ELBO gradient, which leads to faster and more stable optimization. The efficacy of our approach relies on the complexity of the RGE distribution to be well-captured by linear structure which may not be true for all models. However, we found the idea effective for non-conjugate hierarchical Bayesian models and a neural network.

Our presentation is a specific instantiation of a more general idea — using cheap linear structure to remove variation from stochastic gradient estimates. This method described in this work is tailored to Gaussian approximating families for Monte Carlo variational inference, but could be easily extended to location-scale families. We plan to extend this idea to more flexible variational distributions, including flow distributions [Rezende and Mohamed, 2015] and hierarchical distributions [Ranganath et al., 2016], which would require approximating different functional forms within the variational objective. We also plan to adapt our technique to model and inference schemes with recognition networks [Kingma and Welling, 2013], which would require back-propagating de-noised gradients into the parameters of an inference network.

Appendix

5.A1 Control Variates

Control variates are random quantities that are used to reduce the variance of a statistical estimator without trading any bias. Concretely, given an unbiased estimator $\hat{\boldsymbol{g}}$ such that $\mathbb{E}[\hat{\boldsymbol{g}}] = \boldsymbol{g}$ (the quantity of interest), our goal is to construct another unbiased estimator with lower variance. We can do this by defining a *control variate* $\tilde{\boldsymbol{g}}$ with *known expectation* $\tilde{\boldsymbol{m}}$. We can write our new estimator as

$$\boldsymbol{g}^{(cv)} = \hat{\boldsymbol{g}} - c \cdot (\tilde{\boldsymbol{g}} - \tilde{\boldsymbol{m}}) \,. \tag{5.28}$$

Clearly the new estimator has the same expectation as the original estimator, but a different variance. We can reduce the variance of $\boldsymbol{g}^{(cv)}$ by setting c optimally.

Consider a univariate $\hat{\boldsymbol{g}}$ and $\tilde{\boldsymbol{g}}$, and without loss of generality, take $\tilde{\boldsymbol{m}} = 0$. The variance of $\boldsymbol{g}^{(cv)}$ can be written

$$\begin{aligned}
\mathbb{V}(\boldsymbol{g}^{(cv)}) &= \mathbb{E}[(\hat{\boldsymbol{g}} - c \cdot \tilde{\boldsymbol{g}})^2] - \mathbb{E}[\hat{\boldsymbol{g}}]^2 && (5.29)\\
&= \mathbb{E}[\hat{\boldsymbol{g}}^2 + c^2 \cdot \tilde{\boldsymbol{g}}^2 - 2c\hat{\boldsymbol{g}}\tilde{\boldsymbol{g}}] - \mathbb{E}[\hat{\boldsymbol{g}}]^2 && (5.30)\\
&= \mathbb{E}[\hat{\boldsymbol{g}}^2] + c^2\mathbb{E}[\tilde{\boldsymbol{g}}^2] - 2c\mathbb{E}[\hat{\boldsymbol{g}}\tilde{\boldsymbol{g}}] - \mathbb{E}[\hat{\boldsymbol{g}}]^2 && (5.31)
\end{aligned}$$

We minimize the variance with respect to c by taking the derivative and setting equal to zero, which implies

$$c^* = \frac{\mathbb{E}[\hat{\boldsymbol{g}}\tilde{\boldsymbol{g}}]}{\mathbb{E}[\tilde{\boldsymbol{g}}^2]} = \frac{\mathbb{C}(\hat{\boldsymbol{g}}, \tilde{\boldsymbol{g}})}{\mathbb{V}(\tilde{\boldsymbol{g}})} \tag{5.32}$$

The covariance $\mathbb{C}(\hat{\boldsymbol{g}}, \tilde{\boldsymbol{g}})$ is typically not known a priori and must be estimated. It can be shown, under the optimal c^*, that the variance of $\boldsymbol{g}^{(cv)}$ is

$$\mathbb{V}(\boldsymbol{g}^{(cv)}) = (1 - \rho^2)\mathbb{V}(\hat{\boldsymbol{g}}) \tag{5.33}$$

where ρ is the correlation coefficient between $\tilde{\boldsymbol{g}}$ and $\hat{\boldsymbol{g}}$.

When $\hat{\boldsymbol{g}}$ and $\tilde{\boldsymbol{g}}$ are length D vectors, we can construct an estimator that depends on a matrix-valued free parameter, $\boldsymbol{C} \in \mathbb{R}^{D \times D}$

$$\boldsymbol{g}^{(cv)} = \hat{\boldsymbol{g}} - \boldsymbol{C}(\tilde{\boldsymbol{g}} - \tilde{\boldsymbol{m}})\,. \tag{5.34}$$

We can show that the $\boldsymbol{C}$ that minimizes the $\texttt{Tr}(\mathbb{C}(\boldsymbol{g}^{(cv)}))$ — the sum of the marginal variances — is given by

$$\boldsymbol{C}^* = \Sigma_{\tilde{\boldsymbol{g}}}^{-1}\Sigma_{\hat{\boldsymbol{g}},\tilde{\boldsymbol{g}}} \tag{5.35}$$

where $\Sigma_{\tilde{\boldsymbol{g}}}$ is the covariance matrix of the control variate vector, and $\Sigma_{\hat{\boldsymbol{g}},\tilde{\boldsymbol{g}}}$ is the cross covariance between $\hat{\boldsymbol{g}}$ and $\tilde{\boldsymbol{g}}$.

Intuitively, a control variate is injecting information into the estimator in the form of linear structure. If the two quantities, $\tilde{\boldsymbol{g}}$ and $\hat{\boldsymbol{g}}$ are perfectly correlated, then we already know the mean and estimation is not necessary. As the two become uncorrelated, the linear estimator becomes less and less informative, and reverts to the original quantity.

5.A1.1 Control Variates and Approximate Functions

In our setting, we approximate the distribution of some function $\boldsymbol{f}(\epsilon)$ where $\epsilon \sim \mathcal{N}(0, I)$ by a first order Taylor expansion about 0 — for now we examine the univariate case

$$\boldsymbol{f}_1(\epsilon) = \boldsymbol{f}(0) + \boldsymbol{f}'(0)\epsilon \quad \epsilon \in \mathbb{R} \tag{5.36}$$

If we wish to use $\boldsymbol{f}_1(\epsilon)$ as a control variate for $\boldsymbol{f}(\epsilon)$, we need to characterize the covariance between the two random variables. Because the form of $\boldsymbol{f}(\epsilon)$ is general, it is difficult to analyze. We instead derive the covariance between $\boldsymbol{f}_1(\epsilon)$ and the second-order expansion

$$\boldsymbol{f}_2(\epsilon) = \boldsymbol{f}(0) + \boldsymbol{f}'(0)\epsilon + \boldsymbol{f}''(0)/2\epsilon^2 \tag{5.37}$$

as a surrogate.

$$\begin{aligned}
\mathbb{C}(\boldsymbol{f}_1(\epsilon), \boldsymbol{f}_2(\epsilon)) &= \mathbb{E}\left[(\boldsymbol{f}_1(\epsilon) - \mathbb{E}[\boldsymbol{f}_1(\epsilon)])(\boldsymbol{f}_2(\epsilon) - \mathbb{E}[\boldsymbol{f}_2(\epsilon)])\right] && (5.38)\\
&= \mathbb{E}\left[(\boldsymbol{f}'(0)\epsilon)\left(\boldsymbol{f}'(0)\epsilon + \boldsymbol{f}''(0)/2\epsilon^2 - \boldsymbol{f}''(0)/2\right)\right] && (5.39)\\
&= \mathbb{E}\left[\boldsymbol{f}'(0)^2\epsilon^2 + (\boldsymbol{f}'(0)\boldsymbol{f}''(0)/2)\epsilon^3 - (\boldsymbol{f}'(0)\boldsymbol{f}''(0)/2)\epsilon\right] && (5.40)\\
&= \mathbb{E}\left[\boldsymbol{f}'(0)^2\epsilon^2\right] && (5.41)\\
&= \mathbb{V}[\boldsymbol{f}_1(\epsilon)] && (5.42)
\end{aligned}$$

where note that $\mathbb{E}[\epsilon^3] = 0$. Recall that the optimal control variate can be written

$$\begin{aligned}
c^* &= \mathbb{C}(\boldsymbol{f}_1(\epsilon), \boldsymbol{f}_2(\epsilon))/\mathbb{V}[\boldsymbol{f}_1(\epsilon)] && (5.43)\\
&= \mathbb{V}[\boldsymbol{f}_1(\epsilon)]/\mathbb{V}[\boldsymbol{f}_1(\epsilon)] = 1\,. && (5.44)
\end{aligned}$$

5.A2 Algorithm Details

We summarize an optimization routine using RV-RGE in Algorithm 2. The different variants rely on the different forms of $\boldsymbol{H}(\cdot,\cdot)$ and $\mathrm{diag}(\boldsymbol{H})$. The *full Hessian* estimator calculates these terms exactly. The *diagonal Hessian* estimates the Hessian-vector product with the diagonal of the Hessian. The *HVP+Local* estimator computes the Hessian-vector product exactly, but estimates the scale approximation mean using other samples.

We also note that there are ways to optimize the additional Hessian-vector product computation. Because each Hessian is evaluated at the same $\boldsymbol{m}_{\boldsymbol{\lambda}}$, we can cache the computation in the forward pass, and only repeat the backwards pass for each sample, as implemented in Maclaurin et al. [2015b].

5.A3 Model Definitions

5.A3.1 Multi-level Poisson GLM

Our second test model is a 37-dimensional posterior resulting from a hierarchical Poisson GLM. This model measures the relative rates of stop-and-frisk events for different ethnicities and in different precincts Gelman et al. [2007], and has been used as illustrative example of multi-level modeling [Gelman and Hill, 2006, Chapter 15, Section 1].

$$\begin{aligned}
\mu &\sim \mathcal{N}(0, 10^2) && \text{mean offset} \\
\ln \sigma_\alpha^2, \ln \sigma_\beta^2 &\sim \mathcal{N}(0, 10^2) && \text{group variances} \\
\alpha_e &\sim \mathcal{N}(0, \sigma_\alpha^2) && \text{ethnicity effect} \\
\beta_p &\sim \mathcal{N}(0, \sigma_\beta^2) && \text{precinct effect} \\
\ln \lambda_{ep} &= \mu + \alpha_e + \beta_p + \ln N_{ep} && \text{log rate} \\
Y_{ep} &\sim \mathcal{P}(\lambda_{ep}) && \text{stop-and-frisk events}
\end{aligned}$$

where Y_{ep} are the number of stop-and-frisk events within ethnicity group e and precinct p over some fixed period of time; N_{ep} is the total number of arrests of ethnicity group e in precinct p over the same period of time; α_e and β_p are the ethnicity and precinct effects.

5.A3.2 Bayesian Neural Network

We implement a 50-unit hidden layer neural network with ReLU activation functions. We place a normal prior over each weight in the neural network, governed by the same variance (with an inverse Gamma prior). We also place an inverse Gamma prior over the observation variance The model can be written as

$$\begin{aligned}
\alpha &\sim \text{Gamma}(1, .1) && \text{weight prior hyper} && (5.45) \\
\tau &\sim \text{Gamma}(1, .1) && \text{noise prior hyper} && (5.46) \\
w_i &\sim \mathcal{N}(0, 1/\alpha) && \text{weights} && (5.47) \\
y|x, w, \tau &\sim \mathcal{N}(\phi(x, w), 1/\tau) && \text{output distribution} && (5.48)
\end{aligned}$$

where $w = \{w\}$ is the set of weights, and $\phi(x, w)$ is a multi-layer perceptron that maps input x to approximate output y as a function of parameters w. We denote the set of parameters as $\theta \triangleq (w, \alpha, \tau)$. We approximate the posterior $p(w, \alpha, \tau | \mathcal{D})$, where $\mathcal{D}$ is the training set of $\{x_n, y_n\}_{n=1}^N$ input-output pairs.

We use a 100-row subsample of the `wine` dataset from the UCI repository https://archive.ics.uci.edu/ml/datasets/Wine+Quality.

5.A4 Variance Reduction

Below are additional variance reduction measurements for the `frisk` model for different values of L, samples drawn per iteration. We measure the variance of the variational parameter gradient at three points during the optimization procedure: (i) early, near initialization, (ii) mid, before convergence, (iii) late, near convergence. We compare four methods

- MC: Monte Carlo estimator using the reparameterization trick
- Full Hessian: Our reduced variance gradient using the full hessian calculation
- Hessian Diag: Our reduced variance gradient using only diagonal Hessian information
- HVP + Local: Our fast reduced variance gradient estimator, using only Hessian-vector products and a local baseline
- Score Delta: Method described in Paisley et al. [2012] using a control variate with the score function estimator of the gradient.

Table 5.A4.1: `frisk` model variance comparison: $L = 3$-sample estimators

		$\boldsymbol{g}_{\boldsymbol{m}_\lambda}$		$\ln \boldsymbol{g}_{\boldsymbol{s}_\lambda}$		$\boldsymbol{g}_{\boldsymbol{\lambda}}$	
Iteration	Estimator	Ave $\mathbb{V}(\cdot)$	$\mathbb{V}(\lVert\cdot\rVert)$	Ave $\mathbb{V}(\cdot)$	$\mathbb{V}(\lVert\cdot\rVert)$	Ave $\mathbb{V}(\cdot)$	$\mathbb{V}(\lVert\cdot\rVert)$
early	(MC abs.)	(5.4e+02)	(1.7e+04)	(9.6e+04)	(5.9e+05)	(4.8e+04)	(1.9e+04)
	MC	100.000	100.000	100.000	100.000	100.000	100.000
	Full Hessian	1.184	1.022	0.001	0.002	0.007	0.902
	Hessian Diag	35.541	25.012	0.003	0.011	0.201	22.090
	HVP + Local	1.184	1.022	0.012	0.039	0.019	0.900
	Score Delta Paisley et al. [2012]	6054.168	651.784	1.429	1.783	35.134	574.536
mid	(MC abs.)	(1.4e+04)	(4.5e+05)	(63)	(1.1e+03)	(6.9e+03)	(4.5e+05)
	MC	100.000	100.000	100.000	100.000	100.000	100.000
	Full Hessian	0.080	0.075	0.122	0.169	0.081	0.075
	Hessian Diag	39.016	22.832	6.617	8.097	38.868	22.804
	HVP + Local	0.080	0.075	31.992	46.160	0.227	0.078
	Score Delta Paisley et al. [2012]	4787.771	1031.561	2833.663	1619.190	4778.818	1033.613
late	(MC abs.)	(5.6e+03)	(5.4e+04)	(4.1)	(74)	(2.8e+03)	(5.4e+04)
	MC	100.000	100.000	100.000	100.000	100.000	100.000
	Full Hessian	0.044	0.024	1.782	0.879	0.045	0.023
	Hessian Diag	39.280	38.799	22.915	21.913	39.268	38.725
	HVP + Local	0.044	0.024	98.290	99.679	0.116	0.014
	Score Delta Paisley et al. [2012]	5019.294	2804.652	15681.050	5650.339	5027.114	2810.160

Table 5.A4.2: `frisk` model variance comparison: $L = 50$-sample estimators

		$\boldsymbol{g}_{\boldsymbol{m}_\lambda}$		$\ln \boldsymbol{g}_{\boldsymbol{s}_\lambda}$		$\boldsymbol{g}_{\boldsymbol{\lambda}}$	
Iteration	Estimator	Ave $\mathbb{V}(\cdot)$	$\mathbb{V}(\lVert\cdot\rVert)$	Ave $\mathbb{V}(\cdot)$	$\mathbb{V}(\lVert\cdot\rVert)$	Ave $\mathbb{V}(\cdot)$	$\mathbb{V}(\lVert\cdot\rVert)$
early	(MC abs.)	(34)	(1.1e+03)	(6.1e+03)	(4e+04)	(3.1e+03)	(1.1e+03)
	MC	100.000	100.000	100.000	100.000	100.000	100.000
	Full Hessian	1.276	1.127	0.001	0.002	0.008	1.080
	Hessian Diag	35.146	24.018	0.003	0.012	0.197	23.028
	HVP + Local	1.276	1.127	0.013	0.039	0.020	1.079
	Score Delta Paisley et al. [2012]	6084.473	765.666	1.384	0.535	34.957	734.007
mid	(MC abs.)	(7.4e+02)	(2.4e+04)	(3.4)	(81)	(3.7e+02)	(2.4e+04)
	MC	100.000	100.000	100.000	100.000	100.000	100.000
	Full Hessian	0.081	0.074	0.125	0.121	0.081	0.074
	Hessian Diag	37.534	21.773	7.204	7.035	37.394	21.752
	HVP + Local	0.081	0.074	31.278	32.275	0.225	0.076
	Score Delta Paisley et al. [2012]	5115.048	557.946	3047.996	354.204	5105.546	557.329
late	(MC abs.)	(3.3e+02)	(1.8e+03)	(0.23)	(4.4)	(1.7e+02)	(1.8e+03)
	MC	100.000	100.000	100.000	100.000	100.000	100.000
	Full Hessian	0.042	0.043	1.894	0.296	0.044	0.043
	Hessian Diag	39.972	101.263	24.450	27.174	39.961	101.019
	HVP + Local	0.042	0.043	98.588	99.539	0.112	0.033
	Score Delta Paisley et al. [2012]	5192.542	1422.083	16907.603	1376.037	5200.855	1424.831

6

Taylor Residual Estimators

Monte Carlo estimation is a foundational statistical technique. Using samples from a probability distribution to estimate expectations is effective, efficient, and well-motivated by the law of large numbers. Further, Monte Carlo estimators are incredibly general—convergence of the estimator to the expectation $\mathbb{E}_{x\sim\pi}[f(x)]$ applies to a very general class of functions f and distributions π.

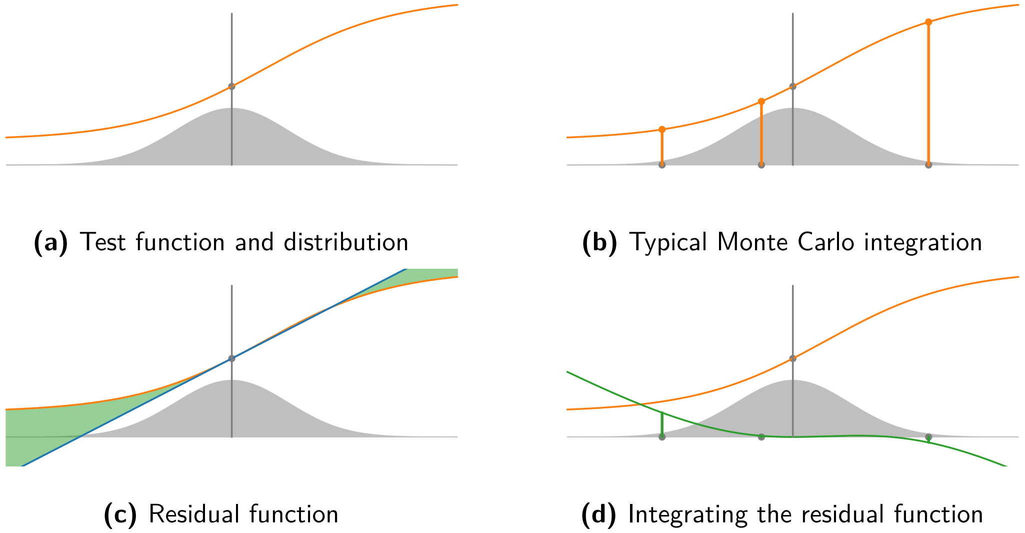
However, reducing the variance of Monte Carlo estimators typically requires the incorporation of additional samples. These additional samples will come at a cost—it may be expensive or practically difficult to collect additional data; new samples may come at an prohibitive computational cost. However, there may exist other sources of structure we can exploit to reduce the variance of our sample-based estimator—e.g. if f is smooth, or if we can efficiently compute the moments of π.

In this work, we develop a variant of Monte Carlo estimators termed *Taylor residual estimators* that we show can have better variance properties than general Monte Carlo estimators under certain conditions. Although this technique is generally applicable, we apply this to a variational inference algorithm where Monte Carlo estimation is a subroutine. The material in this chapter builds on the work in Miller et al. [2017b].

6.1 Introduction

Many fundamental problems in machine learning and statistics can be framed as the expectation of a function of a random variable. For example, modern variational inference algorithms for complex probabilistic models hinge on well-behaved Monte Carlo estimates of gradients. If the variance of the estimated gradient is large then gradient-based optimization can exhibit chaotic behavior or require such small step-sizes that the algorithm does not converge in a reasonable amount of time. A common approach is a Monte Carlo estimator, where the random variable is sampled (perhaps multiple times), the function is computed, and the values are averaged. The variance of the Monte Carlo estimate is a crucial property when applying Monte Carlo methods since a large variance can make the estimate unreliable. There has been a large body of literature on controlling the variance of Monte Carlo estimates such as *control variates* that reduce variance using a correlated estimate with the same mean as the original estimate. However, obtaining variance reduction is still a challenging problem.

In this work we develop a family of Monte Carlo estimators based on the Taylor expansion of the function being integrated. These estimators can be efficient to compute and easy to implement with modern automatic differentiation tools. We can interpret the resulting estimator as a control variate and we study the conditions under which the variance of the estimator is reduced. We apply the estimator to a Monte Carlo variational inference problem and show that the method achieves lower variance estimates of gradients.



(a) Test function and distribution

(b) Typical Monte Carlo integration

(c) Residual function

(d) Integrating the residual function

Figure 6.1: Illustration of Taylor residual estimators. (a) The function $f(x)$ (orange) and the probability distribution $\pi(x)$ (grey). (b) a typical Monte Carlo estimator simply draws values from $\pi(x)$ and computes $f(\cdot)$ at each value. (c) a first order approximation of $f(x)$ and the residual highlighted in green. (d) the Taylor residual estimator computes a Monte Carlo estimate of the expectation of the residual, analytically integrating out the expectation against the linear function. This can, under certain conditions, achieve lower variance than the naive Monte Carlo estimator in (b).

6.2 Taylor Residual Monte Carlo Estimator

Let $X \in \mathbb{R}^D$ be a random variable with distribution π. Consider a function $f : \mathbb{R}^D \mapsto \mathbb{R}$ whose expectation we would like to take with respect to π which we write $\mathbb{E}_\pi [f] = \int f(x)\pi(dx)$. In this work we assume the we can easily draw i.i.d. samples from π. The standard Monte Carlo estimator of $\mathbb{E}_\pi[f]$ is constructed by sampling from π and then computing the sample mean of f:

$$x^{(n)} \sim \pi, \quad \hat{f} = \frac{1}{N} \sum_{n=1}^{N} f(x^{(n)}). \tag{6.1}$$

While Eq. (6.1) is an extremely general way to estimate an expectation, it can be inefficient to ignore known structure in f and π which can manifest as a large amount of variance in the $\hat{f}$. We will assume that all moments of π are known and computable and we denote the mth moment of π about the point x_0 as

$$\mathcal{M}_{x_0}^{(m)} = \int (x - x_0)^m \pi(dx)\,. \tag{6.2}$$

Now, consider decomposing f into (i) its first order Taylor expansion around x_0 and (ii) the residual:

$$f(x) = \underbrace{f(x_0) + (x - x_0)^\intercal \left[\frac{\partial f}{\partial x}(x_0)\right]}_{\triangleq f_{x_0}^{(1)}(x)} + R_{x_0}^{(1)}(x), \tag{6.3}$$

where the Taylor remainder $R_{x_0}^{(1)}(x)$ can be determined from the second-order derivatives of f. We can re-write the target expectation as

$$\begin{aligned}
\mathbb{E}_\pi[f] &= \mathbb{E}_\pi\left[f_{x_0}^{(1)}(x) + R_{x_0}^{(1)}(x)\right] && (6.4)\\
&= f(x_0) + \mathbb{E}_\pi[(x - x_0)]^\intercal \left[\frac{\partial f}{\partial x}(x_0)\right] + \mathbb{E}_\pi\left[R_{x_0}^{(1)}(x)\right] && (6.5)\\
&= f(x_0) + \mathcal{M}_{x_0}^{(1)\intercal} \left[\frac{\partial f}{\partial x}(x_0)\right] + \mathbb{E}_\pi\left[R_{x_0}^{(1)}(x)\right], && (6.6)
\end{aligned}$$

In general, we can use an M^{th}-order Taylor expansion about x_0 and write the expectation as

$$\mathbb{E}_\pi[f] = f(x_0) + \sum_m \mathcal{M}_{x_0}^{(m)} \left[\frac{\partial^m f}{\partial x^m}(x_0)\right] + \mathbb{E}_\pi\left[R_{x_0}^{(M)}(x)\right]. \tag{6.7}$$

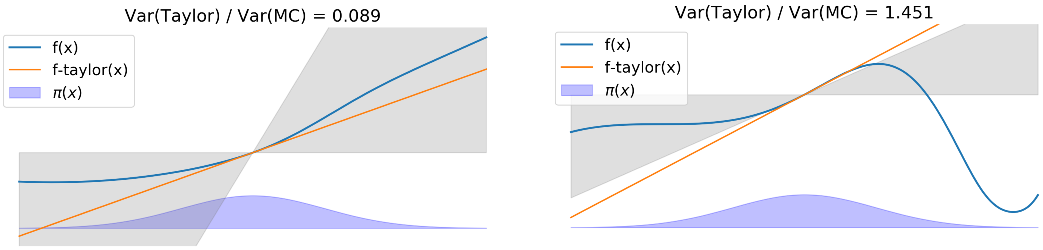
In this case the Taylor remainder $R_{x_0}^{(M)}(x)$ can be found from the $(M+1)^{\text{st}}$ order derivatives of f.

Because we assume all moments $\mathcal{M}_{x_0}^{(m)}$ are known we see that all of the randomness in the estimators given in Eqs. (6.3) and (6.7) comes from the expectation of the remainder term. We call estimators of this form *Taylor residual Monte Carlo estimators* (TREs). Note that we have simply shifted the variance of the Monte Carlo estimate into the higher-order derivatives of the function. As such, we can expect the residual to have low variance when the low order derivatives well-approximate f around x_0. Taylor residual Monte Carlo estimates can be viewed as performing approximate Rao-Blackwellization in that the aspects of f captured in the low-order derivatives is being integrated out and replaced with non-random quantities. Furthermore, using modern automatic differentiation tools [Maclaurin et al., 2015b, Abadi et al., 2016, Paszke et al., 2017] we can easily compute higher order derivatives of scalar functions and the requisite tensor contractions.

We can interpret the first-order Taylor residual estimate in Eq. (6.3) as a control-variate estimator, implying that TREs may achieve smaller variance than that of pure Monte Carlo estimators. To see the connection to control variates, consider a single sample first order TRE:

$$\begin{aligned}
\mathbb{E}_\pi[f] &= f(x_0) + \mathcal{M}_{x_0}^{(1)\intercal}\left[\frac{\partial f}{\partial x}(x_0)\right] + R_{x_0}^{(1)}(x) && (6.8)\\
&= f(x_0) + \mathcal{M}_{x_0}^{(1)\intercal}\left[\frac{\partial f}{\partial x}(x_0)\right] + \left[f(x) - \left(f(x_0) + (x - x_0)^\intercal\left[\frac{\partial f}{\partial x}(x_0)\right]\right)\right] && (6.9)\\
&= f(x) - \left(\mathcal{M}_{x_0}^{(1)} - (x - x_0)\right)^\intercal\left[\frac{\partial f}{\partial x}(x_0)\right] && (6.10)
\end{aligned}$$

where we recognize Eq. (6.10) as the equation for a control variate with scale coefficient 1. In fact, first-order Taylor residual estimators generalize the reduced variance gradient estimators presented in [Miller et al., 2017a] and provide a framework to study when such gradient estimators will be effective. In the next section we study the variance properties of TREs to determine conditions under which we attain variance reduction.



(a) Variance reduced

(b) Variance increased

Figure 6.1: Illustration of the conditions for TRE variance reduction. In each example, the gray area indicates the set of linear approximations to $f(x)$ that result in decreased variance, as indicated by Equation (6.20). (a) When the first-order Taylor approximation of f at x_0 (orange line) is in the gray region then the corresponding TRE will have smaller variance than the Monte Carlo estimator. Functions that are close to linear in the range of π will have a larger region where variance reduction occurs while highly nonlinear functions will have smaller regions. (b) The TRE estimator can have larger variance than the MC estimator when the gradient at x_0 falls outside of the gray region.

6.3 Variance Analysis

The Taylor residual estimator is useful if its variance is lower than that of the standard Monte Carlo estimator. In this section we will show that the variance properties of the TRE depend on the relationship between the locally linear Taylor approximation and the global linear structure captured by linear least squares regression.

Consider the Monte Carlo estimator given in Eq. (6.12) and the first order Taylor residual estimator in Eq. (6.13) using a single sample from π. For notational simplicity, we take $x_0 = 0$ and define $f_0 = f(0)$, as well as $f_0' = \frac{\partial f}{\partial x}(0)$ as shorthand. We write the estimators as

$$x \sim \pi \qquad \text{sample from distribution} \tag{6.11}$$

$$\hat{f} = f(x) \qquad \text{Monte Carlo estimator} \tag{6.12}$$

$$\hat{f}_1 = f(x) - (f_1(x) - \mathbb{E}[f^{(1)}]) \qquad \text{First order Taylor residual estimator} \tag{6.13}$$

$$= f(x) - xf_0' + \mu f_0', \tag{6.14}$$

where $\mu = \mathbb{E}(x)$ is the known first moment of $\pi(x)$. The variances of the two estimators are then

$$\mathbb{V}(\hat{f}) = \mathbb{E}\left[\hat{f}^2\right] - \mathbb{E}[f]^2 \tag{6.15}$$

$$\mathbb{V}(\hat{f}_1) = \mathbb{V}(f(x) - xf_0' + \mu f_0') = \mathbb{E}\left[(f(x) - xf_0')^2\right] - \left(\mathbb{E}[f] - \mu f_0'\right)^2. \tag{6.16}$$

We want to find sufficient conditions such that the variance of the new estimator is smaller than the original, $\mathbb{V}(\hat{f}) \geq \mathbb{V}(\hat{f}_1)$. We first substitute the variances with their definitions into the inequality

$$\mathbb{E}[f(x)^2] - \mathbb{E}[f]^2 \geq \mathbb{E}\left[(f(x) - xf_0')^2\right] - \left(\mathbb{E}[f] - \mu f_0'\right)^2. \tag{6.17}$$

Expanding the two quadratics, and canceling terms, we get

$$\begin{aligned} 0 &\geq \mathbb{E}[x^2](f_0')^2 - 2f_0'\mathbb{E}[xf(x)] - \mu^2(f_0')^2 + 2f_0'\mu\mathbb{E}[f] && (6.18)\\ &= (f_0')^2\mathbb{V}(x) - 2f_0'\mathbb{E}[xf(x)] + 2f_0'\mu\mathbb{E}[f] && (6.19)\\ &\implies 1 \leq \frac{2}{f_0'}\frac{\mathbb{C}(x, f(x))}{\mathbb{V}(x)} \implies |f_0'| \leq 2\left|\frac{\mathbb{C}(x, f(x))}{\mathbb{V}(x)}\right|. && (6.20)\end{aligned}$$

Eq. (6.20) indicates a relationship between linear control-variate methods and linear least-squares regression. Since $\mathbb{V}(x)^{-1}\mathbb{C}(x, f(x))$ is the population least squares solution for f regressed on x, we see that variance reduction depends on whether the first order Taylor expansion of f is within a cone around the linear least squares approximation. We visually depict both successful and unsuccessful variance reduction for a one-dimensional example in Fig. 6.1.

Now the question is, under what conditions of $f(x)$ and $\pi(x)$ is this condition true?

We can start by re-writing the covariance using a taylor-expanded $f(x)$

$$\begin{aligned}
\mathbb{C}(x, f(x)) &= \mathbb{E}[xf(x)] - \mathbb{E}[x]\mathbb{E}[f(x)] && (6.21)\\
&= \mathbb{E}\left[x(f_0 + \sum_{n=1}^{\infty} \frac{1}{n!} f_0^{(n)} x^n)\right] - \mu\mathbb{E}\left[f_0 + \sum_{n=1}^{\infty} \frac{1}{n!} f_0^{(n)} x^n\right] && (6.22)\\
&= \sum_{n=1} \frac{1}{n!} f_0^{(n)} \left(\mathbb{E}[x^{n+1}] - \mu\mathbb{E}[x^n]\right) && (6.23)\\
&= \sum_{n=1} \frac{1}{n!} f_0^{(n)} \left(\mathcal{M}_\pi^{(n+1)} - \mu\mathcal{M}_\pi^{(n)}\right) \qquad \text{moments of } \pi && (6.24)
\end{aligned}$$

The first term of this series is a simple function of the variance of π

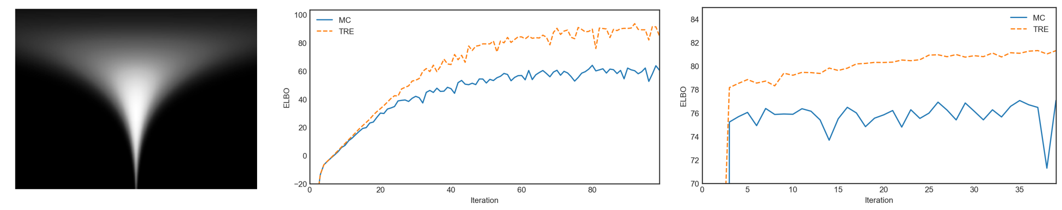
$$\begin{aligned}
&\sum_{n=1} \frac{1}{n!} f_0^{(n)} \left(\mathcal{M}_\pi^{(n+1)} - \mu\mathcal{M}_\pi^{(n)}\right) && (6.25)\\
&= f_0^{(1)}(\mathcal{M}_\pi^{(2)} - \mu\mathcal{M}_\pi^{(1)}) + \sum_{n=2} \frac{1}{n!} f_0^{(n)} \left(\mathcal{M}_\pi^{(n+1)} - \mu\mathcal{M}_\pi^{(n)}\right) && (6.26)\\
&= f_0^{(1)}\mathbb{V}(x) + \sum_{n=2} \frac{1}{n!} f_0^{(n)} \left(\mathcal{M}_\pi^{(n+1)} - \mu\mathcal{M}_\pi^{(n)}\right) && (6.27)
\end{aligned}$$

So we can express the inequality above as a bound on the variance of $x \sim \pi$ as a function of the higher moments of π and derivatives of f

$$\sum_{n=2} \frac{1}{n!} \frac{f_0^{(n)}}{f_0^{(1)}} \left(\mathcal{M}_\pi^{(n+1)} - \mu\mathcal{M}_\pi^{(n)}\right) \leq \frac{1}{2}\mathbb{V}(x) \tag{6.28}$$

For the first order estimator to have reduced variance, a scaled sum of the difference of higher order moments needs to be smaller than the variance of $x \sim \pi$.

For example, if $\pi(x) = \mathcal{N}(0, \sigma^2)$, then the n'th even moment is $\sigma^n(n-1)!!$ (note that $(n-1)!!$ is the double factorial, which is the product of a decreasing sequence of numbers with the same parity, e.g. $(n-1)(n-3)(n-5)...$), and the odd moments are



(a) Target distribution

(b) Gaussian Approximation

(c) Normalizing Flows Approximation (4 layers)

Figure 6.1: Comparison of Taylor residual and Monte Carlo estimators on Monte Carlo variational inference optimization using both a Gaussian variational distribution and a normalizing flow. In both cases, TREs provide lower variance gradient estimates and attain higher lower-bounds.

zero. We can write the inequality as

$$\sum_{n=2} \frac{1}{n!}\frac{f_0^{(n)}}{f_0^{(1)}}\sigma^{(n+1)}(n)!! = \sum_{n=2}\frac{f_0^{(n)}}{f_0^{(1)}}\frac{\sigma^{(n+1)}}{(n-1)!!} \leq \frac{1}{2}\mathbb{V}(x) \tag{6.29}$$

So in this case we can see that the inequality is easily achieved when the variance of π, σ^2, is small, and when the ratio of higher order derivatives to the first derivative $\frac{f_0^{(n)}}{f_0^{(1)}}$ is small.

6.4 Experiments

We demonstrate the variance reduction capabilities of TREs in the context of Monte Carlo variational inference. Specifically, we compare a TRE to the pure Monte Carlo estimator on the variational *evidence lower bound* (ELBO). We target a 20-dimensional "funnel" distribution that exhibits features typical of posteriors from hierarchical models [Neal, 2003]. A bivariate marginal of the "funnel" is depicted in Fig. 6.1a. We consider two variational approximations, a Gaussian and a normalizing flow distribution and show that TREs attain lower variance estimates and yield more robust optimization.

Gaussian Approximation For variational approximation $q(x;\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\lambda}_\mu, \boldsymbol{\lambda}_\sigma)$, with variational parameters $\boldsymbol{\lambda}$, the Monte Carlo ELBO estimator can be computed by first drawing a sample $x \sim q(x; \boldsymbol{\lambda})$, and then computing

$$f(x) = \ln \pi(x, \mathcal{D}) - \ln q(x; \boldsymbol{\lambda}). \tag{6.30}$$

We optimize the ELBO by using estimators of the gradient of Eq (6.30) with respect to $\boldsymbol{\lambda}$. We compute the pathwise gradient estimator (reparameterization gradient) [Glasserman, 2004] for both the MC and TRE estimators, and use these noisy gradient estimates in gradient ascent.

At a random initialization of $\boldsymbol{\lambda}$, we measure the variance of the first order Taylor residual estimator to be about 320 times lower than the Monte Carlo estimator (for 2 samples). We show the results of ELBO optimization in Figure 6.1 using a 2-sample Monte Carlo estimator and a 2-sample Taylor residual estimator. The TRE estimator has a smaller variance for more iterations than the MC estimator allowing it to attain larger ELBO values. After convergence, we measure the TR estimator to have .8 the variance of the MC estimator.

Normalizing Flows We also apply the Taylor residual estimator to a more flexible posterior approximation, a planar normalizing flow distribution [Rezende and Mohamed, 2015]. A normalizing flow distributed random variable is constructed by applying a sequence of parameterized invertible maps to a simple random variable (e.g. $x_0 \sim \mathcal{N}(0, I_D)$). Here, we broke the ELBO into two pieces

$$\mathcal{L}(\boldsymbol{\lambda}) = \underbrace{\mathbb{E}_q[\ln \pi(x, \mathcal{D})]}_{\text{model term}} - \underbrace{\mathbb{E}[\ln q(x; \boldsymbol{\lambda})]}_{\text{entropy term}}. \tag{6.31}$$

Unlike for the Gaussian variational family where the entropy term can be computed exactly and the model term is the only random component, for normalizing flows, we must estimate the entropy term using Monte Carlo. Here, we apply a TRE to the model term and use the simple Monte Carlo estimator for the entropy term. We found this resulted in consistent variance reduction compared to the Monte Carlo estimator. At initialization we measure a 40× variance reduction over the standard Monte Carlo estimator, and a 2× reduction at convergence. Fig. 6.1c shows the results of optimization using the TRE where it is clear that the optimization is more stable.

6.5 Conclusion

We presented Taylor residual estimators to efficiently compute lower variance Monte Carlo estimators by using a Taylor expansion. We showed that when a selected locally linear Taylor approximation aligns with the global least squares linear approximation the proposed estimator will have lower variance than the standard Monte Carlo estimator. The advantages of the TRE method were demonstrated on performing Monte Carlo variational inference where we obtained more robust optimization results under two variational approximations. We plan to extend the method to estimate highly nonlinear functions using a hierarchical approach that combines locally linear approximations.

7

Conclusions and Future Directions

We have developed new statistical models and methods to help scientists answer quantitative questions and more efficiently explore their data. The first part of this thesis focused on the development of applied probabilistic models in three domains: astronomy, healthcare, and sports analytics. Throughout this presentation, we highlighted application-driven decisions as well as some common statistical and computational challenges. Our applied goal in each instance was to model some difficult-to-prescribe structure—spectral energy distribution function shape, electrocardiogram traces, and basketball player trajectories—in service of some interpretable estimand (e.g. redshift, cardiac cycle morphology, offense structure). Through the lens of these projects, we discuss the common challenges of specifying interpretable-yet-flexible probabilistic models in applied settings.

Motivated by the applied probabilistic models for massive data, the second part of this thesis described new scalable inference algorithms for approximate Bayesian inference. We focused on improving approaches to variational inference, a widely used class of approximation algorithms. We developed two techniques to address two common shortcomings of variational inference algorithms, the expressivity of the approximating family and the variance in stochastic estimators used when optimizing the variational lower bound. This work led to further generalization of a class of computationally efficient Monte Carlo estimators.

7.1 Directions of future research

While models for specific phenomena often require problem-specific structure, understanding more general properties of probabilistic models for high-dimensional data is an open area of research. Deep generative models [Kingma and Welling, 2013, Rezende et al., 2014] are a promising avenue for specifying accurate high-dimensional distributions for data that exist on a low-dimensional manifold. However, controlling the information contained in (and how it is represented) the latent space is still an open and active area of research. Recent approaches incorporate weak supervision [Kingma et al., 2014], and information theoretic concepts to better understand and control the properties of the latent representation of a high-dimensional signal [Tishby and Zaslavsky, 2015, Achille and Soatto, 2018, Chen et al., 2018]. Shaping these latent variables into an interpretable representation will provide a window into the subtle patterns learned by these generative models. This will highlight dimensions of variation within the data that we expect, and, importantly, dimensions of variation that we have yet to explain with scientific theories.

When analyzing passively collected data (e.g. electronic medical records), we often have to deal with problematic patterns of non-random missing data. Similarly, when inferring causal effects from observational data, we often have to cope with non-random treatment assignments and selection bias [Little and Rubin, 2014]. Coping with this requires modeling the patterns of missing-ness, often by making structural assumptions about the underlying data generating procedure. However, it is often difficult (and sometimes impossible) to validate such assumptions—and when a model is providing answers to high-leverage decisions (e.g. a medical diagnosis) this lack of validation is untenable. How to *efficiently* validate modeling assumptions by the design of a new experiment or an efficient, targeted collection of new data is an avenue of research that can help bring new models and methods from theory to practice.

When datasets become massive and particularly when the amount of data collected *per individual* grows, our expectations of the data analysis also tends to grow—we want more personalized predictions and more nuanced measurements that may have been inestimable with less information. However, naively applying predictions from a probabilistic models can lead to problematic decision-making. For example, systems that predict criminal risk used for bail decisions Angwin et al. or systems that serve advertisements for employment opportunities Garcia can be fraught with unfairness and insidious bias—models trained on observed data that reflects exist discriminatory practices can exacerbate them when algorithmic decision-making is naive applied. Defining and understanding precise notions of fairness in machine learning and algorithmic decision-making is an active and important area of research [Kleinberg et al., 2016, Hardt et al., 2016].

Massive datasets and per-individual inference expectations also lead to an increasingly complex computational problem. For example, hierarchical models that may enable more personalized statistical characterizations can introduce a cumbersome number of parameters, requiring us to approximate posterior interactions between units. While Markov chain Monte Carlo remains the gold standard for approximate Bayesian inference in terms of asymptotic correctness, it can be difficult to scale as data size increases. An active area of research is devising algorithms with correctness guarantees that scale favorably with data size. Furthermore, models that fully characterize uncertainty may need to represent and infer multi-modal posterior distributions, which may be intractable in theory and difficult to approximate in practice.

References

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

Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

Shadab Alam, Franco D Albareti, Carlos Allende Prieto, F Anders, Scott F Anderson, Brett H Andrews, Eric Armengaud, Éric Aubourg, Stephen Bailey, Julian E Bautista, et al. The eleventh and twelfth data releases of the Sloan digital sky survey: Final data from SDSS-III. *arXiv preprint arXiv:1501.00963*, 2015.

Julia Angwin, Jeff Larson, Surya Mattu, Lauren Kirchner, and ProPublica. Machine Bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Accessed: 2017-11-04.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

Costas Bekas, Effrosyni Kokiopoulou, and Yousef Saad. An estimator for the diagonal of a matrix. *Applied numerical mathematics*, 57(11):1214–1229, 2007.

C Bishop. Pattern recognition and machine learning, 2006.

David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 2017a.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017b.

Matt Hoffman Daniel Lee Ben Goodrich Michael Betancourt Michael A. Brubaker Jiqiang Guo Peter Li Bob Carpenter, Andrew Gelman and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 2015.

R Bousseljot, D Kreiseler, and A Schnabel. Nutzung der ekg-signaldatenbank cardiodat der ptb über das internet. *Biomedizinische Technik/Biomedical Engineering*, 40(s1): 317–318, 1995.

Jo Bovy, Adam D Myers, Joseph F Hennawi, David W Hogg, Richard G McMahon, David Schiminovich, Erin S Sheldon, Jon Brinkmann, Donald P Schneider, and Benjamin A Weaver. Photometric redshifts and quasar probabilities from a single, data-driven generative model. *The Astrophysical Journal*, 749(1):41, 2012.

M Brescia, S Cavuoti, R D'Abrusco, G Longo, and A Mercurio. Photometric redshifts for quasars in multi-band surveys. *The Astrophysical Journal*, 772(2):140, 2013.

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

Tamas Budavari, Istvan Csabai, Alexander S Szalay, Andrew J Connolly, Gyula P Szokoly, Daniel E Vanden Berk, Gordon T Richards, Michael A Weinstein, Donald P Schneider, Narciso Benitez, et al. Photometric redshifts from reconstructed quasar templates. *The Astronomical Journal*, 122(3):1163, 2001.

George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA.

Daniel Cervone, Alexander D'Amour, Luke Bornn, and Kirk Goldsberry. A multiresolution stochastic process model for predicting basketball possession outcomes. *arXiv preprint arXiv:1408.0777*, 2014a.

Daniel Cervone, Alexander D'Amour, Luke Bornn, and Kirk Goldsberry. Pointwise: predicting points and valuing decisions in real time with NBA optical tracking data. 2014b.

Gal Chechik, Amir Globerson, Naftali Tishby, and Yair Weiss. Information bottleneck for gaussian variables. *Journal of machine learning research*, 6(Jan):165–188, 2005.

Tian Qi Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. *arXiv preprint arXiv:1802.04942*, 2018.

Chih-Chun Chia and Zeeshan Syed. Scalable noise mining in long-term electrocardiographic time-series to predict death following heart attacks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 125–134. ACM, 2014.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Harald Cramér. *Mathematical methods of statistics*. Princeton university press, 1946.

Kyle S Dawson, David J Schlegel, Christopher P Ahn, Scott F Anderson, Éric Aubourg, Stephen Bailey, Robert H Barkhouser, Julian E Bautista, Alessandra Beifiori, Andreas A Berlind, et al. The baryon oscillation spectroscopic survey of SDSS-III. *The Astronomical Journal*, 145(1):10, 2013.

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

David Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, Computational and Biological Learning Laboratory, University of Cambridge, 2014.

Bradley Efron and Carl Morris. Data analysis using stein's estimator and its generalizations. *Journal of the American Statistical Association*, 70(350):311–319, 1975.

Alexander Franks, Andrew Miller, Luke Bornn, and Kirk Goldsberry. Counterpoints: Advanced defensive metrics for nba basketball. In *2015 MIT Sloan Sports Analytics Conference*, 2015a.

Alexander Franks, Andrew Miller, Luke Bornn, Kirk Goldsberry, et al. Characterizing the spatial structure of defensive skill in professional basketball. *The Annals of Applied Statistics*, 9(1):94–121, 2015b.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

Megan Garcia. How To Keep Your AI From Turning Into A Racist Monster. https://www.wired.com/2017/02/keep-ai-turning-racist-monster/. Accessed: 2017-11-04.

Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, 2006.

Andrew Gelman and Xiao-Li Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, pages 163–185, 1998.

Andrew Gelman and Cosma Rohilla Shalizi. Philosophy and the practice of bayesian statistics. *British Journal of Mathematical and Statistical Psychology*, 66(1):8–38, 2013.

Andrew Gelman, Jeffrey Fagan, Alex Kiss, et al. An analysis of the nypd's stop-and-frisk policy in the context of claims of racial bias. *Journal of the American Statistical Association*, 102:813–823, 2007.

Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*, volume 2. Taylor & Francis, 2014.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. In *Readings in Computer Vision*, pages 564–584. Elsevier, 1987.

Samuel Gershman, Matt Hoffman, and David M Blei. Nonparametric variational inference. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 663–670, 2012.

Zoubin Ghahramani and Geoffrey E Hinton. The em algorithm for mixtures of factor analyzers. Technical report, 1996.

Paul Glasserman. *Monte Carlo Methods in Financial Engineering*, volume 53. Springer Science & Business Media, 2004.

Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media, 2013.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. JHU Press, 2013.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

RO Gray, PW Graham, and SR Hoyt. The physical basis of luminosity classification in the late a-, f-, and early g-type stars. ii. basic parameters of program stars and the role of microturbulence. *The Astronomical Journal*, 121(4):2159, 2001.

Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

Fangjian Guo, Xiangyu Wang, Kai Fan, Tamara Broderick, and David B. Dunson. Boosting variational inference. arXiv:1611.05559 [stat.ML], 2016.

Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016.

Edward Harrison. The redshift-distance and velocity-distance laws. *The Astrophysical Journal*, 403:28–31, 1993.

D. A. Harville. *Matrix Algebra from a Statistician's Perspective.* Springer-Verlag, 1997.

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

José Miguel Hernández-Lobato and Ryan P Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. 2015.

Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15 (1):1593–1623, 2014.

Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. 2016.

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

David W Hogg. Distance measures in cosmology. *arXiv preprint astro-ph/9905116*, 1999.

Tommi S Jaakkola and Michael I Jordan. Improving the mean field approximation via the use of mixture distributions. In *Learning in graphical models*, pages 163–173. Springer, 1998.

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

Edwin T Jaynes. Prior probabilities. *IEEE Transactions on systems science and cybernetics*, 4(3):227–241, 1968.

Matthew J. Johnson, David K. Duvenaud, Alex B. Wiltschko, Sandeep R. Datta, and Ryan P. Adams. Composing graphical models with neural networks for structured representations and fast inference. *Arxiv preprint arXiv:1603.06277*, 2016.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

Michael I Jordan. Are you a bayesian or a frequentist?, 2009.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2): 183–233, 1999.

Stephen M Kent, Chris Stoughton, Heidi Newberg, Jonathan Loveday, Don Petravick, Vijay Gurbani, Eileen Berman, and Gary Sergey. Sloan digital sky survey. *Astronomical Data Analysis Software and Systems III*, 61.

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

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

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*, 2016.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

D Kreiseler and R Bousseliot. Automatisierte ekg-auswertung mit hilfe der ekg-signaldatenbank cardiodat der ptb. *Biomedizinische Technik/Biomedical Engineering*, 40(s1):319–320, 1995.

Q. Li. *Estimation of Mixture Models.* PhD thesis, Yale University, May 1999.

Q. J. Li and A. R. Barron. Mixture density estimation. In *Advances in Neural Information Processing Systems*, 1999.

Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 333. John Wiley & Sons, 2014.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

Dougal Maclaurin and Ryan P Adams. Firefly monte carlo: Exact mcmc with subsets of data. *arXiv preprint arXiv:1403.5693*, 2014.

Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Autograd: Reverse-mode differentiation of native python. *ICML workshop on Automatic Machine Learning*, 2015a.

Dougal Maclaurin, David Duvenaud, Matthew Johnson, and Ryan P. Adams. Autograd: Reverse-mode differentiation of native Python, 2015b. URL http://github.com/HIPS/autograd.

James Martens, Ilya Sutskever, and Kevin Swersky. Estimating the Hessian by backpropagating curvature. In *Proceedings of the International Conference on Machine Learning*, 2012.

D Christopher Martin, James Fanson, David Schiminovich, Patrick Morrissey, Peter G Friedman, Tom A Barlow, Tim Conrow, Robert Grange, Patrick N Jelinksy, Bruno Millard, et al. The galaxy evolution explorer: A space ultraviolet survey mission. *The Astrophysical Journal Letters*, 619(1), 2005.

Bertil Matérn. *Spatial Variation.* Springer, 1986.

Patrick E McSharry, Gari D Clifford, Lionel Tarassenko, and Leonard A Smith. A dynamical model for generating synthetic electrocardiogram signals. *IEEE transactions on biomedical engineering*, 50(3):289–294, 2003.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Andrew Miller, Luke Bornn, Ryan Adams, and Kirk Goldsberry. Factorized point process intensities: A spatial analysis of professional basketball. In *International Conference on Machine Learning*, pages 235–243, 2014.

Andrew Miller, Albert Wu, Jeff Regier, Jon McAuliffe, Dustin Lang, Mr Prabhat, David Schlegel, and Ryan P Adams. A gaussian process model of quasar spectral energy distributions. In *Advances in Neural Information Processing Systems*, pages 2494–2502, 2015.

Andrew Miller, Nick Foti, Alexander D'Amour, and Ryan P Adams. Reducing reparameterization gradient variance. In *Advances in Neural Information Processing Systems*, pages 3711–3721, 2017a.

Andrew C Miller and Luke Bornn. Possession sketches: Mapping nba strategies. In *Sloan Sports Analytics Conference*, 2017.

Andrew C Miller, Nicholas J Foti, and Ryan P Adams. Taylor residual estimators via automatic differentiation. In *Advances in Approximate Bayesian Inference, NIPS Workshop*, 2017b.

Andrew C Miller, Nicholas J Foti, and Ryan P Adams. Variational boosting: Iteratively refining posterior approximations. In *International Conference on Machine Learning*, pages 2420–2429, 2017c.

Andrew C Miller, Sendhil Mullainathan, and Ziads Obermeyer. A hierarchical generative model of electrocardiogram records. In *Machine Learning for Health (NIPS Workshop)*, 2017d.

Andriy Mnih and Danilo Rezende. Variational inference for Monte Carlo objectives. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2188–2196, 2016.

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

D. A. Moore. Symmetrized variational inference. In *NIPS Workshop on Advances in Approximate Bayesian Inferece*, 2016.

Kevin P Murphy. *Machine learning: a probabilistic perspective.* 2012.

Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11(2): 125–139, 2001.

Radford M Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003.

Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

V. M.-H. Ong, D. J. Nott, and M. S. Smith. Gaussian variational approximation with factor covariance structure. *arXiv preprint arXiv:1701.03208*, 2017.

Julien Oster, Joachim Behar, Omid Sayadi, Shamim Nemati, Alistair EW Johnson, and Gari D Clifford. Semisupervised ecg ventricular beat classification with novelty detection based on switching kalman filters. *IEEE Transactions on Biomedical Engineering*, 62(9):2125–2134, 2015.

John Paisley, David M Blei, and Michael I Jordan. Variational bayesian inference with stochastic search. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1363–1370. Omnipress, 2012.

Isabelle Pâris, Patrick Petitjean, Éric Aubourg, Nicholas P Ross, Adam D Myers, Alina Streblyanska, Stephen Bailey, Patrick B Hall, Michael A Strauss, Scott F Anderson, et al. The Sloan digital sky survey quasar catalog: tenth data release. *Astronomy & Astrophysics*, 563:A54, 2014.

Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch. https://github.com/pytorch/pytorch, 2017.

Barak A Pearlmutter. Fast exact multiplication by the Hessian. *Neural computation*, 6(1):147–160, 1994.

A. Rakhlin, Panchenko D., and Mukherjee S. Risk bounds for mixture density estimation. *ESAIM: Probability and Statistics*, 9:220–229, 2006.

Rajesh Ranganath, Sean Gerrish, and David M Blei. Black box variational inference. In *AISTATS*, pages 814–822, 2014.

Rajesh Ranganath, Dustin Tran, and David M Blei. Hierarchical variational models. In *International Conference on Machine Learning*, 2016.

Carl Edward Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, Cambridge, Massachusetts, 2006.

Jeffrey Regier, Andrew Miller, Jon McAuliffe, Ryan Adams, Matt Hoffman, Dustin Lang, David Schlegel, and Prabhat. Celeste: Variational inference for a generative model of astronomical images. In *Proceedings of The 32nd International Conference on Machine Learning*, 2015.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1530–1538, 2015.

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

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

Geoffrey Roeder, Yuhuai Wu Wu, and David Duvenaud. Sticking the landing: An asymptotically zero-variance gradient estimator for variational inference. *arXiv preprint arXiv:1703.09194*, 2017.

Francisco R Ruiz, Michalis Titsias RC AUEB, and David Blei. The generalized reparameterization gradient. In *Advances in Neural Information Processing Systems*, pages 460–468, 2016.

Ruslan Salakhutdinov, Sam T Roweis, and Zoubin Ghahramani. Optimization with em and expectation-conjugate-gradient. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 672–679, 2003.

Tim Salimans, David A Knowles, et al. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

SDSSIII. Measures of flux and magnitude. 2013. https://www.sdss3.org/dr8/algorithms/magnitudes.php.

M. W. Seeger. Gaussian covariance and scalable variational inference. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

Long Sha, Patrick Lucey, Yisong Yue, Peter Carr, Charlie Rohlf, and Iain Matthews. Chalkboarding: A new spatiotemporal query paradigm for sports play retrieval. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 336–347. ACM, 2016.

Joseph Silk and Martin J Rees. Quasars and galaxy formation. *Astronomy and Astrophysics*, 1998.

John Skilling et al. Nested sampling for general bayesian computation. *Bayesian analysis*, 1(4):833–859, 2006.

Chris Stoughton, Robert H Lupton, Mariangela Bernardi, Michael R Blanton, Scott Burles, Francisco J Castander, AJ Connolly, Daniel J Eisenstein, Joshua A Frieman, GS Hennessy, et al. Sloan digital sky survey: early data release. *The Astronomical Journal*, 123(1):485, 2002.

Nao Suzuki. Quasar spectrum classification with principal component analysis (PCA): Emission lines in the Lyα forest. *The Astrophysical Journal Supplement Series*, 163 (1):110, 2006.

Robert H Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE, 2015.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1971–1979, 2014.

I. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Schoelkopf. Adagan: Boosting generative models. *arXiv preprint arXiv:1701.02386*, 2016.

Dustin Tran, Matthew D Hoffman, Rif A Saurous, Eugene Brevdo, Kevin Murphy, and David M Blei. Deep probabilistic programming. In *Proceedings of the International Conference on Learning Representations*, 2017.

Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 1998.

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

Jakob Walcher, Brent Groves, Tamás Budavári, and Daniel Dale. Fitting the integrated spectral energy distributions of galaxies. *Astrophysics and Space Science*, 331(1):1–51, 2011.

Chong Wang, Xi Chen, Alexander J Smola, and Eric P Xing. Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, pages 181–189, 2013.

Kuan-Chieh Wang and Richard Zemel. Classifying nba offensive plays using neural networks. 2016.

David H Weinberg, Romeel Dav'e, Neal Katz, and Juna A Kollmeier. The Lyman-alpha forest as a cosmological tool. *Proceedings of the 13th Annual Astrophysica Conference in Maryland*, 666, 2003.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

David Williams. *Probability with martingales*. Cambridge university press, 1991.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.

T. Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, 49:682–691, 2003.