On the Local Sensitivity of M-Estimation: Bayesian and Frequentist Applications

by

Ryan Giordano

A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Statistics and the Designated Emphasis in Communication, Computation and Statistics in the Graduate Division of the University of California, Berkeley

Committee in charge:
Professor Michael I. Jordan, Chair Professor Jon McAuliffe, Co-chair Professor Tamara Broderick Professor Martin Wainwright Professor Phillip Geissler

Spring 2019
On the Local Sensitivity of M-Estimation: Bayesian and Frequentist Applications

Copyright 2019
by
Ryan Giordano
Abstract

On the Local Sensitivity of M-Estimation: Bayesian and Frequentist Applications

by

Ryan Giordano

Doctor of Philosophy in Statistics

Designated Emphasis in Communication, Computation and Statistics

University of California, Berkeley

Professor Michael I. Jordan, Chair

Professor Jon McAuliffe, Co-chair

This thesis uses the local sensitivity of M-estimators to address a number of extant problems in Bayesian and frequentist statistics. First, by exploiting a duality from the Bayesian robustness literature between sensitivity and covariances, I provide significantly improved covariance estimates for mean field variational Bayes (MFVB) procedures at little extra computational cost. Prior to this work, applications of MFVB have arguably been limited to prediction problems rather than inference problems for lack of reliable uncertainty measures. Second, I provide practical finite-sample accuracy bounds for the “infinitesimal jackknife” (IJ), a classical measure of local sensitivity to an empirical process. In doing so, I bridge a gap between classical IJ theory and recent machine learning practice, showing that stringent classical conditions for the consistency of the IJ can be relaxed for restricted but useful classes of weight vectors, such as those of leave-$K$-out cross validation. Finally, I provide techniques to quantify the sensitivity of the inferred number of clusters in Bayesian nonparametric (BNP) unsupervised clustering problems to the form of the Dirichlet process prior. By considering local sensitivity to be an approximation to global sensitivity rather than a measure of robustness per se, I provide tools with considerably improved ability to extrapolate to different priors. Because each of these diverse applications are based on the same formal technique—the Taylor series expansion of an M-estimator—this work captures in a unified way the computational difficulties associated with each, and I provide open-source tools in Python and R to assist in their computation.
To Dora and Kai, for your sensitivity in all things.

Figure 0: A Hessian, as imagined by a humanist.
# Contents

## 1 Introduction
1.1 Local hyperparameter sensitivity ........................................ 3  
1.2 M-estimators .................................................................. 6  
1.3 Sensitivity and uncertainty .................................................. 8

## 2 Covariances and robustness
2.1 Bayesian covariances and sensitivity ................................... 12  
2.2 Variational Bayesian covariances and sensitivity ................. 16  
2.3 Calculation and uses of sensitivity .................................... 20  
2.4 Experiments ..................................................................... 30  
2.5 Proof of Theorem 2.1 ....................................................... 55  
2.6 Comparison with MCMC importance sampling ................. 56  
2.7 Our use of the terms “sensitivity” and “robustness” .............. 57  
2.8 Proof of Theorem 2.2 ....................................................... 60  
2.9 Exactness of multivariate Normal posterior means .............. 61

## 3 Exponential families
3.1 Linear response covariance estimation ............................... 66  
3.2 Discussion and comparison with alternative methods .......... 71  
3.3 Experiments ................................................................... 75

## 4 Infinitesimal jackknife
4.1 Methods and results .......................................................... 84  
4.2 Examples ....................................................................... 90  
4.3 Related work .................................................................. 91  
4.4 Simulated experiments ...................................................... 93  
4.5 Genomics experiments ...................................................... 93  
4.6 Detailed assumptions, lemmas, and proofs ....................... 97


**Acknowledgments**

I am grateful to more people, circumstances, and institutions than I could possibly enumerate, but now I have to try. This thesis began at the welcome party during the first week of the PhD program, when Jon memorably and enthusiastically first told me about variational Bayes. I have benefited enormously from his technical and personal guidance and generosity ever since. Having Mike as a model of scholarship for these years has been an extraordinary experience. The depth and breadth of topics in his reading group has inculcated in me the habit of fearlessly reading outside my own expertise, and his unshakable generosity and kindness are the model of academic mentorship. Finally, I owe more to Tamara than I can possibly repay except, perhaps, by trying to pass some small fraction of her generosity forwards. Incredibly, when Tamara was a very busy senior PhD student and I was an anonymous second-year, Tamara found the time to read and critique an early draft of the work that lead to the LRVB parts of this thesis. Since then, Tamara has been an integral part of all my academic work, to my great benefit. Of all the ways I have grown during my PhD, it is my writing that has improved the most, and I owe this improvement to Tamara’s expert and meticulous guidance.

This work was additionally informed and improved by conversations with many students within the incredibly collaborative culture of the Berkeley statistics department. Runjing Liu, in particular, played a major role running the genomics experiments in Chapter 4 and co-authoring Chapter 5 and Chapter 6.

For my parents, who always let me do my own thing, and for Dora, who has made these years the happiest of my life (and who also lets me do my own thing), no words will suffice. This work is as much yours as it is mine.
Chapter 1

Introduction

This thesis applies local sensitivity analysis to several current problems in Bayesian and frequentist statistics. We will focus on applications to M-estimators, which are straightforward generalizations of smooth optimization problems, and so encompass a wide range of contemporary machine learning and statistical applications.

The thesis can be conceptually divided into two halves. The first half, consisting of Chapter 2 and Chapter 3, is concerned with improved covariance estimation in mean field variational Bayes (MFVB). MFVB techniques are popular in machine learning for their speed and scalability [Blei et al., 2016], but are well known to underestimates posterior covariances [MacKay, 2003, Turner and Sahani, 2011, Wang and Titterington, 2004]. Inaccurate uncertainty estimates are not necessarily problematic for prediction problems, but are crucial for inference. As a consequence, interest in MFVB has been largely limited to machine learning and not statistical inference. I exploit a classical duality between local sensitivity and posterior covariances to compute improved covariance estimates from a single run of the MFVB algorithm without the need to use a more complex approximation. Chapter 2 lays out the basic principles, and Chapter 3 applies the idea to the common special case of moment-parameterized exponential families.

The second half, consisting of Chapter 4, Chapter 5, and Chapter 6, develops local sensitivity techniques within their traditional domain of application, hyperparameter sensitivity. In the frequentist chapters, Chapter 4 and Chapter 5, I provide techniques for approximate cross validation using a connection between cross validation and the classical infinitesimal jackknife (IJ). The IJ can be seen as an instance of local sensitivity measures applied to a vector of data weights [Jaeckel, 1972]. To the best of the author’s knowledge, previous work on the accuracy of the IJ has been stochastic and asymptotic in nature [Reeds, 1978, Shao, 1993, Shao and Tu, 2012, Rad and Maleki, 2018]. Furthermore, previous work has required termwise-bounded gradients (with few exceptions, notably [Reeds, 1978]), a pro-
hibitively strict condition for many machine learning applications. In Chapter 4, I develop non-stochastic, finite-sample accuracy measures for the IJ that do not require termwise-bounded gradients. Consistency of cross-validation follows as an immediate corollary of the central theorem of Chapter 4. Chapter 5 then applies the IJ to the question of evaluating cluster stability in an unsupervised learning problem in genomics.

In Chapter 6, I apply local sensitivity techniques to a classical question of Bayesian prior sensitivity (in the spirit of [Gustafson, 2000]), again in the context of unsupervised clustering. Specifically, I develop MFVB versions of classical Bayesian robustness measures in the context of Bayesian nonparametrics (BNP), including sensitivity to the functional form of a stick-breaking prior. Chapter 6 improves on previous Bayesian robustness results for BNP [Basu, 2000] both by providing sensitivity results for MFVB, which, for speed and scalability, is commonly used in practical BNP problems, as well as by linearizing only the most computationally expensive part of inference—the optimization problem—while allowing for non-linearity in the map from the optimal parameters to the predictive quantity of interest. As a result, my methods provide considerably improved ability to quickly extrapolate to new forms of the stick-breaking prior.

Every application of local sensitivity described in this thesis depends ultimately on the ability to solve a particular linear system involving the Hessian matrix of the objective function. It would be error-prone and tedious to manually compute the necessary derivatives, and this has limited applicability of local sensitivity techniques in the past [Shao and Tu, 2012, Chapter 2]. However, modern automatic differentiation obviates this difficulty [Baydin et al., 2018, Maclaurin et al., 2015]. I argue that the advent of easy-to-use, general automatic differentiation motivates revisiting classical local sensitivity techniques. In the course of running the experiments for this thesis, I have developed and released two open-source software packages in Python, paragami and vittles, to facilitate black-box sensitivity analysis. I have also developed an R package, rstansensitivity, for automatically calculating local sensitivity of Bayesian posterior expectation estimated with Markov chain Monte Carlo (MCMC) using the popular Hamiltonian Monte Carlo software Stan [Stan Team, 2015]. All three packages are unit-tested, extensively documented, and freely available on github at rgiordan/paragami, rgiordan/vittles, and rgiordan/rstansensitivity respectively.

For the remainder of the introduction, I will provide a brief survey to the key concepts explored in the thesis. Chapter 7 discusses future work.
1.1 Local hyperparameter sensitivity

I will begin by introducing the concept of local sensitivity to hyperparameters [Cook, 1986, Gustafson, 2000, Borgonovo, 2017]. The present thesis will consist entirely of applications and variants of this concept.

A typical workflow for a statistician is to gather and clean data, posit a loss function, and find model parameters that at least locally minimize the loss function on the observed data. Optimization-based procedures are most commonly used in frequentist statistics, but are also used for approximate Bayesian inference through variational Bayes techniques. It is important to know to what extent the estimated parameters are sensitive to different plausible modeling choices.

As a classical toy example, consider normally distributed IID scalar data, \( x_n \) for \( n = 1, \ldots, N \), centered at an unknown parameter \( \theta \). Specifically, letting \( x = (x_1, \ldots, x_N) \),

\[
p(x_n | \theta) = \mathcal{N}(x_n | \theta, 1) \\
p(x | \theta) = \prod_{n=1}^{N} p(x_n | \theta),
\]

where \( \mathcal{N}(\cdot | \mu, \sigma^2) \) denotes a normal distribution with mean \( \mu \) and variance \( \sigma^2 \). To estimate the unknown \( \theta \) from the observations \( x \), a Bayesian approach might be to place a normal prior on \( \theta \) with mean \( \mu \) and variance \( \sigma^2 \):

\[
p(\theta | \mu, \sigma) = \mathcal{N}(\mu, \sigma^2).
\]

Together, \( p(x | \theta) \) and \( p(\theta | \mu, \sigma) \) define a joint distribution on \( x \) and \( \theta \) conditional on \( \mu \) and \( \sigma \), i.e., \( p(x | \theta)p(\theta | \mu, \sigma) = p(x, \theta | \mu, \sigma) \). Suppose we want to provide an estimate of \( \theta \), \( \hat{\theta}(x) \), minimizing the loss function of a statistic \( t(x) \),

\[
L(t(x)) = \mathbb{E} \left[ (t(x) - \theta)^2 | \mu, \sigma \right],
\]

where the expectation is taken over the joint distribution \( p(x, \theta | \mu, \sigma) \). The minimizer of \( L(\hat{\theta}(x)) \) is known to be the Bayesian posterior mean

\[
\hat{\theta}(x) = \mathbb{E} [\theta | x, \mu, \sigma].
\]

In this particular case, by conjugacy and the symmetry of the Normal distribution, \( \hat{\theta}(x) \) also corresponds to the maximizer of the log joint distribution (i.e., the "maximum a posteriori" or MAP estimator):

\[
\hat{\theta}(x) = \arg \max_{\theta} \log p(x, \theta | \mu, \sigma). \tag{1.1}
\]
CHAPTER 1. INTRODUCTION

Whether viewed as a posterior integral or solution to an optimization problem, the loss-minimizing statistic has the same value. Let \( \bar{x} := \frac{1}{N} \sum_{n=1}^{N} x_n \) be the sample average of the observations. Then the loss-minimizing statistic is given by

\[
\hat{\theta}(x) = \frac{N \bar{x} + \sigma^{-2} \mu}{N + \sigma^{-2}}.
\] (1.2)

Although I have written \( \hat{\theta}(x) \) as a function of \( x \) to emphasize that it is a statistic, it is evident from Eq. (1.2) that it is also a function of \( \mu \) and \( \sigma \), which may be thought of either as model or loss parameters depending on the practitioner’s perspective. I will denote this when necessary as \( \hat{\theta}(x) = \hat{\theta}(x|\mu, \sigma) \). The point is that, in many analogous practical cases, parameters like \( \mu \) and \( \sigma \) might take a range of plausible values. The variation in \( \hat{\theta}(x|\mu, \sigma) \) over the plausible range of \( \mu \) and \( \sigma \) must contribute to our epistemic uncertainty in our belief about the value of the unknown \( \theta \) by any reasonable definition of uncertainty.\(^1\)

In this case, it is clear from Eq. (1.2) that, all else equal, \( \hat{\theta}(x|\mu, \sigma) \) is less sensitive to \( \mu \) and \( \sigma \) as \( N \) increases. As expected, the prior has less influence as the amount of data grows. In general, however, the dependence of an estimator on hyperparameters is not available in closed form. This thesis will be predominantly concerned with cases where the dependence is implicitly defined through the solution of an optimization problem as in Eq. (1.1), though in Chapter 2 I will also discuss the sensitivity of Monte Carlo procedures.

To quantify the uncertainty induced in \( \hat{\theta}(x|\mu, \sigma) \) by the uncertainty in \( \mu \) and \( \sigma \), one might consider quantities such as

\[
\sup_{\mu, \sigma \in A} \hat{\theta}(x|\mu, \sigma) - \inf_{\mu, \sigma \in A} \hat{\theta}(x|\mu, \sigma),
\] (1.3)

for some set \( A \) of plausible values of \( (\mu, \sigma) \) or

\[
\int_A (\hat{\theta}(x|\mu, \sigma) - \theta_0)^2 \lambda(d\mu d\sigma),
\] (1.4)

for some measure \( \lambda \) on the space of \( \mu \) and \( \sigma \) and reference value \( \theta_0 \). For example, \( \lambda \) might be \( p(\mu, \sigma|x) \) for suitably chosen priors on \( \mu \) and \( \sigma \), and \( \theta_0 \) might be the mean of \( \hat{\theta}(x|\mu, \sigma) \) under this measure.

\(^1\)I will not need to take a stand in this thesis on whether one must always use the Bayesian formalism for coherent decision-making. Suffice it to say that one must consider the sensitivity of estimators to hyperparameters somehow. Below, I will be considering one method for approximately quantifying sensitivity, namely Taylor series expansions. Precisely how these approximations are used or interpreted—as approximations of Bayesian or non-Bayesian procedures—can be left up to the practitioner.
Both Eq. (1.3) and Eq. (1.4) are instances of “global sensitivity” measures, in that they require the calculation of $\hat{\theta}(x|\mu, \sigma)$ globally over a set $\mathcal{A}$. Although easy in our simple example, in general the computation of global sensitivity measures requires either special structure or repeated evaluation of an expensive estimation procedure such as an optimization problem like Eq. (1.1).

A more computationally feasible alternative to global sensitivity is to approximate the dependence of the estimator on the hyperparameter with a Taylor expansion around some hyperparameter base value. For our example, let us fix $\mu_0 = 0$ and $\sigma_0 = 1$. Then define the linear approximation $\hat{\theta}_{lin}(x|\mu, \sigma)$ as follows:

$$
\hat{\theta}_{lin}(x|\mu, \sigma)-\hat{\theta}(x|\mu_0, \sigma_0)
:= \frac{d\hat{\theta}(x|\mu, \sigma)}{d\mu}|_{\mu_0, \sigma_0} (\mu - \mu_0) + \frac{d\hat{\theta}(x|\mu, \sigma)}{d\sigma}|_{\mu_0, \sigma_0} (\sigma - \sigma_0)
= \frac{\mu}{N + 1} + \frac{2N \bar{x}}{(N + 1)^2} (\sigma - 1).
$$

(1.5)

The linear approximation of Eq. (1.5) is an instance of a “local sensitivity” measure, so named because it can in general only be expected to hold in a region near the center of the Taylor expansion. The linear approximation Eq. (1.5) captures some salient features of our toy problem. For example, as $N$ goes to infinity, the local sensitivity goes to zero, as expected. In this case $\hat{\theta}(x|\mu, \sigma)$ depends linearly on $\mu$, and so the linear approximation is in fact exact for all $\mu$ if $\sigma$ is fixed. However, as $\sigma$ goes to 0, then we know by Eq. (1.2) that $\hat{\theta}(x|\mu, \sigma)$ approaches $\mu$, a phenomenon that is not captured by Eq. (1.5) because the dependence of $\hat{\theta}(x|\mu, \sigma)$ on $\sigma$ is highly nonlinear near 0.

It is often easier to compute the derivative for a local approximation than to compute global sensitivity measures. To the extent that linearity holds over the set $\mathcal{A}$, we can use local sensitivity to easily approximate global sensitivity by the respective variants of Eq. (1.3) and Eq. (1.4):

$$
\sup_{\mu, \sigma \in \mathcal{A}} \hat{\theta}_{lin}(x|\mu, \sigma) - \inf_{\mu, \sigma \in \mathcal{A}} \hat{\theta}_{lin}(x|\mu, \sigma)
$$

and

$$
\int_{\mathcal{A}} (\hat{\theta}_{lin}(x|\mu, \sigma) - \theta_0)^2 \lambda(d\mu d\sigma).
$$

Of course, increased computational speed comes at the cost of accuracy due to the linear approximation.
The toy example in this section is an example typical of local Bayesian robustness, which has historically mostly been concerned with sensitivity to prior specification \cite{Gustafson2000}. The situations in which local sensitivity is most useful are situations where

- The prior remains influential on the kinds of datasets used in practice,
- Evaluation or approximation of the posterior means is expensive,
- The dependence of the posterior on the prior is complicated and difficult to reason about, and
- The prior could plausibly take a wide range of values.

An example that fulfills all these desiderata is the problem of inferring observed number of distinct clusters in unsupervised clustering with Bayesian nonparametrics \cite{GershmanBlei2012}, where the prior is the stick-breaking process itself. I analyze this problem in depth in Chapter 6. In addition to considering sensitivity to scalar hyperparameters, Chapter 6 develops a variational Bayes version of local sensitivity to the functional form of the prior \cite{Gustafson2000}. I elaborate on more general results for local sensitivity of Bayesian posteriors in Chapter 2.

1.2 M-estimators

M-estimators are a general class of estimation procedures for which the estimator \( \hat{\theta} \) is defined as the root of a set of estimating equations. Specifically, for a vector-valued function \( G \), an M-estimator \( \hat{\theta} \) is defined as

\[
\hat{\theta} := \theta \text{ such that } G(\theta) = 0.
\]

So defined, smooth optimization problems are instances of M-estimators. To see this, define a smooth function \( F \), and let

\[
\hat{\theta}_{\text{opt}} := \arg\min_{\theta} F(\theta).
\]

By the first order condition, a solution \( \hat{\theta}_{\text{opt}} \) satisfies

\[
\nabla_{\theta} F(\hat{\theta}_{\text{opt}}) = 0,
\]

\footnote{Originally, M-estimators were defined as maximizers of objective functions, not as roots of estimating equations \cite{Huber1964, VanDerVaart2000}. Van der Vaart \cite{VanDerVaart2000} observes that the definition I use here might be more appropriately called a “Z-estimator.” Nevertheless, I will use the more commonly recognizable “M-estimator.”}
which is an M-estimator with $G = \nabla_{\theta} F$. Our motivation for using the more general definition is that solutions of sequences of optimization problems can be represented as M-estimators, even though the solution may not be the optimum of any single objective. For example, suppose that we estimate $\hat{\theta}_1$ and $\hat{\theta}_2$ by first solving an optimization problem for $\hat{\theta}_1$ and then using the result as a hyperparameter in an optimization problem for $\hat{\theta}_2$:

$$\hat{\theta}_1 := \arg\min_{\theta_1} F_1(\theta)$$
$$\hat{\theta}_2 := \arg\min_{\theta_2} F_2(\theta_1, \theta_2).$$  \hfill (1.6)

The solution to Eq. (1.6) can be represented as a single M-estimator by defining

$$G_1(\theta_1, \theta_2) := \nabla_{\theta_1} F_1(\theta_1) \quad \text{(No \(\theta_2\) dependence)}$$
$$G_2(\theta_1, \theta_2) := \nabla_{\theta_2} F_2(\theta_1, \theta_2)$$

and taking $G$ to be the concatenation of $G_1$ and $G_2$. However, in general, $G$ is not the gradient of any single optimization function unless $G_2$ does not depend on $\theta_1$ because, if it were, the corresponding Hessian matrix $\nabla_{\theta_1, \theta_2} G$ would not be symmetric. The use of sensitivity analysis for sequences of optimization problems is discussed and demonstrated in detail in Chapter 4.

The root of an estimating equation may not be be unique, so additional conditions must be stated for an M-estimator to be well-defined. Chapter 4 is most explicit about this point, effectively stating conditions under which the M-estimator is unique for all hyperparameter values. Chapter 5 discusses experimental evidence that local sensitivity may be of limited value for certain practical problems (e.g. unsupervised clustering) due to the possibility of non-unique solutions. The application of local sensitivity to Bayesian covariances in Chapter 2 and Chapter 3 are not materially affected by potential non-uniqueness, since it is the local sensitivity of a particular root \textit{per se} that is of interest.

An M-estimator that will be of particular interest in this thesis is the optimization problem associated with mean field variational Bayes (MFVB) [Blei et al., 2016; Wainwright and Jordan, 2008]. Variational Bayes (VB) refers to procedures that produce an approximation to a Bayesian posterior or posterior quantities through the solution of an optimization problem. Often, the optimization problem is cast as finding a distribution in some tractable class that minimizes a divergence measure to the true posterior. MFVB uses factorizing distributions as its tractable class, and typically uses Kullback-Leibler (KL) divergence as its divergence measure. More details can be found in Chapter 2, Chapter 3, and Chapter 6.

The local sensitivity of M-estimators has a closed form that can be obtained by Taylor expanding the estimating equation $G$. Suppose that an M-estimator has
CHAPTER 1. INTRODUCTION

hyperparameters $\alpha$, i.e., that

$$\hat{\theta}(\alpha) := \theta \text{ such that } G(\theta|\alpha) = 0.$$ 

Then, in considerable generality,

$$\frac{d\hat{\theta}(\alpha)}{d\alpha} \bigg|_{\alpha_0} (\alpha - \alpha_0) = \left. \frac{\partial G(\theta, \alpha)}{\partial \theta} \right|_{\hat{\theta}(\alpha_0), \alpha_0} \left. \frac{\partial G(\theta, \alpha)}{\partial \alpha} \right|_{\hat{\theta}(\alpha_0), \alpha_0} (\alpha - \alpha_0). \quad (1.7)$$

In the case of an optimization problem with objective $F$, the $\partial G/\partial \theta$ term in Eq. (1.7) becomes the objective’s Hessian matrix. Eq. (1.7) is discussed with particular care in Chapter 2 and Chapter 4 in the context of mean field variational Bayes and the infinitesimal jackknife, respectively. However, the results described therein generalize immediately to the general question of hyperparameter sensitivity for M-estimators.

1.3 Sensitivity and uncertainty

Frequentist uncertainty naturally takes the form of a sensitivity measure. As a simple example, let us consider the case where the data consists of $N$ IID observations, $x = (x_1, ..., x_N)$. Let us define a set of weights, $w = (w_1, ..., w_N)$, which we will treat as hyperparameters. We then define the weighted estimator for some smooth function $h$,

$$\hat{\theta}(x|w) = h \left( \frac{1}{N} \sum_{n=1}^{N} w_n x_n \right).$$

Letting $1_w = (1, ..., 1)$ denote the length-$N$ vector of all ones, we can define the linear approximation

$$\hat{\theta}_{lin}(x|w) := \hat{\theta}(x|1_w) + \frac{d\hat{\theta}(x|w)}{dw^T} \bigg|_{w_1} (w - w_1)$$

$$= h \bar{x} + h' \bar{x} \left( \frac{1}{N} \sum_{n=1}^{N} (w_n - 1) x_n. \right)$$

$\hat{\theta}_{lin}(x|w)$ is a linear approximation of how the estimator $\hat{\theta}$ depends on the empirical distribution. Such a linear approximation is known as the “infinitesimal jackknife” (IJ) because of its close relationship with the jackknife estimator of bias or of variance [Jaeckel 1972, Shao and Tu 2012, Efron 1982]. The IJ is closely related
to the influence function, which characterizes, among other things, the robustness
of an estimator and its asymptotic variance [Huber, 2011], and to the general func-
tional differential calculus of statistics which are functions of the empirical distribu-

Classical theory typically uses the IJ to prove asymptotic consistency or normal-
ity results, for which stringent regularity conditions are needed for differentiability
within a function space capacious enough to include the Brownian bridge [Fernholz
1983]. However, the IJ and closely related approximations have attracted recent in-
terest in machine learning as a tool for approximate cross validation (CV) [Koh and
Liang, 2017, Rad and Maleki, 2018] in applications for which the classical theory
does not apply. In particular, most work employing the IJ to analyze M-estimators
has required the objective function to have bounded gradients [Clarke, 1983, Shao
1993], an assumption that is rarely fulfilled in machine learning. In Chapter 4, I
develop finite-sample theory for the accuracy of the IJ, a corollary of which is con-
sistency of the IJ for leave-\(K\)-out CV even when the objective function’s gradients
are unbounded. Although Chapter 4 is entirely in the context of the IJ, the theory
would apply essentially without modification to the accuracy of any local sensitivity
measure.

It is, perhaps, less obvious how sensitivity relates Bayesian uncertainty. How-
ever, a classical result in Bayesian robustness is that the sensitivity of a Bayesian
posterior expectation to hyperparameters is given by a particular covariance. This
equivalence applies in great generality whenever one can exchange integration and
differentiation; see Chapter 2 for a detailed exposition. As an example of this result,
consider again the toy example of Section 1.1. It turns out that

\[
\frac{d\mathbb{E}[\theta|x, \mu, \sigma_0 = 1]}{d\mu} \bigg|_{\mu_0 = 0} = \left. \text{Cov} \left( \theta, \frac{\partial \log p(\theta, x|\mu, \sigma_0 = 1)}{\partial \mu} \right) \right|_{\mu_0 = 0} = \left. \text{Cov} \left( \theta, \sigma_0^{-2} \theta \right) \right|_{x, \mu_0 = 0, \sigma_0 = 1} = \frac{1}{N + 1},
\]

matching (necessarily) the result given in Eq. (1.5). In general, one does not have
access to a closed form expression for posterior means as a function of hyperpa-
rameters as we do in our toy example. However, if one can estimate posterior
expectations, e.g., by using draws from a Markov chain Monte Carlo procedure,
one can in turn estimate hyperparameter sensitivity using the identity above.

A central idea in this thesis is to turn this idea on its head and use the sensitivity
of M-estimators to estimate covariances for MFVB. Sensitivity-based covariance
estimators have long been known in statistical mechanics and machine learning as “linear response covariances.” Chapter 2 contains a brief literature review. Recall from Section 1.1 the equivalence between the MAP estimator solution to the optimization problem Eq. (1.1) and the posterior mean $\mathbb{E}[\theta|x, \mu, \sigma]$. Unlike generic Bayesian posterior means, M-estimators do have closed form sensitivity measures, as discussed in Section 1.2. The MAP estimator, however, does not come with an inherent estimate of variance. However, an application of Eq. (1.8) provides an estimate of $\sigma_0^{-2} \text{Var}(\theta|x, \mu_0, \sigma_0)$ through the sensitivity of the MAP to perturbation of the hyperparameter $\mu$. By choosing appropriate perturbations, we can in principle use this technique to estimate any covariance we choose.

For the MAP estimator, this technique of estimating sensitivities from covariances turns out to be equivalent to the Laplace approximation, as discussed in Chapter 2. The Laplace covariance approximation is commonly motivated by a quadratic approximation to the log posterior, which does not generalize in any obvious way to more complex VB approximations. The sensitivity-based covariance estimation approach, in contrast, generalizes to any VB procedure. In particular, we show in Chapter 2 and Chapter 3 that, when applied to MFVB, this technique can give significantly improved covariance estimates without sacrificing the speed and simplicity of MFVB.
Chapter 2

Covariances, robustness, and variational Bayes

Most Bayesian posteriors cannot be calculated analytically, so in practice we turn to approximations. Variational Bayes (VB) casts posterior approximation as an optimization problem in which the objective to be minimized is the divergence, among a sub-class of tractable distributions, from the exact posterior. For example, one widely-used and relatively simple flavor of VB is “mean field variational Bayes” (MFVB), which employs Kullback-Leibler (KL) divergence and a factorizing exponential family approximation for the tractable sub-class of posteriors [Wainwright and Jordan, 2008]. MFVB has been increasingly popular as an alternative to Markov Chain Monte Carlo (MCMC) in part due to its fast runtimes on large-scale data sets. Although MFVB does not come with any general accuracy guarantees (except asymptotic ones in special cases [Westling and McCormick, 2017, Wang and Blei [2018]), MFVB produces posterior mean estimates of certain parameters that are accurate enough to be useful in a number of real-world applications [Blei et al., 2016]. Despite this ability to produce useful point estimates for large-scale data sets, MFVB is limited as an inferential tool: in particular, MFVB typically underestimates marginal variances [MacKay 2003, Wang and Titterington 2004, Turner and Sahani 2011]. Moreover, to the best of our knowledge, techniques for assessing Bayesian robustness have not yet been developed for MFVB. It is these inferential issues that are the focus of the current paper.

Unlike the optimization approach of VB, an MCMC posterior estimate is an empirical distribution formed with posterior draws. MCMC draws lend themselves naturally to the approximate calculation of posterior moments, such as those required for covariances. In contrast, VB approximations lend themselves naturally to sensitivity analysis, since we can analytically differentiate the optima with respect to perturbations. However, as has long been known in the Bayesian robust-
ness literature, the contrast between derivatives and moments is not so stark since, under mild regularity conditions that allow the exchange of integration and differentiation, there is a direct correspondence between derivatives and covariance [Gustafson [1996b], Basu et al. [1996], Efron [2015], Section 2.1 below].

Thus, in order to calculate local sensitivity to model hyperparameters, the Bayesian robustness literature re-casts derivatives with respect to hyperparameters as posterior covariances that can be calculated with MCMC. In order to provide covariance estimates for MFVB, we turn this idea on its head and use the sensitivity of MFVB posterior expectations to estimate their covariances. These sensitivity-based covariance estimates are referred to as “linear response” estimates in the statistical mechanics literature [Opper and Saad [2001], so we refer to them here as linear response variational Bayes (LRVB) covariances. Additionally, we derive straightforward MFVB versions of hyperparameter sensitivity measures from the Bayesian robustness literature. Under the assumption that the posterior means of interest are well-estimated by MFVB for all the perturbations of interest, we establish that LRVB provides a good estimate of local sensitivities. In our experiments, we compare LRVB estimates to MCMC, MFVB, and Laplace posterior approximations. We find that the LRVB covariances, unlike the MFVB and Laplace approximations, match the MCMC approximations closely while still being computed over an order of magnitude more quickly than MCMC.

In Section 2.1 we first discuss the general relationship between Bayesian sensitivity and posterior covariance and then define local robustness and sensitivity. Next, in Section 2.2 we introduce VB and derive the linear system for the MFVB local sensitivity estimates. In Section 2.3 we show how to use the MFVB local sensitivity results to estimate covariances and calculate canonical Bayesian hyperparameter sensitivity measures. Finally, in Section 2.4 we demonstrate the speed and effectiveness of our methods with simple simulated data, an application of automatic differentiation variational inference (ADVI), and a large-scale industry data set.

2.1 Bayesian covariances and sensitivity

Local sensitivity and robustness

Denote an unknown model parameter by the vector \( \theta \in \mathbb{R}^K \), assume a dominating measure for \( \theta \) on \( \mathbb{R}^K \) given by \( \lambda \), and denote observed data by \( x \). Suppose that we have a vector-valued hyperparameter \( \alpha \in \mathcal{A} \subseteq \mathbb{R}^D \) that parameterizes some aspects of our model. For example, \( \alpha \) might represent prior parameters, in which case we would write the prior density with respect to \( \lambda \) as \( p(\theta|\alpha) \), or it might parameterize
a class of likelihoods, in which case we could write the likelihood as $p(x|\theta, \alpha)$. Without loss of generality, we will include $\alpha$ in the definition of both the prior and likelihood. For the moment, let $p_\alpha(\theta)$ denote the posterior density of $\theta$ given $x$ and $\alpha$, as given by Bayes’ Theorem (this definition of $p_\alpha(\theta)$ will be a special case of the more general Definition 2.2 below):

$$p_\alpha(\theta) := p(\theta|x, \alpha) = \frac{p(x|\theta, \alpha) p(\theta|\alpha)}{\int p(x|\theta', \alpha) p(\theta'|\alpha) \lambda(d\theta')} = \frac{p(x|\theta, \alpha) p(\theta|\alpha)}{p(x|\alpha)}.$$

We will assume that we are interested in a posterior expectation of some function $g(\theta)$ (e.g., a parameter mean, a posterior predictive value, or squared loss):

$$E_{p_\alpha}[g(\theta)].$$

In the current work, we will quantify the uncertainty of $g(\theta)$ by the posterior variance, $\text{Var}_{p_\alpha}(g(\theta))$. Other measures of central tendency (e.g., posterior medians) or uncertainty (e.g., posterior quantiles) may also be good choices but are beyond the scope of the current work.

Note the dependence of $E_{p_\alpha}[g(\theta)]$ on both the likelihood and prior, and hence on $\alpha$, through Bayes’ Theorem. The choice of a prior and choice of a likelihood are made by the modeler and are almost invariably a simplified representation of the real world. The choices are therefore to some extent subjective, and so one hopes that the salient aspects of the posterior would not vary under reasonable variation in either choice. Consider the prior, for example. The process of prior elicitation may be prohibitively time-consuming; two practitioners may have irreconcilable subjective prior beliefs, or the model may be so complex and high-dimensional that humans cannot reasonably express their prior beliefs as formal distributions. All of these circumstances might give rise to a range of reasonable prior choices. A posterior quantity is “robust” to the prior to the extent that it does not change much when calculated under these different prior choices.

Quantifying the sensitivity of the posterior to variation in the likelihood and prior is one of the central concerns of the field of robust Bayes [Berger et al., 2000]. (We will not discuss the other central concern, which is the selection of priors and likelihoods that lead to robust estimators.) Suppose that we have determined that the hyperparameter $\alpha$ belongs to some open set $\mathcal{A}$, perhaps after expert prior elicitation. Ideally, we would calculate the extrema of $E_{p_\alpha}[g(\theta)]$ as $\alpha$ ranges over all of $\mathcal{A}$. These extrema are a measure of global robustness, and their calculation is intractable or difficult except in special cases [Moreno, 2000; Huber, 2011, Chapter 15]. A more practical alternative is to examine how much $E_{p_\alpha}[g(\theta)]$ changes locally in response to small perturbations in the value of $\alpha$ near some tentative guess, $\alpha_0 \in \mathcal{A}$. To this end we define the local sensitivity at $\alpha_0$ [Gustafson, 2000].
Definition 2.1. The local sensitivity of $E_{p_\alpha} [g(\theta)]$ to hyperparameter $\alpha$ at $\alpha_0$ is given by

$$S_{\alpha_0} := \left. \frac{dE_{p_\alpha} [g(\theta)]}{d\alpha} \right|_{\alpha_0}.$$  \hspace{1cm} (2.1)$$

$S_{\alpha_0}$, the local sensitivity, can be considered a measure of local robustness [Gustafson 2000]. Throughout the paper we will distinguish between sensitivity, which comprises objectively defined quantities such as $S_{\alpha_0}$, and robustness, which we treat as a more subjective concept that may be informed by the sensitivity as well as other considerations. For example, even if one knows $S_{\alpha_0}$ precisely, how much posterior change is too much change and how much prior variation is reasonable remain decisions to be made by the modeler. For a more in-depth discussion of how we use the terms sensitivity and robustness, see Section 2.7.

The quantity $S_{\alpha_0}$ can be interpreted as measuring sensitivity to hyperparameters within a small region near $\alpha = \alpha_0$ where the posterior dependence on $\alpha$ is approximately linear. Then local sensitivity provides an approximation to global sensitivity in the sense that, to first order,

$$E_{p_\alpha} [g(\theta)] \approx E_{p_{\alpha_0}} [g(\theta)] + S_{\alpha_0}^T (\alpha - \alpha_0).$$

Generally, the dependence of $E_{p_\alpha} [g(\theta)]$ on $\alpha$ is not given in any closed form that is easy to differentiate. However, as we will now see, the derivative $S_{\alpha_0}$ is equal, under mild regularity conditions, to a particular posterior covariance that can easily be estimated with MCMC draws.

Covariances and sensitivity

We will first state a general result relating sensitivity and covariance and then apply it to our specific cases of interest as they arise throughout the paper, beginning with the calculation of $S_{\alpha_0}$ from Section 2.1. Consider a general base density $p_0(\theta)$ defined relative to $\lambda$ and define $\rho(\theta, \alpha)$ to be a $\lambda$-measurable log perturbation function that depends on $\alpha \in A \subseteq \mathbb{R}^D$. We will require the following mild technical assumption:

Assumption 2.1. For all $\alpha \in A$, $\rho(\theta, \alpha)$ is continuously differentiable with respect to $\alpha$, and, for a given $\lambda$-measurable $g(\theta)$ there exist $\lambda$-integrable functions $f_0(\theta)$ and $f_1(\theta)$ such that $|p_0(\theta) \exp(\rho(\theta, \alpha)) g(\theta)| < f_0(\theta)$ and $|p_0(\theta) \exp(\rho(\theta, \alpha))| < f_1(\theta)$.

Under Assumption 2.1 we can normalize the log-perturbed quantity $p_0(\theta) \exp(\rho(\theta, \alpha))$ to get a density in $\theta$ with respect to $\lambda$. 

**Definition 2.2.** Denote by \( p_\alpha (\theta) \) the normalized posterior given \( \alpha \):

\[
p_\alpha (\theta) := \frac{p_0 (\theta) \exp (\rho (\theta, \alpha))}{\int p_0 (\theta') \exp (\rho (\theta', \alpha)) \, \lambda (d\theta')}.
\]  

(2.2)

For example, \( p_\alpha (\theta) \) defined in Section 2.1 is equivalent to taking \( p_0 (\theta) = p (\theta | x, \alpha_0) \) and \( \rho (\theta, \alpha) = \log p (x | \theta, \alpha) + \log p (\theta | \alpha) - \log p (x | \theta, \alpha_0) - \log p (\theta | \alpha_0) \).

For a \( \lambda \)-measurable function \( g (\theta) \), consider differentiating the expectation \( \mathbb{E}_{p_\alpha} [g (\theta)] \) with respect to \( \alpha \):

\[
\frac{d}{d\alpha} \mathbb{E}_{p_\alpha} [g (\theta)] := \frac{d}{d\alpha} \int p_\alpha (\theta) g (\theta) \lambda (d\theta).
\]  

(2.3)

When evaluated at some \( \alpha_0 \in \mathcal{A} \), this derivative measures the local sensitivity of \( \mathbb{E}_{p_\alpha} [g (\theta)] \) to the index \( \alpha \) at \( \alpha_0 \). Define \( \mathcal{A}_0 \subseteq \mathcal{A} \) to be an open ball containing \( \alpha_0 \). Under Assumption 2.1 we assume without loss of generality that \( \rho (\theta, \alpha_0) \equiv 0 \) so that \( p_0 (\theta) = p_{\alpha_0} (\theta) \); if \( \rho (\theta, \alpha_0) \) is non-zero, we can simply incorporate it into the definition of \( p_0 (\theta) \). Then, under Assumption 2.1, the derivative in Eq. (2.3) is equivalent to a particular posterior covariance.

**Theorem 2.1.** Under Assumption 2.1,

\[
\frac{d}{d\alpha} \mathbb{E}_{p_\alpha} [g (\theta)] \bigg|_{\alpha_0} = \text{COV}_{p_0} \left( g (\theta), \frac{\partial \rho (\theta, \alpha)}{\partial \alpha} \bigg|_{\alpha_0} \right).
\]  

(2.4)

Theorem 2.1 is a straightforward consequence of the Lebesgue dominated convergence theorem; see Section 2.5 for a detailed proof. Versions of Theorem 2.1 have appeared many times before; e.g., Diaconis and Freedman [1986], Basu et al. [1996], Gustafson [1996b], Pérez et al. [2006] have contributed variants of this result to the robustness literature.

By using MCMC draws from \( p_0 (\theta) \) to calculate the covariance on the right-hand side of Eq. (2.4), one can form an estimate of \( \frac{d}{d\alpha} \mathbb{E}_{p_\alpha} [g (\theta)] / d\alpha^\top \) at \( \alpha = \alpha_0 \). One might also approach the problem of calculating \( \frac{d}{d\alpha} \mathbb{E}_{p_\alpha} [g (\theta)] / d\alpha^\top \) using importance sampling as follows [Owen, 2013, Chapter 9]. First, an importance sampling estimate of the dependence of \( \mathbb{E}_{p_\alpha} [g (\theta)] \) on \( \alpha \) can be constructed with weights that depend on \( \alpha \). Then, differentiating the weights with respect to \( \alpha \) provides a sample-based estimate of \( \frac{d}{d\alpha} \mathbb{E}_{p_\alpha} [g (\theta)] / d\alpha^\top \). We show in Section 2.6 that this importance sampling approach is equivalent to using MCMC samples to estimate the covariance in Theorem 2.1.

An immediate corollary of Theorem 2.1 allows us to calculate \( S_{\alpha_0} \) as a covariance.
Corollary 2.1. Suppose that Assumption 2.1 holds for some \( \alpha_0 \in A \), some \( g(\theta) \), and for
\[
\rho(\theta, \alpha) = \log p(x|\theta, \alpha) + \log p(\theta|\alpha) - \log p(x|\theta, \alpha_0) - \log p(\theta|\alpha_0).
\]
Then Theorem 2.1 implies that
\[
S_{\alpha_0} = \text{Cov}_{p_0} \left( g(\theta), \left. \frac{\partial \rho(\theta, \alpha)}{\partial \alpha} \right|_{\alpha_0} \right). \tag{2.5}
\]

Corollary 2.1 can be found in Basu et al. [1996], in which a version of Corollary 2.1 is stated in the proof of their Theorem 1, as well as in Pérez et al. [2006] and Efron [2015]. Note that the definition of \( \rho(\theta, \alpha) \) does not contain any normalizing constants and so can typically be easily calculated. Given \( N_s \) MCMC draws \( \{\theta_n\}_{n=1}^{N_s} \) from a chain that we assume to have reached equilibrium at the stationary distribution \( p_0(\theta) \), one can calculate an estimate of \( S_{\alpha_0} \) using the sample covariance version of Eq. (2.4):
\[
\hat{S}_{\alpha_0} := \frac{1}{N_s} \sum_{n=1}^{N_s} g(\theta_n) \left. \frac{\partial \rho(\theta_n, \alpha)}{\partial \alpha} \right|_{\alpha_0} - \left( \frac{1}{N_s} \sum_{n=1}^{N_s} g(\theta_n) \right) \left( \frac{1}{N_s} \sum_{n=1}^{N_s} \left. \frac{\partial \rho(\theta_n, \alpha)}{\partial \alpha} \right|_{\alpha_0} \right) \tag{2.6}
\]
for \( \theta_n \sim p_0(\theta) \), where \( n = 1, \ldots, N_s \).

### 2.2 Variational Bayesian covariances and sensitivity

**Variational Bayes**

We briefly review variational Bayes and state our key assumptions about its accuracy. We wish to find an approximate distribution, in some class \( \mathcal{Q} \) of tractable distributions, selected to minimize the Kullback-Leibler divergence (KL divergence) between \( q \in \mathcal{Q} \) and the exact log-perturbed posterior \( p_\alpha \). We assume that distributions in \( \mathcal{Q} \) are parameterized by a finite-dimensional parameter \( \eta \) in some feasible set \( \Omega_\eta \subseteq \mathbb{R}^{K_\eta} \).

**Definition 2.3.** The approximating variational family is given by
\[
\mathcal{Q} := \{ q : q = q(\theta; \eta) \text{ for } \eta \in \Omega_\eta \} \tag{2.7}
\]
CHAPTER 2. COVARIANCES AND ROBUSTNESS

Given $Q$, we define the optimal $q \in Q$, which we call $q_\alpha(\theta)$, as the distribution that minimizes the KL divergence $KL(q(\theta; \eta) || p_\alpha(\theta))$ from $p_\alpha(\theta)$. We denote the corresponding optimal variational parameters as $\eta^*$.

**Definition 2.4.** The variational approximation $q_\alpha(\theta)$ to $p_\alpha(\theta)$ is defined by

$$ q_\alpha(\theta) := q(\theta; \eta^*) := \text{argmin}_{q \in Q} \{ KL(q(\theta; \eta) || p_\alpha(\theta)) \}, \quad (2.8) $$

where

$$ KL(q(\theta; \eta) || p_\alpha(\theta)) = \mathbb{E}_{q(\theta; \eta)} [\log q(\theta; \eta) - \log p_\alpha(\theta)]. $$

In the KL divergence, the (generally intractable) normalizing constant for $p_\alpha(\theta)$ does not depend on $q(\theta)$ and so can be neglected when optimizing. In order for the KL divergence to be well defined, we assume that both $p_\alpha(\theta)$ and $q(\theta)$ are given with respect to the same base measure, $\lambda$, and that the support of $q(\theta)$ is contained in the support of $p_\alpha(\theta)$. We will require some additional mild regularity conditions in Section 2.2 below.

A common choice for the approximating family $Q$ in Eq. (2.7) is the “mean field family” [Wainwright and Jordan, 2008, Blei et al., 2016],

$$ Q_{mf} := \left\{ q(\theta) : q(\theta) = \prod_k q_k(\theta_k; \eta_k) \right\}, \quad (2.9) $$

where $k$ indexes a partition of the full vector $\theta$ and of the parameter vector $\eta$. That is, $Q_{mf}$ approximates the posterior $p_\alpha(\theta)$ as a distribution that factorizes across sub-components of $\theta$. This approximation is commonly referred to as “MFVB,” for “mean field variational Bayes.” Note that, in general, each function $q_k(\theta_k; \eta_k)$ in the product is different. For notational convenience we write $q_k(\theta_k; \eta_k)$ instead of $q_k(\theta_k; \eta_k)$ when the arguments make it clear which function we are referring to, much as the same symbol $p$ is used to refer to many different probability distributions without additional indexing.

One may additionally assume that the components $q_k(\theta_k; \eta_k)$ are in a convenient exponential family. Although the exponential family assumption does not in general follow from a factorizing assumption, for compactness we will refer to both the factorization and the exponential family assumption as MFVB.

In an MFVB approximation, $\Omega_{\eta}$ could be a stacked vector of the natural parameters of the exponential families, or the moment parameterization, or perhaps a transformation of these parameters into an unconstrained space (e.g., the entries of the log-Cholesky decomposition of a positive definite information matrix). For more concrete examples, see Section 2.4. Although all of our experiments and much of our motivating intuition will use MFVB, our results extend to other choices of $Q$ that satisfy the necessary assumptions.
Variational Bayes sensitivity

Just as MCMC approximations lend themselves to moment calculations, the variational form of VB approximations lends itself to sensitivity calculations. In this section we derive the sensitivity of VB posterior means to generic perturbations—a VB analogue of Theorem 2.1. In Section 2.3 we will choose particular perturbations to calculate VB prior sensitivity and, through Theorem 2.1, posterior covariances.

In Definition 2.4, the variational approximation is a function of $\alpha$ through the optimal parameters $\eta^*(\alpha)$, i.e., $q_\alpha(\theta) = q(\theta, \eta^*(\alpha))$. In turn, the posterior expectation $E_{q_\alpha}[g(\theta)]$ is also a function of $\alpha$, and its derivative at $\alpha_0$—the local sensitivity of the variational approximation to $\alpha$—has a closed form under the following mild technical conditions. As with $p_0$, define $q_0 := q_{\alpha_0}$, and define $\eta^*_0 := \eta^*(\alpha_0)$.

All the following assumptions are intended to hold for a given $p_\alpha(\theta)$, approximating class $Q$, $\lambda$-measurable function $g(\theta)$, and to hold for all $\alpha \in A_0$ and all $\eta$ in an open neighborhood of $\eta_0^*$.

**Assumption 2.2.** The KL divergence at $KL(q(\theta; \eta) || p_0(\theta))$ and expected log perturbation $E_{q(\theta; \eta)}[\rho(\theta, \alpha)]$ are twice continuously differentiable in $\eta$ and $\alpha$.

**Assumption 2.3.** There exists a strict local minimum, $\eta^*(\alpha)$, of $KL(q(\theta; \eta) || p_\alpha(\theta))$ in Eq. (2.8) such that $\eta^*(\alpha)$ is interior to $\Omega_\eta$.

**Assumption 2.4.** The expectation $E_{q(\theta; \eta)}[g(\theta)]$ is a continuously differentiable function of $\eta$.

We define the following quantities for notational convenience.

**Definition 2.5.** Define the following derivatives of variational expectations evaluated at the optimal parameters:

$$H_{\eta\eta} := \left. \frac{\partial^2 KL(q(\theta; \eta) || p_0(\theta))}{\partial \eta \partial \eta^\top} \right|_{\eta = \eta_0^*},$$

$$f_{\alpha\eta} := \left. \frac{\partial^2 E_{q(\theta; \eta)}[\rho(\theta, \alpha)]}{\partial \alpha \partial \eta^\top} \right|_{\eta = \eta_0^*, \alpha = \alpha_0},$$

$$g_\eta := \left. \frac{\partial E_{q(\theta; \eta)}[g(\theta)]}{\partial \eta^\top} \right|_{\eta = \eta_0^*}.$$
Theorem 2.2. Consider a variational approximation $q_\alpha (\theta)$ to $p_\alpha (\theta)$ as given in Definition 2.4 and a $\lambda$-measurable function $g (\theta)$. Then, under Assumptions 2.1–2.4, using the definitions given in Definition 2.5, we have

$$\frac{dE_{q_\alpha} [g (\theta)]}{d\alpha^{-\top}} \bigg|_{\alpha_0} = \eta H^{-1}_{\eta \eta^{-\top}} f_{\alpha \eta}.$$  \hspace{1cm} (2.10)

A proof of Theorem 2.2 is given in Section 2.8. As with Theorem 2.1, by choosing the appropriate $\rho (\theta, \alpha)$ and evaluating $f_{\alpha \eta}$, we can use Theorem 2.2 to calculate the exact sensitivity of VB solutions to any arbitrary local perturbations that satisfy the regularity conditions. Assumptions 2.1–2.4 are typically not hard to verify. For an example, see Section 2.9, where we establish Assumptions 2.1–2.4 for a multivariate normal target distribution and a mean-field approximation.

Eq. (2.10) is formally similar to frequentist sensitivity estimates. For example, the pioneering paper of Cook [1986] contains a formula for assessing the curvature of a marginal likelihood surface [Cook, 1986, Equation 15] that, like our Theorem 2.2, represents the sensitivity as a linear system involving the Hessian of an objective function at its optimum. The geometric interpretation of local robustness suggested by Cook [1986] has been extended to Bayesian settings (see, for example, Zhu et al. [2007, 2011]). In addition to generality, one attractive aspect of their geometric approach is its invariance to parameterization. Investigating geometric interpretations of the present work may be an interesting avenue for future research.

Approximating with variational Bayes

Recall that we are ultimately interested in $E_{p_\alpha} [g (\theta)]$. Variational approximations and their sensitivity measures will be useful to the extent that both the variational means and sensitivities are close to the exact means and sensitivities. We formalize these desiderata as follows.

Condition 2.1. Under Assumptions 2.1–2.4 and the quantities defined therein, we additionally have, for all $\alpha \in \mathcal{A}$,

$$\mathbb{E}_{q_\alpha} [g (\theta)] \approx \mathbb{E}_{p_\alpha} [g (\theta)] \quad \text{and} \quad \frac{d\mathbb{E}_{q_\alpha} [g (\theta)]}{d\alpha^{-\top}} \bigg|_{\alpha_0} \approx \frac{d\mathbb{E}_{p_\alpha} [g (\theta)]}{d\alpha^{-\top}} \bigg|_{\alpha_0}.$$  \hspace{1cm} (2.11)

(2.12)

We will not attempt to be precise about what we mean by the “approximately equal” sign, since we are not aware of any practical tools for evaluating quantitatively whether Condition 2.1 holds other than running both VB and MCMC (or
some other slow but accurate posterior approximation) and comparing the results. However, VB has been useful in practice to the extent that Condition 2.1 holds true for at least some parameters of interest. We provide some intuition for when Condition 2.1 might hold in Section 2.4, and will evaluate Condition 2.1 in each of our experiments below by comparing the VB and MCMC posterior approximate means and sensitivities.

Since Condition 2.1 holds only for a particular choice of \( g(\theta) \), it is weaker than the assumption that \( q_{\alpha} \) is close to \( p_{\alpha} \) in KL divergence, or even that all the posterior means are accurately estimated. For example, as discussed in Appendix B of Giordano et al. [2015] and in Section 10.1.2 of Bishop [2006], a mean-field approximation to a multivariate normal posterior produces inaccurate covariances and may have an arbitrarily bad KL divergence from \( p_{\alpha} \), but Condition 2.1 holds exactly for the location parameters. We discuss the multivariate normal example further in Section 2.3 and Section 2.4 below.

### 2.3 Calculation and uses of sensitivity

In this section, we discuss two applications of Theorem 2.1 and Theorem 2.2: calculating improved covariance estimates and prior sensitivity measures for MFVB. Throughout this section, we will assume that we can apply Theorem 2.1 and Theorem 2.2 unless stated otherwise.

**Covariances for variational Bayes**

Consider the mean field approximating family, \( Q_{mf} \), from Section 2.2 and a fixed exact posterior \( p_{\theta} \). It is well known that the resulting marginal variances also tend to be under-estimated even when parameters means are well-estimated (see, e.g., MacKay [2003], Wang and Titterington [2004], Turner and Sahani [2011], Bishop [2006], Chapter 10). Even more obviously, any \( q \in Q_{mf} \) yields zero as its estimate of the covariance between sub-components of \( \theta \) that are in different factors of the mean field approximating family. It is therefore unreasonable to expect that \( \text{Cov}_{q_{\theta}}(g(\theta)) \approx \text{Cov}_{p_{\theta}}(g(\theta)) \). However, if Condition 2.1 holds, we may expect the sensitivity of MFVB means to certain perturbations to be accurate by Condition 2.1 and, by Theorem 2.1, we expect the corresponding covariances to be accurately estimated by the MFVB sensitivity. In particular, by taking \( \rho(\theta, \alpha) = \alpha^{T} g(\theta) \) and \( \alpha_{0} = 0 \), we have by Condition 2.1 that

\[
\frac{dE_{q_{\theta}}[g(\theta)]}{d\alpha^{T}} \bigg|_{\alpha=0} \approx \frac{dE_{p_{\theta}}[g(\theta)]}{d\alpha^{T}} \bigg|_{\alpha=0} = \text{Cov}_{p_{\theta}}(g(\theta)) .
\]  

(2.13)
We can consequently use Theorem 2.2 to provide an estimate of $\text{Cov}_{p_0}(g(\theta))$ that may be superior to $\text{Cov}_{q_0}(g(\theta))$. With this motivation in mind, we make the following definition.

**Definition 2.6.** The linear response variational Bayes (LRVB) approximation, $\text{Cov}_{LR}^{q_0}(g(\theta))$, is given by

$$\text{Cov}_{LR}^{q_0}(g(\theta)) := g_\eta H^{-1}_{\eta\eta} g_\eta^\top. \quad (2.14)$$

**Corollary 2.2.** For a given $p_0(\theta)$, class $Q$, and function $g(\theta)$, when Assumptions 2.1–2.4 and Condition 2.7 hold for $\rho(\theta, \alpha) = \alpha^\top g(\theta)$ and $\alpha_0 = 0$, then

$$\text{Cov}_{LR}^{q_0}(g(\theta)) \approx \text{Cov}_{p_0}(g(\theta)).$$

The strict optimality of $\eta_0^*$ in Assumption 2.3 guarantees that $H_{\eta\eta}$ will be positive definite and symmetric, and, as desired, the covariance estimate $\text{Cov}_{LR}^{q_0}(g(\theta))$ will be positive semidefinite and symmetric. Since the optimal value of every component of $\mathbb{E}_{q_0}[g(\theta)]$ may be affected by the log perturbation $\alpha^\top g(\theta)$, $\text{Cov}_{LR}^{q_0}(g(\theta))$ can estimate non-zero covariances between elements of $g(\theta)$ even when they have been partitioned into separate factors of the mean field approximation.

Note that $\text{Cov}_{LR}^{q_0}(g(\theta))$ and $\text{Cov}_{q_0}(g(\theta))$ differ only when there are at least some moments of $p_0$ that $q_0$ fails to accurately estimate. In particular, if $q_0$ provided a good approximation to $p_\alpha$ for both the first and second moments of $g(\theta)$, then we would have $\text{Cov}_{LR}^{q_0}(g(\theta)) \approx \text{Cov}_{q_0}(g(\theta))$ since, for $q_0$ and $p_0$,

$$\mathbb{E}_{q_0}[g(\theta)] \approx \mathbb{E}_{p_0}[g(\theta)] \quad \text{and}$$

$$\mathbb{E}_{q_0}[g(\theta) g(\theta)^\top] \approx \mathbb{E}_{p_0}[g(\theta) g(\theta)^\top] \Rightarrow$$

$$\text{Cov}_{q_0}(g(\theta)) \approx \text{Cov}_{p_0}(g(\theta)),$$

and, for $q_\alpha$ and $p_\alpha$,

$$\mathbb{E}_{q_\alpha}[g(\theta)] \approx \mathbb{E}_{p_\alpha}[g(\theta)] \Rightarrow$$

$$\text{Cov}_{LR}^{q_\alpha}(g(\theta)) \approx \text{Cov}_{p_\alpha}(g(\theta)).$$

Putting these two approximate equalities together, we see that, when the first and second moments of $q_\alpha$ approximately match those of $p_\alpha$,

$$\text{Cov}_{q_0}(g(\theta)) \approx \text{Cov}_{LR}^{q_0}(g(\theta)).$$

However, in general, $\text{Cov}_{LR}^{q_0}(g(\theta)) \neq \text{Cov}_{q_0}(g(\theta))$. In this sense, any discrepancy between $\text{Cov}_{LR}^{q_0}(g(\theta))$ and $\text{Cov}_{q_0}(g(\theta))$ indicates an inadequacy of the variational approximation for at least the second moments of $g(\theta)$. 


Let us consider a simple concrete illustrative example which will demonstrate both how $\text{Cov}_{q_0}(g(\theta))$ can be a poor approximation to $\text{Cov}_{p_0}(g(\theta))$ and how $\text{Cov}^{LR}_{q_0}(g(\theta))$ can improve the approximation for some moments but not others.

Suppose that the exact posterior is a bivariate normal,

$$p_0(\theta) = \mathcal{N}(\theta | \mu, \Sigma),$$

(2.15)

where $\theta = (\theta_1, \theta_2)^T$, $\mu = (\mu_1, \mu_2)^T$, $\Sigma$ is invertible, and $\Lambda := \Sigma^{-1}$. One may think of $\mu$ and $\Sigma$ as known functions of $x$ via Bayes’ theorem, for example, as given by a normal-normal conjugate model. Suppose we use the MFVB approximating family $Q_{mf} = \{q(\theta) : q(\theta) = q(\theta_1) q(\theta_2)\}$.

One can show (see Section 2.9) that the optimal MFVB approximation to $p_0$ in the family $Q_{mf}$ is given by

$$q_0(\theta_1) = \mathcal{N}(\theta_1 | \mu_1, \Lambda_{11}^{-1})$$

$$q_0(\theta_2) = \mathcal{N}(\theta_2 | \mu_2, \Lambda_{22}^{-1}).$$

Note that the posterior mean of $\theta_1$ is exactly estimated by the MFVB procedure:

$$\mathbb{E}_{q_0}[\theta_1] = \mu_1 = \mathbb{E}_{p_0}[\theta_1].$$

However, if $\Sigma_{12} \neq 0$, then $\Lambda_{11}^{-1} < \Sigma_{11}$, and the variance of $\theta_1$ is underestimated. It follows that the expectation of $\theta_1^2$ is not correctly estimated by the MFVB procedure:

$$\mathbb{E}_{q_0}[\theta_1^2] = \mu_1^2 + \Lambda_{11}^{-1} < \mu_1^2 + \Sigma_{11} = \mathbb{E}_{p_0}[\theta_1^2].$$

An analogous statement holds for $\theta_2$. Of course, the covariance is also mis-estimated if $\Sigma_{12} \neq 0$ since, by construction of the MFVB approximation,

$$\text{Cov}_{q_0}(\theta_1, \theta_2) = 0 \neq \Sigma_{12} = \text{Cov}_{p_0}(\theta_1, \theta_2).$$

Now let us take the log perturbation $\rho(\theta, \alpha) = \theta_1 \alpha_1 + \theta_2 \alpha_2$. For all $\alpha$ in a neighborhood of zero, the log–perturbed posterior given by Eq. (2.2) remains multivariate normal, so it remains the case that, as a function of $\alpha$, $\mathbb{E}_{q_0}[\theta_1] = \mathbb{E}_{p_0}[\theta_1]$ and $\mathbb{E}_{q_0}[\theta_2] = \mathbb{E}_{p_0}[\theta_2]$. Again, see Section 2.9 for a detailed proof. Consequently, Condition 2.1 holds with equality (not approximate equality) when $g(\theta) = \theta$. However, since the second moments are not accurate (irrespective of $\alpha$), Condition 2.1 does not hold exactly when $g(\theta) = (\theta_1^2, \theta_2^2)^T$, nor when $g(\theta) = \theta_1, \theta_2$. (Condition 2.1 may still hold approximately for second moments when $\Sigma_{12}$ is small.)
Condition 2.1 holds with equality for \( g(\theta) = \theta \) allows us to use Theorem 2.1 and Theorem 2.2 to calculate \( \text{Cov}^{LR}_{\theta_0}(g(\theta)) = \text{Cov}_{\theta_0}(g(\theta)) \), even though \( E_{\theta_0}[\theta_1\theta_2] \) and \( E_{\theta_0}[(\theta_1^2, \theta_2^2)^T] \) are mis-estimated.

In fact, when Condition 2.1 holds with equality for some \( \theta_i \), then the estimated covariance in Eq. (2.14) for all terms involving \( \theta_i \) will be exact as well. Condition 2.1 holds with equality for the means of \( \theta_i \) in the bivariate normal model above, and in fact holds for the general multivariate normal case, as described in Section 2.9. Below, in Section 2.4, in addition to robustness measures, we will also report the accuracy of Eq. (2.14) for estimating posterior covariances. We find that, for most parameters of interest, particularly location parameters, \( \text{Cov}^{LR}_{\theta_0}(g(\theta)) \) provides a good approximation to \( \text{Cov}_{\theta_0}(g(\theta)) \).

**Linear response covariances in previous literature**

The application of sensitivity measures to VB problems for the purpose of improving covariance estimates has a long history under the name “linear response methods.” These methods originated in the statistical physics literature [Tanaka, 2000, Opper and Saad, 2001] and have been applied to various statistical and machine learning problems [Kappen and Rodriguez, 1998, Tanaka, 1998, Welling and Teh, 2004, Opper and Winther, 2003]. The current paper, which builds on this line of work and on our earlier work [Giordano et al., 2015], represents a simplification and generalization of classical linear response methods and serves to elucidate the relationship between these methods and the local robustness literature. In particular, while Giordano et al. [2015] focused on moment-parameterized exponential families, we derive linear-response covariances for generic variational approximations and connect the linear-response methodology to the Bayesian robustness literature.

A very reasonable approach to address the inadequacy of MFVB covariances is simply to increase the expressiveness of the model class \( \mathcal{Q} \)—although, as noted by Turner and Sahani [2011], increased expressiveness does not necessarily lead to better posterior moment estimates. This approach is taken by much of the recent VB literature [e.g., Tran et al., 2015a, b, Ranganath et al., 2016, Rezende and Mohamed, 2015, Liu and Wang, 2016]. Though this research direction remains lively and promising, the use of a more complex class \( \mathcal{Q} \) sometimes sacrifices the speed and simplicity that made VB attractive in the first place, and often without the relatively well-understood convergence guarantees of MCMC. We also stress that the current work is not necessarily at odds with the approach of increasing expressiveness. Sensitivity methods can be a supplement to any VB approximation for which our estimators, which require solving a linear system involving the Hessian of the KL divergence, are tractable.
The Laplace approximation and linear response covariances

In this section, we briefly compare linear response covariances to the Laplace approximation [Gelman et al., 2014, Chapter 13]. The Laplace approximation to $p_0(\theta)$ is formed by first finding the “maximum a posteriori” (MAP) estimate,

$$\hat{\theta}_{Lap} := \arg\max_{\theta} p_0(\theta), \quad (2.16)$$

and then forming the multivariate normal posterior approximation

$$H_{Lap} := \left. -\frac{\partial^2 p_0(\theta)}{\partial \theta \partial \theta^T} \right|_{\hat{\theta}_{Lap}}$$

$$\text{Cov}^{Lap}_{q_{Lap}}(\theta) := H_{Lap}^{-1}$$

$$q_{Lap}(\theta) := \mathcal{N}(\theta|\hat{\theta}_{Lap}, \text{Cov}^{Lap}_{q_{Lap}}(\theta)). \quad (2.18)$$

Since both LRVB and the Laplace approximation require the solution of an optimization problem (Eq. (2.8) and Eq. (2.16) respectively) and the estimation of covariances via an inverse Hessian of the optimization objective (Eq. (2.14) and Eq. (2.17) respectively), it will be instructive to compare the two approaches.

Following [Neal and Hinton, 1998], we can, in fact, view the MAP estimator as a special variational approximation, where we define

$$Q_{Lap} := \left\{ q(\theta; \theta_0) : \int q(\theta; \theta_0) \log p_0(\theta) \lambda(d\theta) = \log p_0(\theta_0) \quad \text{and} \quad \int q(\theta; \theta_0) \log q(\theta; \theta_0) \lambda(d\theta) = \text{Constant} \right\},$$

where the Constant term is constant in $\theta_0$. That is, $Q_{Lap}$ consists of “point masses” at $\theta_0$ with constant entropy. Generally such point masses may not be defined as densities with respect to $\lambda$, and the $KL$ divergence in Eq. (2.8) may not be formally defined for $q \in Q_{Lap}$. However, if $Q_{Lap}$ can be approximated arbitrarily well by well-defined densities (e.g., normal distributions with variance fixed at an arbitrarily small number), then we can use $Q_{Lap}$ as a heuristic tool for understanding the MAP estimator.

Since $Q_{Lap}$ contains only point masses, the covariance of the variational approximation is the zero matrix: $\text{Cov}^{q_{Lap}}(\theta) = 0$. Thus, as when one uses the mean field assumption, $\text{Cov}^{q_{Lap}}(\theta)$ underestimates the marginal variances and magnitudes of the covariances of $\text{Cov}_{p_0}(\theta)$. Of course, the standard Laplace approximation uses $\text{Cov}^{Lap}_{q_{Lap}}(\theta)$, not $\text{Cov}^{q_{Lap}}(\theta)$, to approximate $\text{Cov}_{p_0}(\theta)$. In fact, $\text{Cov}^{Lap}_{q_{Lap}}(\theta)$ is
equivalent to a linear response covariance matrix calculated for the approximating family $Q_{Lap}$:

$$KL(q(\theta; \theta_0) \| \pi_0(\theta)) = - \log \pi_0(\theta_0) - \text{Constant} \Rightarrow$$

$$\hat{\theta}_{Lap} = \underset{\theta}{\arg \max} p_0(\theta) = \underset{\theta_0}{\arg \min} KL(q(\theta; \theta_0) \| \pi_0(\theta)) = \theta^*$$

$$H_{Lap} = - \frac{\partial^2 p_0(\theta)}{\partial \theta \partial \theta^\top} \bigg|_{\theta = \hat{\theta}_{Lap}} = - \frac{\partial^2 KL(q(\theta; \theta_0) \| \pi_0(\theta))}{\partial \theta_0 \partial \theta_0^\top} \bigg|_{\theta_0 = \theta^*} = H_{\eta\eta}.$$ 

So $\hat{\theta}_{Lap} = \theta^*, H_{Lap} = H_{\eta\eta}$, and $\text{Cov}_{q_{Lap}}(\theta) = \text{Cov}_{q_0}^{LR}(\theta)$ for the approximating family $Q_{Lap}$.

From this perspective, the accuracy of the Laplace approximation depends precisely on the extent to which Condition 2.1 holds for the family of point masses $Q_{Lap}$. Typically, VB approximations use a $Q$ that is more expressive than $Q_{Lap}$, and we might expect Condition 2.1 to be more likely to apply for a more expressive family. It follows that we might expect the LRVB covariance estimate $\text{Cov}_{q_{Lap}}^{LR}$ for general $Q$ to be more accurate than the Laplace covariance approximation $\text{Cov}_{q_{Lap}}^{LR}$. We demonstrate the validity of this intuition in the experiments of Section 2.4.

**Local prior sensitivity for MFVB**

We now turn to estimating prior sensitivities for MFVB estimates—the variational analogues of $S_{\alpha_0}$ in Definition 2.1. First, we define the variational local sensitivity.

**Definition 2.7.** The local sensitivity of $E_{q_{\alpha}}[g(\theta)]$ to prior parameter $\alpha$ at $\alpha_0$ is given by

$$S_{\alpha_0}^q := \frac{dE_{q_{\alpha}}[g(\theta)]}{d\alpha} \bigg|_{\alpha_0}.$$ 

**Corollary 2.3.** Suppose that Assumptions 2.1–2.4 and Condition 2.1 hold for some $\alpha_0 \in \mathcal{A}$ and for

$$p(\theta, \alpha) = \log p(x|\theta, \alpha) + \log p(\theta|\alpha) - \log p(x|\theta, \alpha_0) - \log p(\theta|\alpha).$$

Then $S_{\alpha_0}^q \approx S_{\alpha_0}$.

Corollary 2.3 states that, as with the covariance approximations in Section 2.3, $S_{\alpha_0}^q$ is a useful approximation to $S_{\alpha_0}$ to the extent that Condition 2.1 holds—that is, to the extent that the MFVB means are good approximations to the exact means for the prior perturbations $\alpha \in \mathcal{A}_0$. 

Under the $\rho (\theta, \alpha)$ given in Corollary 2.3, Theorem 2.2 gives the following formula for the variational local sensitivity:

$$S^q_{\alpha_0} = g_\eta H_{\eta\eta}^{-1} \frac{\partial}{\partial \eta^T} \mathbb{E}_{q(\theta, \eta)} \left[ \frac{\partial \rho (\theta, \alpha)}{\partial \alpha} \right]_{\alpha_0} \eta_0^* .$$  \hspace{1cm} (2.19)

We now use Eq. (2.19) to reproduce MFVB versions of some standard robustness measures found in the existing literature. A simple case is when the prior $p (\theta|\alpha)$ is believed to be in a given parametric family, and we are simply interested in the effect of varying the parametric family’s parameters [Basu et al., 1996, Giordano et al., 2016]. For illustration, we first consider a simple example where $p (\theta|\alpha)$ is in the exponential family, with natural sufficient statistic $\theta$ and log normalizer $A (\alpha)$, and we take $g (\theta) = \theta$. In this case,

$$\log p (\theta|\alpha) = \alpha^T \theta - A (\alpha)$$

$$f_{\alpha \eta} = \frac{\partial}{\partial \eta^T} \mathbb{E}_{q(\theta, \eta)} \left[ \frac{\partial}{\partial \alpha} (\alpha^T \theta - A (\alpha)) \right]_{\alpha_0} \eta_0^*$$

$$= \left( \frac{\partial}{\partial \eta^T} \mathbb{E}_{q(\theta, \eta)} [\theta] - \frac{\partial}{\partial \eta^T} \frac{\partial A (\alpha)}{\partial \alpha} \right)_{\alpha_0} \eta_0^*$$

$$= \frac{\partial}{\partial \eta^T} \mathbb{E}_{q(\theta, \eta)} [\theta] \bigg|_{\eta_0^*}$$

$$= g_\eta .$$

Note that when $f_{\alpha \eta} = g_\eta$, Eq. (2.19) is equivalent to Eq. (2.14). So we see that

$$S^q_{\alpha_0} = \text{Cov}^{LR}_{q0} (\theta) .$$

In this case, the sensitivity is simply the linear response covariance estimate of the covariance, $\text{Cov}^{LR}_{q0} (\theta)$. By the same reasoning, the exact posterior sensitivity is given by

$$S_{\alpha_0} = \text{Cov}_{p0} (\theta) .$$

Thus, $S^q_{\alpha_0} \approx S_{\alpha_0}$ to the extent that $\text{Cov}^{LR}_{q0} (\theta) \approx \text{Cov}_{p0} (\theta)$, which again holds to the extent that Condition 2.1 holds. Note that if we had used a mean field assumption and had tried to use the direct, uncorrected response covariance $\text{Cov}_{q0} (\theta)$ to try to evaluate $S^q_{\alpha_0}$, we would have erroneously concluded that the prior on one component, $\theta_{k_1}$, would not affect the posterior mean of some other component, $\theta_{k_2}$, for $k_2 \neq k_1$. 
Sometimes it is easy to evaluate the derivative of the log prior even when it is not easy to normalize it. As an example, we will show how to calculate the local sensitivity to the concentration parameter of an LKJ prior [Lewandowski et al., 2009] under an inverse Wishart variational approximation. The LKJ prior is defined as follows. Let $\Sigma$ (as part of $\theta$) be an unknown $K \times K$ covariance matrix. Define the $K \times K$ scale matrix $M$ such that

$$ M_{ij} = \begin{cases} \sqrt{\Sigma_{ij}} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} $$

Using $M$, define the correlation matrix $R$ as

$$ R = M^{-1} \Sigma M^{-1}. $$

The LKJ prior on the covariance matrix $R$ with concentration parameter $\alpha > 0$ is given by:

$$ p_{\text{LKJ}}(R|\alpha) \propto |R|^{\alpha - 1}. $$

The Stan manual recommends the use of $p_{\text{LKJ}}$, together with an independent prior on the diagonal entries of the scaling matrix $M$, for the prior on a covariance matrix that appears in a hierarchical model [Stan Team, 2015, Chapter 9.13].

Suppose that we have chosen the variational approximation

$$ q(\Sigma) := \text{InverseWishart}(\Sigma|\Psi, \nu), $$

where $\Psi$ is a positive definite scale matrix and $\nu$ is the number of degrees of freedom. In this case, the variational parameters are $\eta = (\Psi, \nu)$. We write $\eta$ with the understanding that we have stacked only the upper-diagonal elements of $\Psi$ since $\Psi$ is constrained to be symmetric and $\eta^*$ must be interior. As we show in Appendix A.5

$$ \mathbb{E}_q[\log p_{\text{LKJ}}(R|\alpha)] = $$

$$ (\alpha - 1) \left( \log |\Psi| - \psi_K \left( \frac{\nu}{2} \right) - \sum_{k=1}^{K} \log \left( \frac{1}{2} \Psi_{kk} \right) + K \psi \left( \frac{\nu - K + 1}{2} \right) \right) + Constant, $$
where \( \text{Constant} \) contains terms that do not depend on \( \alpha \), and where \( \psi_K \) denotes the multivariate digamma function. Consequently, we can evaluate

\[
f_{\alpha \eta} = \left. \frac{\partial}{\partial \eta^{\top}} \mathbb{E}_{q(\theta; \eta)} \left[ \frac{\partial}{\partial \alpha} \log p(\Sigma | \alpha) \right] \right|_{\eta = \eta^*_0, \alpha = \alpha_0}
= \left. \frac{\partial}{\partial \eta^{\top}} \left( \log |\Psi| - \psi_K \left( \frac{n}{2} \right) - \sum_{k=1}^{K} \log \left( \frac{1}{2} \Psi_{kk} \right) + K \psi \left( \frac{n - K + 1}{2} \right) \right) \right|_{\eta^*_0}.
\]

This derivative has a closed form, but the bookkeeping required to represent an unconstrained parameterization of the matrix \( \Psi \) within \( \eta \) would be tedious. In practice, we evaluate terms like \( f_{\alpha \eta} \) using automatic differentiation tools [Baydin et al., 2018].

Finally, in cases where we cannot evaluate \( \mathbb{E}_{q(\theta; \eta)} [ \log p(\theta | \alpha) ] \) in closed form as a function of \( \eta \), we can use numerical techniques as described in Section 2.3. We thus view \( S_{q,0}^0 \) as the exact sensitivity to an approximate KL divergence.

**Practical considerations when computing the sensitivity of variational approximations**

We briefly discuss practical issues in the computation of Eq. (2.10), which requires calculating the product \( g_{\eta} H_{\eta \eta}^{-1} \) (or, equivalently, \( H_{\eta \eta}^{-1} g_{\eta}^{\top} \)) since \( H_{\eta \eta} \) is symmetric. Calculating \( H_{\eta \eta} \) and solving this linear system can be the most computationally intensive part of computing Eq. (2.10).

We first note that it can be difficult and time consuming in practice to manually derive and implement second-order derivatives. Even a small programming error can lead to large errors in Theorem 2.2. To ensure accuracy and save analyst time, we evaluated all the requisite derivatives using the Python autograd automatic differentiation library [Maclaurin et al., 2015] and the Stan math automatic differentiation library [Carpenter et al., 2015].

Note that the dimension of \( H_{\eta \eta} \) is as large as that of \( \eta \), the parameters that specify the variational distribution \( q(\theta; \eta) \). Many applications of MFVB employ many latent variables, the number of which may even scale with the amount of data—including several of the cases that we examine in Section 2.4. However, these applications typically have special structure that render \( H_{\eta \eta} \) sparse, allowing the practitioner to calculate \( g_{\eta} H_{\eta \eta}^{-1} \) quickly. Consider, for example, a model with “global” parameters, \( \theta_{\text{glob}} \), that are shared by all the individual datapoint likelihoods, and “local” parameters, \( \theta_{\text{loc},n} \), associated with likelihood of a single datapoint indexed by \( n \). By “global” and “local” we mean the likelihood and assumed variational
distribution factorize as

\[
p(x, \theta_{\text{glob}}, \theta_{\text{loc},1}, \ldots, \theta_{\text{loc},N}) = p(\theta_{\text{glob}}) \prod_{n=1}^{N} p(x|\theta_{\text{loc},n}, \theta_{\text{glob}}) p(\theta_{\text{loc},n}|\theta_{\text{glob}}) \tag{2.21}
\]

\[
q(\theta; \eta) = q(\theta_{\text{glob}}; \eta_{\text{glob}}) \prod_{n=1}^{N} q(\theta_{\text{loc},n}; \eta_{n}) \text{ for all } q(\theta; \eta) \in Q.
\]

In this case, the second derivatives of the variational objective between the parameters for local variables vanish:

\[
\text{for all } n \neq m, \frac{\partial^2 KL(q(\theta; \eta)||p_0(\theta))}{\partial \eta_{\text{loc},n} \partial \eta_{\text{loc},m}^\top} = 0.
\]

The model in Section 2.4 has such a global / local structure; see Section 2.4 for more details. Additional discussion, including the use of Schur complements to take advantage of sparsity in the log likelihood, can be found in Giordano et al. [2015].

When even calculating or instantiating \( H_{\eta \eta} \) is prohibitively time-consuming, one can use conjugate gradient algorithms to approximately compute \( H_{\eta \eta}^{-1} g_\eta^\top \) [Wright and Nocedal 1999 Chapter 5]. The advantage of conjugate gradient algorithms is that they approximate \( H_{\eta \eta}^{-1} g_\eta^\top \) using only the Hessian-vector product \( H_{\eta \eta} g_\eta^\top \), which can be computed efficiently using automatic differentiation without ever forming the full Hessian \( H_{\eta \eta} \). See, for example, the hessian_vector_product method of the Python autograd package [Maclaurin et al. 2015]. Note that a separate conjugate gradient problem must be solved for each column of \( g_\eta^\top \), so if the parameter of interest \( g(\theta) \) is high-dimensional it may be faster to pay the price for computing and inverting the entire matrix \( H_{\eta \eta} \). See 2.4 for more discussion of a specific example.

In Theorem 2.2, we require \( \eta_0^* \) to be at a true local optimum. Otherwise the estimated sensitivities may not be reliable (e.g., the covariance implied by Eq. (2.14) may not be positive definite). We find that the classical MFVB coordinate ascent algorithms [Blei et al. 2016 Section 2.4] and even quasi-second order methods, such as BFGS [e.g., Regier et al. 2015], may not actually find a local optimum unless run for a long time with very stringent convergence criteria. Consequently, we recommend fitting models using second-order Newton trust region methods. When the Hessian is slow to compute directly, as in Section 2.4, one can use the conjugate gradient trust region method of Wright and Nocedal [1999 Chapter 7], which takes advantage of fast automatic differentiation Hessian-vector products without forming or inverting the full Hessian.
2.4 Experiments

We now demonstrate the speed and effectiveness of linear response methods on a number of simulated and real data sets. We begin with simple simulated data to provide intuition for how linear response methods can improve estimates of covariance relative to MFVB and the Laplace approximation. We then develop linear response covariance estimates for ADVI and apply them to four real-world models and data sets taken from the Stan examples library [Stan Team, 2017]. Finally, we calculate both linear response covariances and prior sensitivity measures for a large-scale industry data set. In each case, we compare linear response methods with ordinary MFVB, the Laplace approximation, and MCMC. We show that linear response methods provide the best approximation to MCMC while still retaining the speed of approximate methods. Code and instructions to reproduce the results of this section can be found in the git repository [rgiordan/CovariancesRobustnessVBPaper].

Simple expository examples

In this section we provide a sequence of simple examples comparing MFVB and LRVB with Laplace approximations. These examples provide intuition for the covariance estimate $\text{Cov}^{LR}_{q_0}(g(\theta))$ and illustrate how the sensitivity analysis motivating $\text{Cov}^{LR}_{q_0}(g(\theta))$ differs from the local posterior approximation motivating $\text{Cov}^{Lap}_{q_{Lap}}(g(\theta))$.

For each example, we will explicitly specify the target posterior $p_0(\theta)$ using a mixture of normals. This will allow us to define known target distributions with varying degrees of skewness, over-dispersion, or correlation and compare the truth with a variational approximation. Formally, for some fixed $K_z$, component indicators $z_k$, $k = 1, \ldots, K_z$, component probabilities $\pi_k$, locations $\mu_k$, and covariances $\Sigma_k$, we set

$$p(z) = \prod_{k=1}^{K_z} \pi_k^{z_k}$$
$$p_0(\theta) = \sum_z p(z) p(\theta|z) = \sum_z p(z) \prod_{k=1}^{K_z} \mathcal{N}(\theta; m_k, \Sigma_k)^{z_k}.$$

The values $\pi$, $m$ and $\Sigma$ will be chosen to achieve the desired shape for each example using up to $K_z = 3$ components. There will be no need to state the precise values of $\pi$, $m$, and $\Sigma$; rather, we will show plots of the target density and report the marginal means and variances, calculated by Monte Carlo.\footnote{MFVB is often used to approximate the posterior when the Bayesian generative model for data is complex and exact inference is intractable.}
CHAPTER 2. COVARIANCES AND ROBUSTNESS

We will be interested in estimating the mean and variance of the first component, so we take \( g(\theta) = \theta_1 \). Consequently, in order to calculate \( \text{Cov}^{LR}_{q_0}(\theta_1) \), we will be considering the perturbation \( \rho(\theta, \alpha) = \alpha \theta_1 \) with scalar \( \alpha \) and \( \alpha_0 = 0 \).

For the variational approximations, we will use a factorizing normal approximation:

\[
Q_{mf} = \left\{ q(\theta) : q(\theta) = \prod_{k=1}^{K} \mathcal{N}(\theta_k; \mu_k, \sigma_k^2) \right\}
\]

In terms of Eq. (2.7), we take \( \eta = (\mu_1, ..., \mu_K, \log \sigma_1, ..., \log \sigma_K)^T \). Thus \( \mathbb{E}_{q(\theta; \eta)}[g(\theta)] = \mathbb{E}_{q(\theta; \eta)}[\theta_1] = \mu_1 \). In the examples below, we will use multiple distinct components in the definition of \( p_0(\theta) \), so that \( p_0(\theta) \) is non-normal and \( p_0(\theta) \notin Q_{mf} \).

Since the expectation \( \mathbb{E}_{q(\theta; \eta)}[\log p(\theta)] \) is intractable, we replace the exact KL divergence with a Monte Carlo approximation using the “re-parameterization trick” [Kingma and Welling, 2013, Rezende et al., 2014, Titsias and Lázaro-Gredilla, 2014]. Let \( \circ \) denote the Hadamard (component-wise) product. Let \( \xi_m \overset{iid}{\sim} \mathcal{N}(0, I_K) \) for \( m = 1, ..., M \). We define

\[
\theta_m := \sigma \circ \xi_m + \mu
\]

\[
KL_{\text{approx}}(q(\theta; \eta) \| p_0(\theta)) := -\frac{1}{M} \sum_{m=1}^{M} \log p_0(\theta_m) - \sum_{k=1}^{K} \log \sigma_k,
\]

which is a Monte Carlo estimate of \( KL(q(\theta; \eta) \| p_0(\theta)) \). We found \( M = 10000 \) to be more than adequate for our present purposes of illustration. Note that we used the same draws \( \xi_m \) for both optimization and for the calculation of \( H_{\eta\eta} \) in order to ensure that the \( \eta^*_0 \) at which \( H_{\eta\eta} \) was evaluated was in fact an optimum. This approach is similar to our treatment of ADVI; see Section 2.4 for a more detailed discussion.

Multivariate Normal targets

If we take only a single component in the definition of \( p_0(\theta) (K_z = 1) \), then \( p_\alpha(\theta) \) is a multivariate normal distribution for all \( \alpha \), and the Laplace approximation \( q_{\text{Lap}}(\theta) \) is equal to \( p_\alpha(\theta) \) for all \( \alpha \). Furthermore, as discussed in Section 2.3 and Section 2.9, the variational means \( \mathbb{E}_{q_\alpha}[\theta] = \mu \) are exactly equal to the exact posterior mean \( x \) is a mixture model (e.g., Blei et al., 2003). By contrast, we note for clarity that we are not using the mixture model as a generative model for \( x \) here. E.g., \( z \) is not one of the parameters composing \( \theta \), and we are not approximating the distribution of \( z \) in the variational distribution \( q(\theta) \). Rather, we are using mixtures as a way of flexibly defining skewed and over-dispersed targets, \( p(\theta) \).
\( E_{p_0}[\theta] = m_1 \) for all \( \alpha \) (even though in general \( \text{Cov}_{q_0}(\theta) \neq \Sigma_1 \)). Consequently, for all \( \alpha \), the variational approximation, the Laplace approximation, and the exact \( p_0(\theta) \) all coincide in their estimates of \( E[\theta] \), and by Corollary 2.2 \( \Sigma = \text{Cov}_{p_0}(\theta) = \text{Cov}_{q_0}^{LR}(\theta) = \text{Cov}_{q_0}^{Lap}(\theta) \). Of course, if \( \Sigma \) is not diagonal, \( \text{Cov}_{q_0}(\theta) \neq \Sigma \) because of the mean field assumption. Since this argument holds for the whole vector \( \theta \), it holds a fortiori for our quantity of interest, the first component \( g(\theta) = \theta_1 \).

In other words, the Laplace approximation will differ only from the LRVB approximation when \( p_0(\theta) \) is not multivariate normal, a situation that we will now bring about by adding new components to the mixture; i.e., by increasing \( K_z \).

A univariate skewed distribution

If we add a second component \((K_z = 2)\), then we can make \( p_0(\theta) \) skewed, as shown (with the approximations) in Fig. (2.1). In this case, we expect \( E_{q_{nf}}[\theta_1] \) to be more accurate than the Laplace approximation \( E_{q_{Lap}}[\theta_1] \) because \( Q_{mf} \) is more expressive than \( Q_{Lap} \). This intuition is born out in the left panel of Fig. (2.1). Since \( \hat{\theta}_{Lap} \) uses only information at the mode, it fails to take into account the mass to the right of the mode, and the Laplace approximation’s mean is too far to the left. The MFVB approximation, in contrast, is quite accurate for the posterior mean of \( \theta_1 \), even though it gets the overall shape of the distribution wrong.

This example also shows why, in general, one cannot naively form a “Laplace approximation” to the posterior centered at the variational mean rather than at the MAP. As shown in the left panel of Fig. (2.1), in this case the posterior distribution is actually convex at the MFVB mean. Consequently, a naive second-order approximation to the log posterior centered at the MFVB mean would imply a negative variance.
The perturbation $\rho(\theta, \alpha) = \alpha \theta_1$ is sometimes also described as a “tilting,” and the right panel of Fig. (2.1) shows the effect of tilting on this posterior approximation. Tilting increases skew, but the MFVB approximation remains accurate, as shown in Fig. (2.2). Since local sensitivity of the expectation of $\theta_1$ to $\alpha$ is the variance of $\theta_1$ (see Eq. (2.13)), we have in Fig. (2.2) that:

- The slope of the exact distribution’s line is $\text{Cov}_{p_0}(\theta_1)$;
- The slope of the MFVB line is the LRVB variance $\text{Cov}_{q_{LR}}^{LR}(\theta_1)$; and
- The slope of the Laplace line is $\text{Cov}_{q_{Lap}}^{Lap}(\theta_1)$.

Since the MFVB and exact lines nearly coincide, we expect the LRVB variance estimate to be quite accurate for this example. Similarly, since the slope of the Laplace approximation line is lower, we expect the Laplace variance to underestimate the exact variance. This outcome, which can be seen visually in the left-hand panel of Fig. (2.2), is shown quantitatively in the corresponding table in the right-hand panel. The columns of the table contain information for the exact distribution and the three approximations. The first row, labeled “mean,” shows $\mathbb{E}[\theta_1]$ and the second row, labeled “var,” shows $\text{Cov}(\theta_1)$. (The “LRVB” entry for the mean is blank.)
Figure 2.3: A univariate over-dispersed distribution. Vertical lines show the location of the means.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Exact</th>
<th>LRVB</th>
<th>MFVB</th>
<th>Laplace</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>-0.001</td>
<td>0.027</td>
<td>-0.000</td>
<td></td>
</tr>
<tr>
<td>var</td>
<td>4.218</td>
<td>4.153</td>
<td>4.161</td>
<td>1.107</td>
</tr>
</tbody>
</table>

Figure 2.4: Effect of tilting on a univariate over-dispersed distribution.

because LRVB differs from MFVB only in covariance estimates.) We conclude that, in this case, Condition 2.1 holds for $Q_{mf}$ but not for $Q_{Lap}$. 
A univariate over-dispersed distribution

Having seen how MFVB can outperform the Laplace approximation for a univariate skewed distribution, we now apply that intuition to see why the linear response covariance can be superior to the Laplace approximation covariance for over-dispersed but symmetric distributions. Such a symmetric but over-dispersed distribution, formed with \( K_z = 3 \) components, is shown in Fig. (2.3) together with its approximations. By symmetry, both the MFVB and Laplace means are exactly correct (up to Monte Carlo error), as can be seen in the left panel of Fig. (2.3).

However, the right panel of Fig. (2.3) shows that symmetry is not maintained as the distribution is tilted. For \( \alpha > 0 \), the distribution becomes skewed to the right. Thus, by the intuition from the previous section, we expect the MFVB mean to be more accurate as the distribution is tilted and \( \alpha \) increases from zero. In particular, we expect that the Laplace approximation’s mean will not shift enough as \( \alpha \) varies, i.e., that the Laplace approximation variance will be underestimated. Fig. (2.4) shows that this is indeed the case. The slopes in the left panel once again correspond to the estimated variances shown in the table, and, as expected the LRVB variance estimate is superior to the Laplace approximation variance.

In this case, Condition 2.1 holds for \( Q_{mf} \). For the Laplace approximation, 
\[
\mathbb{E}_{q_{Lap}} [g(\theta)] = \mathbb{E}_{p_0} [g(\theta)] \text{ for } \alpha = 0,
\]
so \( Q_{Lap} \) satisfies Eq. (2.11) of Condition 2.1. For \( \alpha \) near zero, the derivatives of the two expectations with respect to \( \alpha \) are quite different, so Eq. (2.12) of Condition 2.1 does not hold for \( Q_{Lap} \).

A bivariate over-dispersed distribution

In the previous two examples the mean field approximation in \( Q \) did not matter, since the examples were one-dimensional. The only reason that the variational approximation was different from the exact \( p_0(\theta) \) was the normal assumption in \( Q_{mf} \).
Indeed, the tables in Fig. (2.2) and Fig. (2.4) show that the MFVB variance estimate is also reasonably close to the exact variance. In order to demonstrate why the LRVB variance can be better than both the Laplace approximation and the MFVB approximation, we turn to a bivariate, correlated, over-dispersed \( p_\theta (\theta) \). For this we use \( K_z = 3 \) correlated normal distributions, shown in the left panel of Fig. (2.5). The right panel of Fig. (2.5) shows the marginal distribution of \( \theta_1 \), in which the over-dispersion can be seen clearly. As Fig. (2.5) shows, unlike in the previous two examples, the mean field approximation causes \( q_0 (\theta) \) to dramatically underestimate the marginal variance of \( \theta_1 \). Consequently, the MFVB means will also be under-responsive to the skew introduced by tilting with \( \alpha \). Though the Laplace approximation has a larger marginal variance, it remains unable to take skewness into account. Consequently, as seen in Fig. (2.6), the LRVB variance, while not exactly equal to the correct variance, is still an improvement over the Laplace covariance, and a marked improvement on the badly under-estimated MFVB variance.

One might say, in this case, that Condition 2.1 does not hold for either \( Q_{mf} \) or \( Q_{Lap} \), or, if it does, it is with a liberal interpretation of the “approximately equals” sign. However, the expressiveness of \( Q_{mf} \) allows LRVB to improve on the Laplace approximation, and the linear response allows it to improve over the MFVB ap-
proximation, and so LRVB gives the best of both worlds.

Thinking about problems in terms of these three simple models can provide intuition about when and whether Condition 2.1 might be expected to hold in a sense that is practically useful.

**Automatic differentiation variational inference (ADVI)**

In this section we apply our methods to automatic differentiation variational inference (ADVI) [Kucukelbir et al. 2017]. ADVI is a “black-box” variational approximation and optimization procedure that requires only that the user provide the log posterior, \( \log p_0(\theta) \), up to a constant that does not depend on \( \theta \). To achieve this generality, ADVI employs:

- A factorizing normal variational approximation
- An unconstraining parameterization,
- The “re-parameterization trick,” and
- Stochastic gradient descent.

ADVI uses a family employing the factorizing normal approximation

\[
Q_{ad} := \left\{ q(\theta) : q(\theta) = \prod_{k=1}^{K} \mathcal{N}(\theta_k|\mu_k, \exp(2\zeta_k)) \right\}.
\]

That is, \( Q_{ad} \) is a fully factorizing normal family with means \( \mu_k \) and log standard deviations \( \zeta_k \). Because we are including exponential family assumptions in the definition of MFVB (as described in Section 2.2), \( Q_{ad} \) is an instance of a mean-field family \( Q_{mf} \). In the notation of Eq. (2.7),

\[
\eta = (\mu_1, \ldots, \mu_K, \zeta_1, \ldots, \zeta_K)^T,
\]

\[\Omega_\eta = \mathbb{R}^{2K}, \lambda \text{ is the Lebesgue measure}, \text{ and the objective function Eq. (2.8) is} \]

\[
KL(q(\theta; \eta) || p_0(\theta)) = - \int \mathcal{N}(\theta_k|\mu_k, \exp(2\zeta_k)) \log p_0(\theta) \lambda(d\theta) - \sum_{k=1}^{K} \zeta_k,
\]

[Kucukelbir et al. 2017] describe a non-factorizing version of ADVI, which is called “fullrank” ADVI in Stan. The factorizing version that we describe here is called “meanfield” ADVI in Stan. On the examples we describe, in the current Stan implementation, we found that fullrank ADVI provided much worse approximations to the MCMC posterior means than the meanfield version, and so we do not consider it further.
where we have used the form of the univariate normal entropy up to a constant.

The unconstraining parameterization is required because the use of a normal variational approximation dictates that the base measure on the parameters $\theta \in \mathbb{R}^K$ be supported on all of $\mathbb{R}^K$. Although many parameters of interest, such as covariance matrices, are not supported on $\mathbb{R}^K$, there typically exist differentiable maps from an unconstrained parameterization supported on $\mathbb{R}^K$ to the parameter of interest. Software packages such as Stan automatically provide such transforms for a broad set of parameter types. In our notation, we will take these constraining maps to be the function of interest, $g(\theta)$, and take $\theta$ to be unconstrained. Note that, under this convention, the prior $p(\theta | \alpha)$ must be a density in the unconstrained space. In practice (e.g., in the Stan software package), one usually specifies the prior density in the constrained space and converts it to a density $p(\theta | \alpha)$ in the unconstrained space using the determinant of the Jacobian of the constraining transform $g(\cdot)$.

The re-parameterization trick allows easy approximation of derivatives of the (generally intractable) objective $KL(q(\theta; \eta) \mid \mid p_0(\theta))$. By defining $z_k$ using the change of variable

$$z_k := (\theta_k - \mu_k) / \exp(\zeta_k),$$

$KL(q(\theta; \eta) \mid \mid p_0(\theta))$ can be re-written as an expectation with respect to a standard normal distribution. We write $\theta = \exp(\zeta) \circ z + \mu$ by using the component-wise Hadamard product $\circ$. Then

$$KL(q(\theta; \eta) \mid \mid p_0(\theta)) = -\mathbb{E}_z \left[ \log p_0(\exp(\zeta) \circ z + \mu) \right] - \sum_{k=1}^{K} \zeta_k + \text{Constant}.$$  

The expectation is still typically intractable, but it can be approximated using Monte Carlo and draws from a $K$-dimensional standard normal distribution. For a fixed number $M$ of draws $z_1, ..., z_M$ from a standard $K$-dimensional normal, we can define the approximate KL divergence

$$\hat{KL}(\eta) := -\frac{1}{M} \sum_{m=1}^{M} \log p_0(\exp(\zeta) \circ z_m + \mu) - \sum_{k=1}^{K} \zeta_k + \text{Constant}. $$

For any fixed $M$,

$$\mathbb{E} \left[ \frac{\partial}{\partial \eta} \hat{KL}(\eta) \right] = \frac{\partial}{\partial \eta} KL(q(\theta; \eta) \mid \mid p_0(\theta)),$$

so gradients of $\hat{KL}(\eta)$ are unbiased for gradients of the exact KL divergence. Furthermore, for fixed draws $z_1, ..., z_M$, $\hat{KL}(\eta)$ can be easily differentiated (using,
again, the re-parameterization trick). Standard ADVI uses this fact to optimize \( K L (q(\theta; \eta) \mid \mid p_0(\theta)) \) using the unbiased gradient draws \( \frac{\partial}{\partial \eta} \hat{K L}(\eta) \) and a stochastic gradient optimization method, where the stochasticity comes from draws of the standard normal random variable \( z \). Note that stochastic gradient methods typically use a new draw of \( z \) at every gradient step.

**Linear response for ADVI (LR-ADVI)**

Since ADVI uses a factorizing normal approximation, the intuition from Section 2.4 may be expected to apply. In particular, we might expect that the ADVI means \( \hat{\mu} \) might be a good approximation to \( \mathbb{E}_{p_0}[\theta] \), that the ADVI variances \( \exp(2\hat{\zeta}) \) would be under-estimates of the posterior variance \( \text{Cov}_{p_0}(\theta) \), so that using \( \text{Cov}_{LR}^{\text{q0}}(\theta) \) could improve the approximations to the posterior variance. We refer to LRVB covariances calculated using an ADVI approximation as LR-ADVI.

To apply linear response to an ADVI approximation, we need to be able to approximate the Hessian of \( K L (q(\theta; \eta) \mid \mid p_0(\theta)) \) and to be assured that we have found an optimal \( \eta_0 \). But, by using a stochastic gradient method, ADVI avoids ever actually calculating the expectation in \( K L (q(\theta; \eta) \mid \mid p_0(\theta)) \). Furthermore even if a stochastic gradient method finds an point that is close to the optimal value of \( K L (q(\theta; \eta) \mid \mid p_0(\theta)) \) it may not be close to an optimum of \( \hat{K L}(\eta) \) for a particular finite \( M \). Indeed, we found that, even for very large \( M \), the optimum found by ADVI’s stochastic gradient method is typically not close enough to an optimum of the approximate \( \hat{K L}(\eta) \) for sensitivity calculations to be useful. Sensitivity calculations are based on differentiating the fixed point equation given by the gradient being zero (see the proof in Section 2.8), and do not apply at points for which the gradient is not zero either in theory nor in practice.

Consequently, in order to calculate the local sensitivity, we simply eschew the stochastic gradient method and directly optimize \( \hat{K L}(\eta) \) for a particular choice of \( M \). (We will discuss shortly how to choose \( M \).) We can then use \( \hat{K L}(\eta) \) in Eq. (2.10) rather than the exact KL divergence. Directly optimizing \( \hat{K L}(\eta) \) both frees us to use second-order optimization methods, which we found to converge more quickly to a high-quality optimum than first-order methods, and guarantees that we are evaluating the Hessian \( \mathbf{H}_{\eta \eta} \) at an optimum of the objective function used to calculate Eq. (2.10).

As \( M \) approaches infinity, we expect the optimum of \( \hat{K L}(\eta) \) to approach the optimum of \( K L (q(\theta; \eta) \mid \mid p_0(\theta)) \) by the standard frequentist theory of estimating equations [Keener, 2011, Chapter 9]. In practice we must fix a particular finite \( M \), with larger \( M \) providing better approximations of the true KL divergence but at increased computational cost. We can inform this tradeoff between accuracy
and computation by considering the frequentist variability of $\eta^*_0$ when randomly sampling $M$ draws of the random variable $z$ used to approximate the intractable integral in $\hat{KL}(\eta)$. Denoting this frequentist variability by $\text{Cov}_z(\eta^*_0)$, standard results [Keener, 2011, Chapter 9] give that

$$\text{Cov}_z(\eta^*_0) \approx H^{-1}_{\eta\eta} \partial \hat{KL}(\eta) \bigg|_{\eta^*_0} H^{-1}_{\eta\eta}. \quad (2.25)$$

A sufficiently large $M$ will be one for which $\text{Cov}_z(\eta^*_0)$ is adequately small. One notion of “adequately small” might be that the ADVI means found with $\hat{KL}(\eta)$ are within some fraction of a posterior standard deviation of the optimum of $KL(q(\theta; \eta) || p_0(\theta))$. Having chosen a particular $M$, we can calculate the frequentist variability of $\mu^*$ using $\text{Cov}_{q_0}(g(\theta))$ and estimate the posterior standard deviation using Eq. (2.14). If we find that each $\mu^*$ is probably within 0.5 standard deviations of the optimum of $KL(q(\theta; \eta) || p_0(\theta))$, we can keep the results; otherwise, we increase $M$ and try again. In the examples we consider here, we found that the relatively modest $M = 10$ satisfies this condition and provides sufficiently accurate results.

Finally, we note a minor departure from Eq. (2.14) when calculating $\text{Cov}_{q_0}(g(\theta))$ from $H_{\eta\eta}$. Recall that, in this case, we are taking $g(\cdot)$ to be ADVI’s constraining transform, and that Eq. (2.14) requires the Jacobian, $g_\eta$, of this transform. At the time of writing, the design of the Stan software package did not readily support automatic calculation of $g_\eta$, though it did support rapid evaluation of $g(\theta)$ at particular values of $\theta$. Consequently, we used linear response to estimate $\text{Cov}_{q_0}(\theta)$, drew a large number $N_\theta$ of Monte Carlo draws from $\theta_n \sim \mathcal{N}(\mu, \text{Cov}_{q_0}(\theta))$ for $n = 1, ..., N_\theta$, and then used these draws to form a Monte Carlo estimate of the sample covariance of $g(\theta)$. Noting that $\mathbb{E}_{q_\alpha}[\theta] = \mu$, and recalling the definition of $\eta$ for ADVI in Eq. (2.22), by Eq. (2.14) we have

$$\text{Cov}_{q_0}(\theta) = \frac{\partial \mathbb{E}_{q_\alpha}[\theta]}{\partial \eta^\top} H^{-1}_{\eta\eta} \frac{\partial \mathbb{E}_{q_\alpha}[\theta^\top]}{\partial \eta} = \left( \begin{array}{ccc} I_K & 0 & 0 \\ 0 & 0 & 0 \end{array} \right) H^{-1}_{\eta\eta} \left( \begin{array}{ccc} I_K & 0 & 0 \\ 0 & 0 & 0 \end{array} \right),$$

which is the upper-left quarter of the matrix $H^{-1}_{\eta\eta}$. In addition to obviating the need for $g_\eta$, this approach also allowed us to take into account possible nonlinearities in $g(\cdot)$ at little additional computational cost.

**Results**

We present results from four models taken from the Stan example set, namely the models election88 (“Election model”), sesame.street1 (“Sesame Street model”),
We experimented with many models from the Stan examples and selected these four as representative of the type of model where LR-ADVI can be expected to provide a benefit—specifically, they are models of a moderate size. For very small models, MCMC runs quickly enough in Stan that fast approximations are not necessary, and for very large models (with thousands of parameters) the relative advantages of LR-ADVI and the Laplace approximation diminish due to the need to calculate $H_{\eta\eta}$ or $H_{\text{Lap}}$ using automatic differentiation. The size of the data and size of the parameter space for our four chosen models are shown in Fig. (2.11). We also eliminated from consideration models where Stan’s MCMC algorithm reported divergent transitions or where Stan’s ADVI algorithm returned wildly inaccurate posterior mean estimates.

For brevity, we do not attempt to describe the models or data in any detail here; rather, we point to the relevant literature in their respective sections. The data and Stan implementations themselves can be found on the Stan website [Stan Team, 2017].

To assess the accuracy of each model, we report means and standard deviations for each of Stan’s model parameters as calculated by Stan’s MCMC and ADVI algorithms and a Laplace approximation, and we report the standard deviations as calculated by $\text{Cov}^{LR}_{q_0}(g(\theta))$. Recall that, in our notation, $g(\cdot)$ is the (generally non-linear) map from the unconstrained latent ADVI parameters to the constrained space of the parameters of interest. The performance of ADVI and Laplace vary, and only LR-ADVI provides a consistently good approximation to the MCMC standard deviations. LR-ADVI was somewhat slower than a Laplace approximation or ADVI alone, but it was typically about five times faster than MCMC; see Section 2.4 for detailed timing results.

**Election model accuracy**

We begin with election88, which models binary responses in a 1988 poll using a Bernoulli hierarchical model with normally distributed random effects for state, ethnicity, and gender and a logit link. The model and data are described in detail in Gelman and Hill [2006, Chapter 14]. Fig. (2.7) shows that both the Laplace approximation and ADVI do a reasonable job of matching to MCMC, though LR-ADVI is slightly more accurate for standard deviations.

---

3We calculated $H_{\eta\eta}$ using a custom branch of Stan’s automatic differentiation software [Carpenter et al., 2015] that exposes Hessians and Hessian-vector products in the Rstan modelfit class. When this custom branch is merged with the main branch of Stan, it will be possible to implement LR-ADVI for generic Stan models.
Sesame Street model accuracy

Next, we show results for `sesame_street1`, an analysis of a randomized controlled trial designed to estimate the causal effect of watching the television show Sesame Street on a letter-recognition test. To control for different conditions in the trials, a hierarchical model is used with correlated multivariate outcomes and unknown...
covariance structure. The model and data are described in detail in Gelman and Hill [2006, Chapter 23].

As can be seen in Fig. 2.8, the MAP under-estimates the variability of the random effects $a_g$, and, in turn, under-estimates the variance parameter $\sigma_a$. Because the MAP estimate of $\sigma_a$ is close to zero, the log posterior has a very high curvature with respect to the parameter $a_g$ at the MAP, and the Hessian used for the Laplace approximation is numerically singular. ADVI, which integrates out the uncertainty in the random effects, provides reasonably good estimates of the posterior means but underestimates the posterior standard deviations due to the mean-field assumption. Only LR-ADVI provides accurate estimates of posterior uncertainty.

**Radon model accuracy**

We now turn to radon_vary_intercept_floor, a hierarchical model of radon levels in Minnesota homes described in Gelman and Hill [2006, Chapters 16 and 21]. This model is relatively simple, with univariate normal observations and unknown variances. Nevertheless, the Laplace approximation again produces a numerically singular covariance matrix. The ADVI means are reasonably accurate, but the standard deviations are not. Only LR-ADVI produces an accurate approximation to the MCMC posterior standard deviations.
CHAPTER 2. COVARIANCES AND ROBUSTNESS

Finally, we consider a more complicated mark-recapture model from ecology known as the Cormack-Jolly-Seber (CJS) model. This model is described in detail in Kéry and Schaub [2011, Chapter 7], and discussion of the Stan implementation can be found in Stan Team [2015, Section 15.3].

The Laplace approximation is again degenerate, and the ADVI standard deviations again deviate considerably from MCMC. In this case, the ADVI means are also somewhat inaccurate, and some of the LR-ADVI standard deviations are mis-estimated in turn. However, LR-ADVI remains by far the most accurate method for approximating the MCMC standard errors.

Timing results

Detailed timing results for the ADVI experiments are shown in Fig. (2.11). Both the Laplace approximation and ADVI alone are faster than LR-ADVI, which in turn is about five times faster than MCMC. We achieved the best results optimizing $\tilde{KL}(\eta)$ by using the conjugate gradient Newton’s trust region method (trust-ncg of scipy.optimize), but the optimization procedure still accounted for an appreciable proportion of the time needed for LR-ADVI.
We now apply our methods to a real-world data set using a logistic regression with random effects, which is an example of a generalized linear mixed model (GLMM) [Agresti and Kateri, 2011, Chapter 13]. This data and model have several advantages as an illustration of our methods: the data set is large, the model contains a large number of imprecisely-estimated latent variables (the unknown random effects), the model exhibits the sparsity of $H_{\eta\eta}$ that is typical in many MFVB applications, and the results exhibit the same shortcomings of the Laplace approximation seen above. For this model, we will evaluate both posterior covariances and prior sensitivities.

**Criteo dataset**

We now apply our methods to a real-world data set using a logistic regression with random effects, which is an example of a generalized linear mixed model (GLMM) [Agresti and Kateri, 2011, Chapter 13]. This data and model have several advantages as an illustration of our methods: the data set is large, the model contains a large number of imprecisely-estimated latent variables (the unknown random effects), the model exhibits the sparsity of $H_{\eta\eta}$ that is typical in many MFVB applications, and the results exhibit the same shortcomings of the Laplace approximation seen above. For this model, we will evaluate both posterior covariances and prior sensitivities.

**Data and model**

We investigated a custom subsample of the 2014 Criteo Labs conversion logs data set [Criteo Labs, 2014], which contains an obfuscated sample of advertising data collected by Criteo over a period of two months. Each row of the data set corresponds to a single user click on an online advertisement. For each click, the data set records a binary outcome variable representing whether or not the user subsequently “converted” (i.e., performed a desired task, such as purchasing a product or signing up for a mailing list). Each row contains two timestamps (which we ignore), eight numerical covariates, and nine factor-valued covariates. Of the eight numerical covariates, three contain 30% or more missing data, so we discarded them. We
then applied a per-covariate normalizing transform to the distinct values of those remaining. Among the factor-valued covariates, we retained only the one with the largest number of unique values and discarded the others. These data-cleaning decisions were made for convenience. The goal of the present paper is to demonstrate our inference methods, not to draw conclusions about online advertising.

Although the meaning of the covariates has been obfuscated, for the purpose of discussion we will imagine that the single retained factor-valued covariate represents the identity of the advertiser, and the numeric covariates represent salient features of the user and/or the advertiser (e.g., how often the user has clicked or converted in the past, a machine learning rating for the advertisement quality, etc.). As such, it makes sense to model the probability of each row’s binary outcome (whether or not the user converted) as a function of the five numeric covariates and the advertiser identity using a logistic GLMM. Specifically, we observe binary conversion outcomes, $y_{it}$, for click $i$ on advertiser $t$, with probabilities given by observed numerical explanatory variables, $x_{it}$, each of which are vectors of length $K_x = 5$. Additionally, the outcomes within a given value of $t$ are correlated through an unobserved random effect, $u_t$, which represents the “quality” of advertiser $t$, where the value of $t$ for each observation is given by the factor-valued covariate. The random effects $u_t$ are assumed to follow a normal distribution with unknown mean and variance. Formally,

$$y_{it} \mid p_{it} \sim \text{Bernoulli}(p_{it}), \text{ for } t = 1, ..., T \text{ and } i = 1, ..., N_t$$

$$p_{it} := \frac{e^{\rho_{it}}}{1 + e^{\rho_{it}}} \text{ where } \rho_{it} := x_{it}^T \beta + u_t$$

$$u_t \mid \mu, \tau \sim \mathcal{N}(\mu, \tau^{-1})$$

Consequently, the unknown parameters are $\theta = (\beta^T, \mu, \tau, u_1, ..., u_T)^T$. We use the following priors:

$$\mu \mid \mu_0, \tau_\mu \sim \mathcal{N}(\mu_0, \tau_\mu^{-1})$$

$$\tau \mid \alpha_\tau, \beta_\tau \sim \text{Gamma}(\alpha_\tau, \beta_\tau)$$

$$\beta \mid \beta_0, \tau_\beta, \gamma_\beta \sim \mathcal{N}\left(\begin{pmatrix} \beta_0 \\ \vdots \\ \beta_0 \end{pmatrix}, \begin{pmatrix} \tau_\beta & \gamma_\beta & \gamma_\beta \\ \gamma_\beta & \ddots & \gamma_\beta \\ \gamma_\beta & \gamma_\beta & \tau_\beta \end{pmatrix}^{-1}\right)$$

Note that we initially take $\gamma_\beta = 0$ so that the prior information matrix on $\beta$ is diagonal. Nevertheless, by retaining $\gamma_\beta$ as a hyperparameter we will be able to assess the sensitivity to the assumption of a diagonal prior in Section 2.4. The remaining prior values are given in Appendix A.6. It is reasonable to expect that a
modeler would be interested both in the effect of the numerical covariates and in the
goodness of individual advertisers themselves, so we take the parameter of interest to
be \( g(\theta) = (\beta^T, u_1, \ldots, u_T)^T \).

To produce a data set small enough to be amenable to MCMC but large and
sparse enough to demonstrate our methods, we subsampled the data still further.
We randomly chose 5000 distinct advertisers to analyze, and then subsampled each
selected advertiser to contain no more than 20 rows each. The resulting data set had
\( N = 61895 \) total rows. If we had more observations per advertiser, the “random ef-
facts” \( u_t \) would have been estimated quite precisely, and the nonlinear nature of the
problem would not have been important; these changes would thus have obscured
the benefits of using MFVB versus the Laplace approximation. In typical internet
data sets a large amount of data comes from advertisers with few observations each,
so our subsample is representative of practically interesting problems.

Inference and timing

<table>
<thead>
<tr>
<th>Method</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP (optimum only)</td>
<td>12</td>
</tr>
<tr>
<td>VB (optimum only)</td>
<td>57</td>
</tr>
<tr>
<td>VB (including sensitivity for ( \beta ))</td>
<td>104</td>
</tr>
<tr>
<td>VB (including sensitivity for ( \beta ) and ( u ))</td>
<td>553</td>
</tr>
<tr>
<td>MCMC (Stan)</td>
<td>21066</td>
</tr>
</tbody>
</table>

Table 2.1: Timing results

We estimated the expectation and covariance of \( g(\theta) \) using four techniques:
MCMC, the Laplace approximation, MFVB, and linear response (LRVB) methods. For MCMC, we used Stan \[Stan Team, 2015\], and to calculate the MFVB, Laplace,
and LRVB estimates we used our own Python code using numpy, scipy, and autograd \[Jones et al., 2001, Maclaurin et al., 2015\]. As described in Section 2.4, the MAP
estimator did not estimate \( \mathbb{E}_{p_\theta}[g(\theta)] \) very well, so we do not report standard deviations or sensitivity measures for the Laplace approximations. The summary of
the computation time for all these methods is shown in Table 2.1 on page 47, with
details below.

For the MCMC estimates, we used Stan to draw 5000 MCMC draws (not in-
cluding warm-up), which took 351 minutes. We estimated all the prior sensitivities
of Section 2.4 using the Monte Carlo version of the covariance in Eq. (2.5).
For the MFVB approximation, we use the following mean field exponential family approximations:

\[ q(\beta_k) = \mathcal{N}(\beta_k; \eta_{\beta_k}), \text{ for } k = 1, \ldots, K_x \]
\[ q(u_t) = \mathcal{N}(u_t; \eta_{u_t}), \text{ for } t = 1, \ldots, T \]
\[ q(\tau) = \text{Gamma}(\tau; \eta_\tau) \]
\[ q(\mu) = \mathcal{N}(\mu; \eta_\mu) \]
\[ q(\theta) = q(\tau) q(\mu) \prod_{k=1}^{K_x} q(\beta_k) \prod_{t=1}^{T} q(u_t). \]

With these choices, evaluating the variational objective requires the following intractable univariate variational expectation:

\[ \mathbb{E}_{q(\theta; \eta)} [\log (1 - p_{it})] = \mathbb{E}_{q(\theta; \eta)} \left[ \log \left( 1 - \frac{e^{p_{it}}}{1 + e^{p_{it}}} \right) \right]. \]

We used the re-parameterization trick and four points of Gauss-Hermite quadrature to estimate this integral for each observation. See Appendix A.6 for more details.

We optimized the variational objective using the conjugate gradient Newton’s trust region method, trust-ncg, of scipy.optimize. One advantage of trust-ncg is that it performs second-order optimization but requires only Hessian-vector products, which can be computed quickly by autograd without constructing the full Hessian. The MFVB fit took 57 seconds, roughly 370 times faster than MCMC with Stan.

With variational parameters for each random effect \( u_t \), \( \mathbf{H}_{\eta\eta} \) is a 10014 x 10014 dimensional matrix. Consequently, evaluating \( \mathbf{H}_{\eta\eta} \) directly as a dense matrix using autograd would have been prohibitively time-consuming. Fortunately, our model can be decomposed into global and local parameters, and the Hessian term \( \mathbf{H}_{\eta\eta} \) in Theorem 2.2 is extremely sparse. In the notation of Section 2.3, take \( \theta_{\text{glob}} = (\beta^T, \mu, \tau)^T \), take \( \theta_{\text{loc}, t} = u_t \), and stack the variational parameters as \( \eta = (\eta_{\text{glob}}^T, \eta_{\text{loc}, 1}, \ldots, \eta_{\text{loc}, T})^T \). The cross terms in \( \mathbf{H}_{\eta\eta} \) between the local variables vanish:

\[ \frac{\partial^2 \text{KL}(q(\theta; \eta) \mid \mid p(\theta))}{\partial \eta_{\text{loc}, t_1} \partial \eta_{\text{loc}, t_2}} = 0 \text{ for all } t_1 \neq t_2. \]

Equivalently, note that the full likelihood in Appendix A.6, Eq. (A.4), has no cross terms between \( u_{t_1} \) and \( u_{t_2} \) for \( t_1 \neq t_2 \). As the dimension \( T \) of the data grows, so does the length of \( \eta \). However, the dimension of \( \eta_{\text{glob}} \) remains constant, and \( \mathbf{H}_{\eta\eta} \) remains easy to invert. We show an example of the sparsity pattern of the first few rows and columns of \( \mathbf{H}_{\eta\eta} \) in Fig. (2.12) .
CHAPTER 2. COVARIANCES AND ROBUSTNESS

Taking advantage of this sparsity pattern, we used autograd to calculate the Hessian of the KL divergence one group at a time and assembled the results in a sparse matrix using the scipy.sparse Python package. Even so, calculating the entire sparse Hessian took 323 seconds, and solving the system $H^{-1}_{\eta\eta} g^\top_\eta$ using scipy.sparse.linalg.spsolve took an additional 173 seconds. These results show that the evaluation and inversion of $H_{\eta\eta}$ was several times more costly than optimizing the variational objective itself. (Of course, the whole procedure remains much faster than running MCMC with Stan.)

We note, however, that instead of the direct approach to calculating $H^{-1}_{\eta\eta} g^\top_\eta$ one can use the conjugate gradient algorithm of sp.sparse.linalg.cg [Wright and NoCEDAL 1999, Chapter 5] together with the fast Hessian-vector products of autograd to query one column at a time of $H^{-1}_{\eta\eta} g^\top_\eta$. On a typical column of $H^{-1}_{\eta\eta} g^\top_\eta$ in our experiment, calculating the conjugate gradient took only 9.4 seconds (corresponding to 81 Hessian-vector products in the conjugate gradient algorithm). Thus, for example, one could calculate the columns of $H^{-1}_{\eta\eta} g^\top_\eta$ corresponding to the expectations of the global variables $\beta$ in only $9.4 \times K_x = 469$ seconds, which is much less time than it would take to compute the entire $H^{-1}_{\eta\eta} g^\top_\eta$ for both $\beta$ and every random effect in $u$.

For the Laplace approximation, we calculated the MAP estimator and $H_{\text{Lap}}$ using Python code similar to that used for the MFVB estimates. We observe that the MFVB approximation to posterior means would be expected to improve on the MAP estimator only in cases when there is both substantial uncertainty in some parameters and when this uncertainty, through nonlinear dependence between parameters, affects the values of posterior means. These circumstances obtain in the logistic GLMM model with sparse per-advertiser data since the random effects $u_t$.

Figure 2.12: Sparsity pattern of top-left sub-matrix of $H_{\eta\eta}$ for the logit GLMM model. The axis numbers represent indices within $\eta$, and black indicates non-zero entries of $H_{\eta\eta}$.
CHAPTER 2. COVARIANCES AND ROBUSTNESS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MCMC</th>
<th>MFVB</th>
<th>MAP</th>
<th>MCMC std. err.</th>
<th>MCMC Eff. #</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta_1 )</td>
<td>1.454</td>
<td>1.447</td>
<td>1.899</td>
<td>0.02067</td>
<td>33</td>
</tr>
<tr>
<td>( \beta_2 )</td>
<td>0.031</td>
<td>0.033</td>
<td>0.198</td>
<td>0.00025</td>
<td>5000</td>
</tr>
<tr>
<td>( \beta_3 )</td>
<td>0.110</td>
<td>0.110</td>
<td>0.103</td>
<td>0.00028</td>
<td>5000</td>
</tr>
<tr>
<td>( \beta_4 )</td>
<td>-0.172</td>
<td>-0.173</td>
<td>-0.173</td>
<td>0.00016</td>
<td>5000</td>
</tr>
<tr>
<td>( \beta_5 )</td>
<td>0.273</td>
<td>0.273</td>
<td>0.280</td>
<td>0.00042</td>
<td>5000</td>
</tr>
<tr>
<td>( \mu )</td>
<td>2.041</td>
<td>2.041</td>
<td>3.701</td>
<td>0.04208</td>
<td>28</td>
</tr>
<tr>
<td>( \tau )</td>
<td>0.892</td>
<td>0.823</td>
<td>827.724</td>
<td>0.00051</td>
<td>1232</td>
</tr>
<tr>
<td>( u_{1431} )</td>
<td>1.752</td>
<td>1.757</td>
<td>3.700</td>
<td>0.00937</td>
<td>5000</td>
</tr>
<tr>
<td>( u_{4150} )</td>
<td>1.217</td>
<td>1.240</td>
<td>3.699</td>
<td>0.01022</td>
<td>5000</td>
</tr>
<tr>
<td>( u_{4575} )</td>
<td>2.427</td>
<td>2.413</td>
<td>3.702</td>
<td>0.00936</td>
<td>5000</td>
</tr>
<tr>
<td>( u_{4685} )</td>
<td>3.650</td>
<td>3.633</td>
<td>3.706</td>
<td>0.00862</td>
<td>5000</td>
</tr>
</tbody>
</table>

Table 2.2: Results for the estimation of the posterior means

will be quite uncertain and the other posterior means depend on them through the nonlinear logistic function.

**Posterior approximation results**

In this section, we assess the accuracy of the MFVB, Laplace, and LRVB methods as approximations to \( \mathbb{E}_{p_0}[g(\theta)] \) and \( \text{Cov}_{p_0}(g(\theta)) \). We take the MCMC estimates as ground truth. Although, as discussed in Section 2.4, we are principally interested in the parameters \( g(\theta) = (\beta^T, u_1, \ldots, u_T)^T \), we will report the results for all parameters for completeness. For readability, the tables and graphs show results for a random selection of the components of the random effects \( u \).

**Posterior means**

We begin by comparing the posterior means in Table 2.2 on page 50, Fig. (2.13), and Fig. (2.14). We first note that, despite the long running time for MCMC, the \( \beta_1 \) and \( \mu \) parameters did not mix well in the MCMC sample, as is reflected in the MCMC standard error and effective number of draws columns of Table 2.2 on page 50. The \( x_{it} \) data corresponding to \( \beta_1 \) contained fewer distinct values than the other columns of \( x \), which perhaps led to some co-linearity between \( \beta_1 \) and \( \mu \) in the posterior. This co-linearity could have caused both poor MCMC mixing and, perhaps, excessive measured prior sensitivity, as discussed below in Section 2.4. Although we will report the results for both \( \beta_1 \) and \( \mu \) without further comment, the
reader should bear in mind that the MCMC “ground truth” for these two parameters is somewhat suspect.

The results in Table 2.2 on page 50 and Fig. 2.13 show that MFVB does an excellent job of approximating the posterior means in this particular case, even for the random effects $u$ and the related parameters $\mu$ and $\tau$. In contrast, the MAP estimator does reasonably well only for certain components of $\beta$ and does extremely poorly for the random effects parameters. As can be seen in Fig. 2.14, the MAP estimate dramatically overestimates the information $\tau$ of the random effect distribution (that is, it underestimates the variance). As a consequence, it estimates all the random effects to have essentially the same value, leading to mis-estimation of some location parameters, including both $\mu$ and some components of $\beta$. Since the MAP estimator performed so poorly at estimating the random effect means, we will not consider it any further.
CHAPTER 2. COVARIANCES AND ROBUSTNESS

Parameter MCMC LRVB Uncorrected MFVB

\( \beta_1 \) 
0.118 0.103 0.005

\( \beta_2 \) 
0.018 0.018 0.004

\( \beta_3 \) 
0.020 0.020 0.004

\( \beta_4 \) 
0.012 0.012 0.004

\( \beta_5 \) 
0.029 0.030 0.004

\( \mu \) 
0.223 0.192 0.016

\( \tau \) 
0.018 0.033 0.016

\( u_{1431} \) 
0.663 0.649 0.605

\( u_{4150} \) 
0.723 0.707 0.662

\( u_{4685} \) 
0.662 0.649 0.615

\( u_{4685} \) 
0.610 0.607 0.579

Table 2.3: Standard deviation results

Posterior covariances

We now assess the accuracy of our estimates of \( \text{Cov}_{p_0}(g(\theta)) \). The results for the marginal standard deviations are shown in Table 2.3 on page 52 and Fig. 2.15. We refer to the standard deviations of \( \text{Cov}_{q_0}(g(\theta)) \) as the “uncorrected MFVB” estimate, and of \( \text{Cov}_{q_0}^{LR}(g(\theta)) \) as the “LRVB” estimate. The uncorrected MFVB variance estimates of \( \beta \) are particularly inaccurate, but the LRVB variances match
In Fig. (2.16), we compare the off-diagonal elements of $\text{Cov}^{LR}_{q_0}(g(\theta))$ and $\text{Cov}_{p_0}(g(\theta))$. These covariances are zero, by definition, in the uncorrected MFVB estimates $\text{Cov}_{q_0}(g(\theta))$. The left panel of Fig. (2.16) shows the estimated covariances between the global parameters and all other parameters, including the random effects, and the right panel shows only the covariances amongst the random effects. The LRVB covariances are quite accurate, particularly when we recall that the MCMC draws of $\mu$ may be inaccurate due to poor mixing.

### Table 2.4: MFVB normalized prior sensitivity results

<table>
<thead>
<tr>
<th></th>
<th>$\beta_0$</th>
<th>$\tau_\beta$</th>
<th>$\gamma_\beta$</th>
<th>$\mu_0$</th>
<th>$\tau_\mu$</th>
<th>$\alpha_\tau$</th>
<th>$\beta_\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>0.0094</td>
<td>-0.1333</td>
<td>-0.0510</td>
<td>0.0019</td>
<td>-0.3920</td>
<td>0.0058</td>
<td>-0.0048</td>
</tr>
<tr>
<td>$\tau$</td>
<td>0.0009</td>
<td>-0.0086</td>
<td>-0.0142</td>
<td>0.0003</td>
<td>-0.0575</td>
<td>0.0398</td>
<td>-0.0328</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>0.0089</td>
<td>-0.1464</td>
<td>-0.0095</td>
<td>0.0017</td>
<td>-0.3503</td>
<td>0.0022</td>
<td>-0.0018</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>0.0012</td>
<td>-0.0143</td>
<td>-0.0113</td>
<td>0.0003</td>
<td>-0.0516</td>
<td>0.0062</td>
<td>-0.0051</td>
</tr>
<tr>
<td>$\beta_3$</td>
<td>-0.0035</td>
<td>0.0627</td>
<td>-0.0081</td>
<td>-0.0006</td>
<td>0.1218</td>
<td>-0.0003</td>
<td>0.0002</td>
</tr>
<tr>
<td>$\beta_4$</td>
<td>0.0018</td>
<td>-0.0037</td>
<td>-0.0540</td>
<td>0.0004</td>
<td>-0.0835</td>
<td>0.0002</td>
<td>-0.0002</td>
</tr>
<tr>
<td>$\beta_5$</td>
<td>0.0002</td>
<td>0.0308</td>
<td>-0.0695</td>
<td>0.0002</td>
<td>-0.0383</td>
<td>0.0011</td>
<td>-0.0009</td>
</tr>
<tr>
<td>$u_{1431}$</td>
<td>0.0028</td>
<td>-0.0397</td>
<td>-0.0159</td>
<td>0.0006</td>
<td>-0.1169</td>
<td>0.0018</td>
<td>-0.0015</td>
</tr>
<tr>
<td>$u_{4150}$</td>
<td>0.0026</td>
<td>-0.0368</td>
<td>-0.0146</td>
<td>0.0005</td>
<td>-0.1083</td>
<td>0.0022</td>
<td>-0.0018</td>
</tr>
<tr>
<td>$u_{4575}$</td>
<td>0.0028</td>
<td>-0.0406</td>
<td>-0.0138</td>
<td>0.0006</td>
<td>-0.1153</td>
<td>0.0011</td>
<td>-0.0009</td>
</tr>
<tr>
<td>$u_{4685}$</td>
<td>0.0028</td>
<td>-0.0409</td>
<td>-0.0142</td>
<td>0.0006</td>
<td>-0.1163</td>
<td>0.0003</td>
<td>-0.0002</td>
</tr>
</tbody>
</table>
Finally, we compare the MFVB prior sensitivity measures of Section 2.3 to the covariance-based MCMC sensitivity measures of Section 2.1. Since sensitivity is of practical interest only when it is of comparable order to the posterior uncertainty, we report sensitivities normalized by the appropriate standard deviation. That is, we report \( \hat{S}_{\alpha_0} / \sqrt{\text{diag} \left( \text{Cov}_{q_0} \left( g \left( \theta \right) \right) \right)} \), and \( S_{\alpha_0}^q / \sqrt{\text{diag} \left( \text{Cov}_{q_0}^{LR} \left( g \left( \theta \right) \right) \right)} \), etc., where \( \text{diag} (\cdot) \) denotes the diagonal vector of a matrix, and the division is element-wise. Note that we use the sensitivity-based variance estimates \( \text{Cov}_{q_0}^{LR} \), not the uncorrected MFVB estimates \( \text{Cov}_{q_0} \), to normalize the variational sensitivities. We refer to a sensitivity divided by a standard deviation as a “normalized” sensitivity.

The comparison between the MCMC and MFVB sensitivity measures is shown in Fig. (2.17). The MFVB and MCMC sensitivities correspond very closely, though the MFVB means appear to be slightly more sensitive to the prior parameters than
the MCMC means. This close correspondence should not be surprising. As shown in Section 2.4, the MFVB and MCMC posterior means match quite closely. If we assume, reasonably, that they continue to match to first order in a neighborhood of our original prior parameters, then Condition 2.1 will hold and we would expect \( \hat{S}_{\alpha_0} \approx S_{\alpha_0}^q \).

Table 2.4 on page 53 shows the detailed MFVB normalized sensitivity results. Each entry is the sensitivity of the MFVB mean of the row’s parameter to the column’s prior parameter. One can see that several parameters are quite sensitive to the information parameter prior \( \tau_\mu \). In particular, \( \mathbb{E}_{\rho_0}\{\mu\} \) and \( \mathbb{E}_{\rho_0}\{\beta_1\} \) are expected to change approximately \(-0.39\) and \(-0.35\) standard deviations, respectively, for every unit change in \( \tau_\mu \). This size of change could be practically significant (assuming that such a change in \( \tau_\mu \) is subjectively plausible). To investigate this sensitivity further, we re-fit the MFVB model at a range of values of the prior parameter \( \tau_\mu \), assessing the accuracy of the linear approximation to the sensitivity. The results are shown in Fig. (2.18). Even for very large changes in \( \tau_\mu \)—resulting in changes to \( \mathbb{E}_{\rho_0}\{\mu\} \) and \( \mathbb{E}_{\rho_0}\{\beta_1\} \) far in excess of two standard deviations—the linear approximation holds up reasonably well. Fig. (2.18) also shows a (randomly selected) random effect to be quite sensitive, though not to a practically important degree relative to its posterior standard deviation. The insensitivity of \( \mathbb{E}_{\rho_0}\{\beta_2\} \) is also confirmed. Of course, the accuracy of the linear approximation cannot be guaranteed to hold as well in general as it does in this particular case, and the quick and reliable evaluation of the linearity assumption without re-fitting the model remains interesting future work.

Since we started the MFVB optimization close to the new, perturbed optimum, each new MFVB fit took only 27.2 seconds on average. Re-estimating the MCMC posterior so many times would have been extremely time-consuming. (Note that importance sampling would be useless for prior parameter changes that moved the posterior so far from the original draws.) The considerable sensitivity of this model to a particular prior parameter, which is perhaps surprising on such a large data set, illustrates the value of having fast, general tools for discovering and evaluating prior sensitivity. Our framework provides just such a set of tools.

### 2.5 Proof of Theorem 2.1

In this section we prove Theorem 2.1.

**Proof.** Under Assumption 2.1, we can exchange differentiation and integration in \( \frac{\partial}{\partial \alpha} \int p_0(\theta) \exp(\rho(\theta, \alpha)) g(\theta) \lambda(d\theta) \) and \( \frac{\partial}{\partial \alpha} \int p_0(\theta) \exp(\rho(\theta, \alpha)) \lambda(d\theta) \) by Fleming [1965, Chapter 5-11, Theorem 18], which ultimately depends on the Lebesgue dominated convergence theorem. By Assumption 2.1 \( \mathbb{E}_{\rho_0}\{g(\theta)\} \) is
well-defined for \( \alpha \in A_0 \) and
\[
\frac{\partial p_0 (\theta) \exp (\rho (\theta, \alpha))}{\partial \alpha} = p_0 (\theta) \exp (\rho (\theta, \alpha)) \frac{\partial \rho (\theta, \alpha)}{\partial \alpha} \quad \lambda\text{-almost everywhere.}
\]

Armed with these facts, we can directly compute
\[
\frac{dE_{p_0} [g (\theta)]}{d\alpha^\top} \bigg|_{\alpha_0} = -\frac{\partial}{\partial \alpha} \int p_0 (\theta) \exp (\rho (\theta, \alpha)) \lambda (d\theta) \bigg|_{\alpha_0}
\]
\[
\text{E}_{p_0} [g (\theta)] \frac{\partial}{\partial \alpha} \int p_0 (\theta) \exp (\rho (\theta, \alpha)) \lambda (d\theta) \bigg|_{\alpha_0}
\]
\[
\int g (\theta) p_0 (\theta) \exp (\rho (\theta, \alpha)) \frac{\partial \rho (\theta, \alpha)}{\partial \alpha} \bigg|_{\alpha_0} \lambda (d\theta) - \text{E}_{p_0} [g (\theta)] \text{E}_{p_0} \left[ \frac{\partial \rho (\theta, \alpha)}{\partial \alpha} \bigg|_{\alpha_0} \right]
\]
\[
= \text{Cov}_{p_0} \left( g (\theta), \frac{\partial \rho (\theta, \alpha)}{\partial \alpha} \bigg|_{\alpha_0} \right).
\]

\[\square\]

### 2.6 Comparison with MCMC importance sampling

In this section, we show that using importance sampling with MCMC samples to calculate the local sensitivity in Eq. (2.1) is precisely equivalent to using the same MCMC samples to estimate the covariance in Eq. (2.4) directly. For this section, will suppose that Assumption 2.1 holds. Further suppose, without loss of generality, we have samples \( \theta_i \) drawn IID from \( p_0 (\theta) \):
\[
\theta_n \overset{iid}{\sim} p_0 (\theta), \quad \text{for } n = 1, ..., N_s
\]
\[
\text{E}_{p_0} [g (\theta)] \approx \frac{1}{N_s} \sum_{n=1}^{N_s} g (\theta_n).
\]

Typically we cannot compute the dependence of the normalizing constant \( \int p (\theta') \exp (\rho (\theta', \alpha)) \lambda (d\theta') \) on \( \alpha \), so we use the following importance sampling
estimate for $\mathbb{E}_{p_\alpha} [g (\theta)]$ [Owen 2013, Chapter 9]:

$$w_n = \exp (\rho (\theta_n, \alpha) - \rho (\theta_n, \alpha_0))$$

$$\tilde{w}_n := \frac{w_n}{\sum_{n'=1}^{N_s} w_{n'}}$$

$$\mathbb{E}_{p_\alpha} [g (\theta)] \approx \sum_{n=1}^{N_s} \tilde{w}_n g (\theta_n).$$

Note that $\tilde{w}_n |_{\alpha_0} = \frac{1}{N_s}$, so the importance sampling estimate recovers the ordinary sample mean at $\alpha_0$. The derivatives of the weights are given by

$$\frac{\partial w_n}{\partial \alpha} = w_n \frac{\partial \rho (\theta_n, \alpha)}{\partial \alpha}$$

$$\frac{\partial \tilde{w}_n}{\partial \alpha} = \frac{\partial w_n}{\partial \alpha} - \frac{w_n N_s \sum_{n'=1}^{N_s} \frac{\partial w_{n'}}{\partial \alpha}}{\left(\sum_{n'=1}^{N_s} w_{n'}\right)^2}$$

$$= \frac{w_n}{\sum_{n'=1}^{N_s} w_{n'}} \frac{\partial \rho (\theta_n, \alpha)}{\partial \alpha} - \frac{w_n}{\sum_{n'=1}^{N_s} w_{n'}} \sum_{n'=1}^{N_s} \frac{w_n}{\sum_{n'=1}^{N_s} w_{n'}} \frac{\partial \rho (\theta_{n'}, \alpha)}{\partial \alpha}$$

$$= \tilde{w}_n \frac{\partial \rho (\theta_n, \alpha)}{\partial \alpha} - \tilde{w}_n \sum_{n'=1}^{N_s} \tilde{w}_{n'} \frac{\partial \rho (\theta_{n'}, \alpha)}{\partial \alpha}.$$

It follows that

$$\frac{\partial}{\partial \alpha} \sum_{n=1}^{N_s} \tilde{w}_n g (\theta_n) \bigg|_{\alpha_0} = \sum_{n=1}^{N_s} \left( \tilde{w}_n \frac{\partial \rho (\theta_n, \alpha)}{\partial \alpha} - \tilde{w}_n \sum_{n'=1}^{N_s} \tilde{w}_{n'} \frac{\partial \rho (\theta_{n'}, \alpha)}{\partial \alpha} \right) \bigg|_{\alpha_0} g (\theta_n)$$

$$= \frac{1}{N_s} \sum_{n=1}^{N_s} \frac{\partial \rho (\theta_n, \alpha)}{\partial \alpha} \bigg|_{\alpha_0} g (\theta_n) -$$

$$\left[ \frac{1}{N_s} \sum_{n=1}^{N_s} \frac{\partial \rho (\theta_n, \alpha)}{\partial \alpha} \bigg|_{\alpha_0} \right] \left[ \frac{1}{N_s} \sum_{n=1}^{N_s} g (\theta_n) \right],$$

which is precisely the sample version of the covariance in Theorem 2.1.

### 2.7 Our use of the terms “sensitivity” and “robustness”

In this section we clarify our usage of the terms “robustness” and “sensitivity.” The quantity $S^T_{\alpha_0} (\alpha - \alpha_0)$ measures the sensitivity of $\mathbb{E}_{p_\alpha} [g (\theta)]$ to perturbations
in the direction $\Delta \alpha$. Intuitively, as sensitivity increases, robustness decreases, and, in this sense, sensitivity and robustness are opposites of one another. However, we emphasize that sensitivity is a clearly defined, measurable quantity and that robustness is a subjective judgment informed by sensitivity, but also by many other less objective considerations.

Suppose we have calculated $S_{\alpha_0}$ from Eq. (2.1) and found that it has a particular value. To determine whether our model is robust, we must additionally decide

1. How large of a change in the prior, $||\alpha - \alpha_0||$, is plausible, and
2. How large of a change in $E_{p_\alpha}[g(\theta)]$ is important.

The set of plausible prior values necessarily remains a subjective decision. Whether or not a particular change in $E_{p_\alpha}[g(\theta)]$ is important depends on the ultimate use of the posterior mean. For example, the posterior standard deviation can be a guide: if the prior sensitivity is swamped by the posterior uncertainty then it can be neglected when reporting our subjective uncertainty about $g(\theta)$, and the model is robust. Similarly, even if the prior sensitivity is much larger than the posterior standard deviation but small enough that it would not affect any actionable decision made on the basis of the value of $E_{p_\alpha}[g(\theta)]$, then the model is robust. Intermediate values remain a matter of judgment. An illustration of the relationship between sensitivity and robustness is shown in Fig. (2.19).

Finally, we note that if $\mathcal{A}$ is small enough that $E_{p_\alpha}[g(\theta)]$ is roughly linear in $\alpha$ for $\alpha \in \mathcal{A}$, then calculating Eq. (2.1) for all $\alpha \in \mathcal{A}$ and finding the worst case

\footnote{This decision can be cast in a formal decision theoretic framework based on a partial ordering of subjective beliefs [Insua and Criado 2000].}
can be thought of as a first-order approximation to a global robustness estimate. Depending on the problem at hand, this linearity assumption may not be plausible except for very small \( A \). This weakness is inherent to the local robustness approach. Nevertheless, even when the perturbations are valid only for a small \( A \), these easily-calculable measures may still provide valuable intuition about the potential modes of failure for a model.

If \( g(\theta) \) is a scalar, it is natural to attempt to summarize the high-dimensional vector \( S_{\alpha_0} \) in a single easily reported number such as

\[
S_{\alpha_0}^{\text{sup}} := \sup_{\alpha: \|\alpha - \alpha_0\| \leq 1} \left| S_{\alpha_0}^T (\alpha - \alpha_0) \right|.
\]

For example, the calculation of \( S_{\alpha_0}^{\text{sup}} \) is the principal ambition of Basu et al. [1996]. The use of such summaries is also particularly common in work that considers function-valued perturbations [e.g., Gustafson 1996b, Roos et al., 2015]. (Function-valued perturbations can be connected to the finite-dimensional perturbations of the present work through the notion of the Gateaux derivative [Huber, 2011, Chapter 2.5], the elaboration of which we leave to future work.) Although the summary \( S_{\alpha_0}^{\text{sup}} \) has obvious merits, in the present work we emphasize the calculation only of \( S_{\alpha_0} \) in the belief that its interpretation is likely to vary from application to application and require some critical thought and subjective judgment. For example, the unit ball \( \|\alpha - \alpha_0\| \leq 1 \) (as in Basu et al. [1996]) may not make sense as a subjective description of the range of plausible variability of \( p(\theta|\alpha) \). Consider, e.g.: why should the off-diagonal term of a Wishart prior plausibly vary as widely as the mean of some other parameter, when the two might not even have the same units? This problem is easily remedied by choosing an appropriate scaling of the parameters and thereby making the unit ball an appropriate range for the problem at hand, but the right scaling will vary from problem to problem and necessarily be a somewhat subjective choice, so we refrain from taking a stand on this decision. As another example, the worst-case function-valued perturbations of Gustafson [1996a,b] require a choice of a metric ball in function space whose meaning may not be intuitively obvious, may provide worst-case perturbations that depend on the data to a subjectively implausible degree, and may exhibit interesting but perhaps counter-intuitive asymptotic behavior for different norms and perturbation dimensions. Consequently, we do not attempt to prescribe a particular one-size-fits-all summary measure. The local sensitivity \( S_{\alpha_0} \) is a well-defined mathematical quantity. Its relationship to robustness must remain a matter of judgment.
2.8 Proof of Theorem 2.2

In this section we prove Theorem 2.2.

Proof. For notational convenience, we will define
\[ KL(\eta, \alpha) := KL(q(\theta; \eta) \| p_\alpha(\theta)). \]

By Assumption 2.3, \( \eta^*(\alpha) \) is both optimal and interior for all \( \alpha \in A_0 \), and by Assumption 2.2, \( KL(\eta, \alpha) \) is continuously differentiable in \( \eta \). Therefore, the first-order conditions of the optimization problem in Eq. (2.8) give:
\[
\frac{\partial KL(\eta, \alpha)}{\partial \eta} \bigg|_{\eta = \eta^*(\alpha)} = 0 \quad \text{for all} \quad \alpha \in A_0.
\]
(2.26)

\[
\frac{\partial^2 KL(\eta, \alpha)}{\partial \eta^2} \bigg|_{\eta = \eta^*(\alpha)} \text{ is positive definite by the strict optimality of } \eta^* \text{ in Assumption 2.3, and}
\]
\[ \frac{\partial^2 KL(\eta, \alpha)}{\partial \eta \partial \alpha^T} \bigg|_{\eta = \eta^*(\alpha)} \text{ is continuous by Assumption 2.2. It follows that } \eta^* \text{ is a continuously differentiable function of } \alpha \text{ by application of the implicit function theorem to the first-order condition in Eq. (2.26) [Fleming, 1965, Chapter 4.6]. So we can use the chain rule to take the total derivative of Eq. (2.26) with respect to } \alpha. \]

\[
\frac{d}{d\alpha} \left( \frac{\partial KL(\eta, \alpha)}{\partial \eta} \bigg|_{\eta = \eta^*(\alpha)} \right) = 0 \quad \text{for all } \alpha \in A_0 \Rightarrow
\]
\[
\frac{\partial^2 KL(\eta, \alpha)}{\partial \eta \partial \alpha^T} \bigg|_{\eta = \eta^*(\alpha)} \quad \text{and} \quad \frac{\partial^2 KL(\eta, \alpha)}{\partial \eta^2} \bigg|_{\eta = \eta^*(\alpha)} = 0 \quad \text{for all } \alpha \in A_0.
\]

The strict optimality of \( KL(\eta, \alpha) \) at \( \eta^*(\alpha) \) in Assumption 2.3 requires that \( \frac{\partial^2 KL(\eta, \alpha)}{\partial \eta^2} \bigg|_{\eta = \eta^*(\alpha)} \) be invertible. So we can evaluate at \( \alpha = \alpha_0 \) and solve to find that
\[
\frac{d\eta^*(\alpha)}{d\alpha^T} \bigg|_{\alpha_0} = -\left( \frac{\partial^2 KL(\eta, \alpha)}{\partial \eta \partial \alpha^T} \bigg|_{\eta = \eta^*(\alpha), \alpha = \alpha_0} \right)^{-1} \frac{\partial^2 KL(\eta, \alpha)}{\partial \eta \partial \alpha^T} \bigg|_{\eta = \eta^*_0, \alpha = \alpha_0}.
\]

\( E_{q_\alpha}[g(\theta)] \) is a continuously differentiable function of \( \eta^*(\alpha) \) by Assumption 2.4. So by the chain rule and Assumption 2.2, we have that
\[
\frac{dE_{q(\theta,\eta)}[g(\theta)]}{d\alpha^T} \bigg|_{\alpha_0} = \frac{\partial E_{q(\theta,\eta)}[g(\theta)]}{\partial \eta} \frac{d\eta^*(\alpha)}{d\alpha^T} \bigg|_{\eta = \eta^*_0, \alpha = \alpha_0}.
\]

Finally, we observe that
\[
KL(\eta, \alpha) = E_{q(\theta,\eta)}[\log q(\theta; \eta) - \log p(\theta) - \rho(\theta, \alpha)] + \text{Constant} \Rightarrow
\]
\[
\frac{\partial^2 KL(\eta, \alpha)}{\partial \eta \partial \alpha^T} \bigg|_{\eta = \eta^*_0, \alpha = \alpha_0} = -\left( \frac{\partial^2 E_{q(\theta,\eta)}[\rho(\theta, \alpha)]}{\partial \eta \partial \alpha^T} \bigg|_{\eta = \eta^*_0, \alpha = \alpha_0} \right).\]
Here, the term Constant contains quantities that do not depend on \( \eta \). Plugging in gives the desired result.

**2.9 Exactness of multivariate Normal posterior means**

In this section, we show that the MFVB estimate of the posterior means of a multivariate normal with known covariance is exact and that, as an immediate consequence, the linear response covariance recovers the exact posterior covariance, i.e.,

\[
\text{Cov}_{q_0}^{LR}(\theta) = \text{Cov}_{p_0}(\theta).
\]

Suppose we are using MFVB to approximate a non-degenerate multivariate normal posterior, i.e.,

\[
p_0(\theta) = \mathcal{N}(\theta; \mu, \Sigma)
\]

for full-rank \( \Sigma \). This posterior arises, for instance, given a multivariate normal likelihood \( p(x|\mu) = \prod_{n=1:N} \mathcal{N}(x_n|\theta, \Sigma_x) \) with known covariance \( \Sigma_x \) and a conjugate multivariate normal prior on the unknown mean parameter \( \theta \in \mathbb{R}^K \). Additionally, even when the likelihood is non-normal or the prior is not conjugate, the posterior may be closely approximated by a multivariate normal distribution when a Bayesian central limit theorem can be applied [Le Cam and Yang, 2012, Chapter 8].

We will consider an MFVB approximation to \( p_0(\theta) \). Specifically, let the elements of the vector \( \theta \) be given by scalars \( \theta_k \) for \( k = 1, ..., K \), and take the MFVB normal approximation with means \( m_k \) and variances \( v_k \):

\[
Q = \left\{ q(\theta) : q(\theta) = \prod_{k=1}^{K} \mathcal{N}(\theta_k; m_k, v_k) \right\}.
\]

In the notation of Eq. (2.9), we have \( \eta_k = (m_k, v_k)^T \) with

\[
\Omega_\eta = \{ \eta : v_k > 0, \forall k = 1, ..., K \}.
\]

The optimal variational parameters are given by \( \eta_k^* = (m_k^*, v_k^*)^T \).

**Lemma 2.1.** Let \( p_0(\theta) = \mathcal{N}(\theta; \mu, \Sigma) \) for full-rank \( \Sigma \) and let

\[
Q = \left\{ q(\theta) : q(\theta) = \prod_{k=1}^{K} \mathcal{N}(\theta_k; m_k, v_k) \right\}
\]
be the mean field approximating family. Then there exists an \( \eta^* = (m^*, v^*) \) that solves

\[
\eta^* = \arg\min_{\eta \in \mathcal{Q}} KL(q(\theta; \eta) \mid \mid p_\alpha(\theta))
\]

with \( m^* = \mu \).

Proof. Let \( \text{diag}(v) \) denote the \( K \times K \) matrix with the vector \( v \) on the diagonal and zero elsewhere. Using the fact that the entropy of a univariate normal distribution with variance \( v \) is \( \frac{1}{2} \log v \) plus a constant, the variational objective in Eq. (2.8) is given by

\[
KL(q(\theta; \eta) \mid \mid p_0(\theta)) = \mathbb{E}_{q(\theta; \eta)} \left[ \frac{1}{2} (\theta - \mu)^T \Sigma^{-1} (\theta - \mu) \right] = \frac{1}{2} \sum_k \log v_k + \text{Constant}
\]

\[
= \frac{1}{2} \text{trace} \left( \Sigma^{-1} \mathbb{E}_{q(\theta; \eta)} [\theta \theta^T] \right) - \mu^T \Sigma^{-1} \mathbb{E}_{q(\theta; \eta)} [\theta] - \frac{1}{2} \sum_k \log v_k + \text{Constant}
\]

\[
= \frac{1}{2} \text{trace} \left( \Sigma^{-1} (mm^T + \text{diag}(v)) \right) - \mu^T \Sigma^{-1} m - \frac{1}{2} \sum_k \log v_k + \text{Constant}
\]

\[
= \frac{1}{2} \text{trace} \left( \Sigma^{-1} \text{diag}(v) \right) + \frac{1}{2} m^T \Sigma^{-1} m - \mu^T \Sigma^{-1} m - \frac{1}{2} \sum_k \log v_k + \text{Constant}.
\] (2.27)

The first-order condition for the optimal \( m^* \) is then

\[
\frac{\partial KL(q(\theta; \eta) \mid \mid p_0(\theta))}{\partial m} \bigg|_{m=m^*, v=v^*} = 0 \Rightarrow \Sigma^{-1} m^* - \Sigma^{-1} \mu = 0 \Rightarrow m^* = \mu.
\]

The optimal variances follow similarly:

\[
\frac{\partial KL(q(\theta; \eta) \mid \mid p_0(\theta))}{\partial v_k} \bigg|_{m=m^*, v=v^*} = 0 \Rightarrow \frac{1}{2} (\Sigma^{-1})_{kk} - \frac{1}{2} \frac{1}{v_k} = 0 \Rightarrow v_k^* = \frac{1}{(\Sigma^{-1})_{kk}}.
\]
Since $v_k^* > 0$, we have $\eta^* \in \Omega_\eta$.

Lemma 2.1 can be also be derived via the variational coordinate ascent updates (Bishop [2006, Section 10.1.2] and Giordano et al. [2015, Appendix B]).

Next, we show that Lemma 2.1 holds for all perturbations of the form $\rho(\theta, \alpha) = \alpha^T \theta$ with $\alpha_0 = 0$ and that Assumptions 2.1–2.4 are satisfied for all finite $\alpha$.

**Lemma 2.2.** Under the conditions of Lemma 2.1, let $p_\alpha(\theta)$ be defined from Eq. (2.2) with $\rho(\theta, \alpha) = \alpha^T \theta$ and $\alpha_0 = 0$. Take $g(\theta) = \theta$. Then, for all finite $\alpha$, Assumptions 2.1–2.4 are satisfied, and Condition 2.1 is satisfied with equality.

**Proof.** Up to a constant that does not depend on $\theta$, the log density of $p_\alpha(\theta)$ is

$$\log p_\alpha(\theta) = -\frac{1}{2} (\theta - \mu)^T \Sigma^{-1} (\theta - \mu) + \alpha^T \theta + \text{Constant}$$

$$= -\frac{1}{2} \theta^T \Sigma^{-1} \theta - \frac{1}{2} \mu^T \Sigma^{-1} \mu + (\mu^T \Sigma^{-1} + \alpha^T) \theta + \text{Constant}. $$

Since $\theta$ is a natural sufficient statistic of the multivariate normal distribution and the corresponding natural parameter of $p_\alpha(\theta)$, $\Sigma^{-1} \mu + \alpha$, is interior when $\Sigma$ is full-rank, $p_\alpha(\theta)$ is multivariate normal for any finite $\alpha$. Assumption 2.1 follows immediately.

By inspection of Eq. (2.27), Assumption 2.2 is satisfied. Because $\Omega_\eta$ is an open set and $\Sigma$ is positive definite, Assumption 2.3 is satisfied. Since $E_{q(\theta, \eta)}[g(\theta)] = m$, Assumption 2.4 is satisfied. Finally, by Lemma 2.1, $E_{q_\alpha}[\theta] = E_{p_\alpha}[\theta]$, so Condition 2.1 is satisfied with equality.

It now follows immediately from Definition 2.6 that the linear response variational covariance exactly reproduces the exact posterior covariance for the multivariate normal distribution.

**Corollary 2.4.** Under the conditions of Lemma 2.2, $\text{Cov}_{q_0}^{LR}(\theta) = \text{Cov}_{p_\alpha}(\theta)$. 
Chapter 3

Linear response covariances for exponential families

With increasingly efficient data collection methods, scientists are interested in quickly analyzing ever larger data sets. In particular, the promise of these large data sets is not simply to fit old models but instead to learn more nuanced patterns from data than has been possible in the past. In theory, the Bayesian paradigm yields exactly these desiderata. Hierarchical modeling allows practitioners to capture complex relationships between variables of interest. Moreover, Bayesian analysis allows practitioners to quantify the uncertainty in any model estimates—and to do so coherently across all of the model variables.

Mean field variational Bayes (MFVB), a method for approximating a Bayesian posterior distribution, has grown in popularity due to its fast runtime on large-scale data sets [Blei et al., 2003, Blei and Jordan, 2006, Hoffman et al., 2013]. But a well-known major failing of MFVB is that it gives underestimates of the uncertainty of model variables that can be arbitrarily bad, even when approximating a simple multivariate Gaussian distribution [MacKay, 2003, Bishop, 2006, Turner and Sahani, 2011]. Also, MFVB provides no information about how the uncertainties in different model variables interact [Wang and Titterington, 2004, Bishop, 2006, Rue et al., 2009, Turner and Sahani, 2011].

By generalizing linear response methods from statistical physics [Parisi, 1988, Opper and Winther, 2003, Opper and Saad, 2001, Tanaka, 2000] to exponential family variational posteriors, we develop a methodology that augments MFVB to deliver accurate uncertainty estimates for model variables—both for individual variables and coherently across variables. In particular, as we elaborate in Section 3.1, when the approximating posterior in MFVB is in the exponential family, MFVB defines a fixed-point equation in the means of the approximating posterior, and our approach yields a covariance estimate by perturbing this fixed point. We call our
method *linear response variational Bayes* (LRVB).

We provide a simple, intuitive formula for calculating the linear response correction by solving a linear system based on the MFVB solution (Section 3.1). We show how the sparsity of this system for many common statistical models may be exploited for scalable computation (Section 3.2). We demonstrate the wide applicability of LRVB by working through a diverse set of models to show that the LRVB covariance estimates are nearly identical to those produced by a Markov Chain Monte Carlo (MCMC) sampler, even when MFVB variance is dramatically underestimated (Section 3.3). Finally, we focus in more depth on models for finite mixtures of multivariate Gaussians (Section 3.3), which have historically been a sticking point for MFVB covariance estimates [Bishop, 2006, Turner and Sahani, 2011]. We show that LRVB can give accurate covariance estimates orders of magnitude faster than MCMC (Section 3.3). We demonstrate both theoretically and empirically that, for this Gaussian mixture model, LRVB scales linearly in the number of data points and approximately cubically in the dimension of the parameter space (Section 3.3).

**Previous Work.** Linear response methods originated in the statistical physics literature [Opper and Saad, 2001, Tanaka, 2000, Kappen and Rodriguez, 1998, Opper and Winther, 2003]. These methods have been applied to find new learning algorithms for Boltzmann machines [Kappen and Rodriguez, 1998], covariance estimates for discrete factor graphs [Welling and Teh, 2004], and independent component analysis [Højen-Sørensen et al., 2002]. Tanaka [1998] states that linear response methods could be applied to general exponential family models but works out details only for Boltzmann machines. Opper and Winther [2003], which is closest in spirit to the present work, derives general linear response corrections to variational approximations; indeed, the authors go further to formulate linear response as the first term in a functional Taylor expansion to calculate full pairwise joint marginals. However, it may not be obvious to the practitioner how to apply the general formulas of Opper and Winther [2003]. Our contributions in the present work are (1) the provision of concrete, straightforward formulas for covariance correction that are fast and easy to compute, (2) demonstrations of the success of our method on a wide range of new models, and (3) an accompanying suite of code.
3.1 Linear response covariance estimation

Variational inference

Suppose we observe $N$ data points, denoted by the $N$-long column vector $x$, and denote our unobserved model parameters by $\theta$. Here, $\theta$ is a column vector residing in some space $\Theta$; it has $J$ subgroups and total dimension $D$. Our model is specified by a distribution of the observed data given the model parameters—the likelihood $p(x|\theta)$—and a prior distributional belief on the model parameters $p(\theta)$. Bayes’ Theorem yields the posterior $p(\theta|x)$.

Mean-field variational Bayes (MFVB) approximates $p(\theta|x)$ by a factorized distribution of the form $q(\theta) = \prod_{j=1}^J q(\theta_j)$. $q$ is chosen so that the Kullback-Liebler divergence $\text{KL}(q||p)$ between $q$ and $p$ is minimized. Equivalently, $q$ is chosen so that $E := L + S$, for $L := \mathbb{E}_q[\log p(\theta|x)]$ (the expected log posterior) and $S := -\mathbb{E}_q[\log q(\theta)]$ (the entropy of the variational distribution), is maximized:

$$q^* := \arg\min_q \text{KL}(q||p) = \arg\min_q \mathbb{E}_q [\log q(\theta) - \log p(\theta|x)] = \arg\max_q E. \quad (3.1)$$

Up to a constant in $\theta$, the objective $E$ is sometimes called the “evidence lower bound,” or the ELBO [Bishop, 2006]. In what follows, we further assume that our variational distribution, $q(\theta)$, is in the exponential family with natural parameter $\eta$ and log partition function $A$: $\log q(\theta|\eta) = \eta^T \theta - A(\eta)$ (expressed with respect to some base measure in $\theta$). We assume that $p(\theta|x)$ is expressed with respect to the same base measure in $\theta$ as for $q$. Below, we will make only mild regularity assumptions about the true posterior $p(\theta|x)$ and no assumptions about its form.

If we assume additionally that the parameters $\eta^*$ at the optimum $q^*(\theta) = q(\theta|\eta^*)$ are in the interior of the feasible space, then $q(\theta|\eta)$ may instead be described by the mean parameterization: $m := \mathbb{E}_q \theta$ with $m^* := \mathbb{E}_{q^*} \theta$. Thus, the objective $E$ can be expressed as a function of $m$, and the first-order condition for the optimality of $q^*$ becomes the fixed point equation

$$\frac{\partial E}{\partial m} \bigg|_{m=m^*} = 0 \iff $$

$$\left(\frac{\partial E}{\partial m} + m\right) \bigg|_{m=m^*} = m^* \iff $$

$$M(m^*) = m^* \text{ for } M(m) := \frac{\partial E}{\partial m} + m. \quad (3.2)$$
Linear response

Let $V$ denote the covariance matrix of $\theta$ under the variational distribution $q^*(\theta)$, and let $\Sigma$ denote the covariance matrix of $\theta$ under the true posterior, $p(\theta|x)$:

$$V := \text{Cov}_{q^*}\theta, \quad \Sigma := \text{Cov}_p\theta.$$ 

In MFVB, $V$ may be a poor estimator of $\Sigma$, even when $m^* \approx \mathbb{E}_p\theta$, i.e., when the marginal estimated means match well \[\text{[Wang and Titterington 2004, Bishop 2006, Turner and Sahani 2011]}\]. Our goal is to use the MFVB solution and linear response methods to construct an improved estimator for $\Sigma$. We will focus on the covariance of the natural sufficient statistic $\theta$, though the covariance of functions of $\theta$ can be estimated similarly (see Section 3.1).

The essential idea of linear response is to perturb the first-order condition $M(m^*) = m^*$ around its optimum. In particular, define the distribution $p_t(\theta|x)$ as a log-linear perturbation of the posterior:

$$\log p_t(\theta|x) := \log p(\theta|x) + t^T \theta - \text{Constant}(t), \quad (3.3)$$

where $\text{Constant}(t)$ is a constant in $\theta$. We assume that $p_t(\theta|x)$ is a well-defined distribution for any $t$ in an open ball around 0. Since $\text{Constant}(t)$ normalizes $p_t(\theta|x)$, it is in fact the cumulant-generating function of $p(\theta|x)$, so the derivatives of $\text{Constant}(t)$ evaluated at $t = 0$ give the cumulants of $\theta$. To see why this perturbation may be useful, recall that the second cumulant of a distribution is the covariance matrix, our desired estimand:

$$\Sigma = \text{Cov}_p(\theta) = \frac{d}{dt} \mathbb{E}_p\theta \bigg|_{t=0}.$$

The practical success of MFVB relies on the fact that its estimates of the mean are often good in practice. So we assume that $m_t^* \approx \mathbb{E}_p\theta$, where $m_t^*$ is the mean parameter characterizing $q_t^*$ and $q_t^*$ is the MFVB approximation to $p_t$. (We examine this assumption further in Section 3.3.) Taking derivatives with respect to $t$ on both sides of this mean approximation and setting $t = 0$ yields

$$\Sigma = \text{Cov}_p(\theta) \approx \frac{dm_t^*}{dt} \bigg|_{t=0} =: \hat{\Sigma}, \quad (3.4)$$

where we call $\hat{\Sigma}$ the linear response variational Bayes (LRVB) estimate of the posterior covariance of $\theta$.

We next show that there exists a simple formula for $\hat{\Sigma}$. Recalling the form of the KL divergence (see Eq. (3.1)), we have that $-\text{KL}(q||p_t) = E_t + t^T m =: E_t$. Then
by Eq. (3.2), we have \( m_t^* = M_t(m_t^*) \) for \( M_t(m) := M(m) + t \). It follows from the chain rule that
\[
\frac{dm_t^*}{dt} = \left. \frac{\partial M_t}{\partial m} \right|_{m=m_t^*} \frac{dm_t^*}{dt} + \frac{\partial M_t}{\partial t}
\]
\[
= \left. \frac{\partial M_t}{\partial m} \right|_{m=m_t^*} \frac{dm_t^*}{dt} + I,
\]
(3.5)
where \( I \) is the identity matrix. If we assume that we are at a strict local optimum and so can invert the Hessian of \( E \), then evaluating at \( t = 0 \) yields
\[
\hat{\Sigma} = \left. \frac{dm_t^*}{dt} \right|_{t=0}
\]
\[
= \frac{\partial M}{\partial m} \hat{\Sigma} + I
\]
\[
= \left( \frac{\partial^2 E}{\partial m \partial m^T} + I \right) \hat{\Sigma} + I \Rightarrow
\]
\[
\hat{\Sigma} = -\left( \frac{\partial^2 E}{\partial m \partial m^T} \right)^{-1},
\]
(3.6)
where we have used the form for \( M \) in Eq. (3.2). So the LRVB estimator \( \hat{\Sigma} \) is the negative inverse Hessian of the optimization objective, \( E \), as a function of the mean parameters. It follows from Eq. (3.6) that \( \hat{\Sigma} \) is both symmetric and positive definite when the variational distribution \( q^* \) is at least a local maximum of \( E \).

We can further simplify Eq. (3.6) by using the exponential family form of the variational approximating distribution \( q \). For \( q \) in exponential family form as above, the negative entropy \( -S \) is dual to the log partition function \( A \) \cite{Wainwright2008}, so \( S = -\eta^T m + A(\eta) \); hence,
\[
\frac{dS}{dm} = \frac{\partial S}{\partial \eta^T} \frac{d\eta}{dm} + \frac{\partial S}{\partial \eta} \frac{d\eta}{dm} = \left( \frac{\partial A}{\partial \eta} - m \right) \frac{d\eta}{dm} - \eta(m) = -\eta(m).
\]
Recall that for exponential families, \( \partial \eta(m)/\partial m = V^{-1} \). So Eq. (3.6) becomes\(^1\)
\[
\hat{\Sigma} = -\left( \frac{\partial^2 L}{\partial m \partial m^T} + \frac{\partial^2 S}{\partial m \partial m^T} \right)^{-1}
\]
\[
= -(H - V^{-1})^{-1}, \text{ for } H := \frac{\partial^2 L}{\partial m \partial m^T}, \Rightarrow
\]
\[
\hat{\Sigma} = (I - VH)^{-1}V.
\]
(3.7)
\(^1\)For a comparison of this formula with the frequentist “supplemented expectation-maximization” procedure see Section 3.2.
When the true posterior $p(\theta|x)$ is in the exponential family and contains no products of the variational moment parameters, then $H = 0$ and $\Sigma = V$. In this case, the mean field assumption is correct, and the LRVB and MFVB covariances coincide at the true posterior covariance. Furthermore, even when the variational assumptions fail, as long as certain mean parameters are estimated exactly, then this formula is also exact for covariances. E.g., notably, MFVB is well-known to provide arbitrarily bad estimates of the covariance of a multivariate normal posterior [MacKay, 2003, Wang and Titterington, 2004, Bishop, 2006, Turner and Sahani, 2011], but since MFVB estimates the means exactly, LRVB estimates the covariance exactly (see Section 3.2).

**LRVB estimates of the covariance of functions**

In Section 3.1, we derived an estimate of the covariance of the natural sufficient statistics, $\theta$, of our variational approximation, $q(\theta)$. In this section we derive a version of Eq. (3.7) for the covariance of functions of $\theta$.

We begin by estimating the covariance between $\theta$ and a function $\phi(\theta)$. Suppose we have an MFVB solution, $q(\theta)$, to Eq. (3.1). Define the expectation of $\phi(\theta)$ to be $\mathbb{E}_{q, \alpha} [\phi(\theta)] := f(m)$. This expectation is function of $m$ alone since $m$ completely parameterizes $q$. As in Eq. (3.3), we can consider a perturbed log likelihood that also includes $f(m)$:

$$\log p_t (\theta|x) = \log p + t_0^T m + t_f f(m) := \log p + t^T m_f$$

Using the same reasoning that led to Eq. (3.4), we will define

$$\Sigma_{\theta\phi} = \text{Cov}_p(\theta, \phi(\theta)) \approx \frac{d m^*_t}{d t_f} =: \hat{\Sigma}_{\theta\phi}.$$

We then have the following lemma:

**Lemma 3.1.** If $\mathbb{E}_{q, \alpha} [\phi(\theta)] := f(m)$ is a differentiable function of $m$ with gradient $\nabla f$, then

$$\hat{\Sigma}_{\theta\phi} = \hat{\Sigma} \nabla f.$$

**Proof.** The derivative of the perturbed ELBO, $E_t$, is given by:

$$\frac{d E_t}{d m} = \frac{\partial E_t}{\partial m} = \frac{\partial E}{\partial m} + (I \nabla f) \left( \begin{array}{c} t_0 \\ t_f \end{array} \right)$$

where

$$E_t := E + t^T m_f.$$
The fixed point Eq. (3.2) then gives:

\[
M_t(m) := M(m) + \left( I \nabla f \right) \left( \begin{array}{c} t_0 \\ t_f \end{array} \right)
\]

\[
\frac{dm^*}{dt^T} = \left( \frac{\partial M}{\partial m^T} \right)_{m=m^*} \frac{dm^*}{dt^T} + \frac{\partial M_t}{\partial t^T} \bigg|_{m=m^*} + \frac{\partial M_t}{\partial t^T} \bigg|_{m=m^*} \frac{dm^*}{dt^T} + \left( I \nabla f \right) \left( \begin{array}{c} t_0 \\ t_f \end{array} \right) \frac{dm^*}{dt^T} + \left( I \nabla f \right).
\]

The term \( \frac{\partial}{\partial m^T} \left( I \nabla f \right) \left( \begin{array}{c} t_0 \\ t_f \end{array} \right) \) is awkward, but it disappears when we evaluate at \( t = 0 \), giving

\[
\frac{dm^*}{dt^T} = \left( \frac{\partial M}{\partial m^T} \right)_{m=m^*} \frac{dm^*}{dt^T} + \left( I \nabla f \right)
\]

\[
= \left( \frac{\partial^2 E}{\partial m \partial m^T} + I \right) \frac{dm^*}{dt^T} + \left( I \nabla f \right) \Rightarrow
\]

\[
\frac{dm^*}{dt^T} = -\left( \frac{\partial^2 E}{\partial m \partial m^T} \right)^{-1} \left( I \nabla f \right).
\]

Recalling that

\[
\frac{dm^*}{dt^T \bigg|_{t=0}} := \hat{\Sigma}.
\]

we can plug in to see that

\[
\hat{\Sigma}_{\gamma \phi} = \frac{dm^*}{dt_f} = \hat{\Sigma} \nabla f.
\]

(3.8)

Finally, suppose we are interested in estimating \( \text{Cov}_p(\gamma(\theta), \phi(\theta)) \), where \( g(m) := \mathbb{E}_{q_\alpha}[\gamma(\theta)] \). Again using the same reasoning that led to Eq. (3.4), we will define

\[
\Sigma_{\gamma \phi} = \text{Cov}_p(\gamma(\theta), \phi(\theta)) \approx \frac{d\mathbb{E}_{q_\alpha}[\gamma(\theta)]}{dt_f} =: \hat{\Sigma}_{\gamma \phi}.
\]

**Proposition 3.1.** *If \( \mathbb{E}_{q_\alpha} [\phi(\theta)] = f(m) \) and \( \mathbb{E}_{q_\alpha} [\gamma(\theta)] = g(m) \) are differentiable functions of \( m \) with gradients \( \nabla f \) and \( \nabla g \) respectively, then

\[
\hat{\Sigma}_{\gamma \phi} = \nabla g^T \hat{\Sigma} \nabla f.
\]

\( \square \)
CHAPTER 3. EXPONENTIAL FAMILIES

Proof. By Lemma 3.1 an application of the chain rule,

\[ \hat{\Sigma}_{\gamma \phi} = \frac{d \mathbb{E}_{q_0} [\gamma(\theta)]}{dt_f} = \frac{dg(m)}{dm} \frac{dm}{dt_f} = \nabla g^T \hat{\Sigma} \nabla f. \]

\[ \square \]

3.2 Discussion and comparison with alternative methods

Exactness of LRVB for multivariate Normal means

For any target distribution \( p(\theta | x) \), it is well-known that MFVB cannot be used to estimate the covariances between the components of \( \theta \). In particular, if \( q^* \) is the estimate of \( p(\theta | x) \) returned by MFVB, \( q^* \) will have a block-diagonal covariance matrix—no matter the form of the covariance of \( p(\theta | x) \).

Consider approximating a multivariate Gaussian posterior distribution \( p(\theta | x) \) with MFVB. The Gaussian is the unique distribution that is fully determined by its mean and covariance. This posterior arises, for instance, given a multivariate normal likelihood \( p(x | \mu) = \prod_{n=1:N} \mathcal{N}(x_n | \mu, S) \) with fixed covariance \( S \) and an improper uniform prior on the mean parameter \( \mu \). We make the mean field factorization assumption \( q(\mu) = \prod_{d=1:D} q(\mu_d) \), where \( D \) is the total dimension of \( \mu \). This fact is often used to illustrate the shortcomings of MFVB [Wang and Titterington, 2004; Bishop, 2006; Turner and Sahani, 2011]. In this case, it is well known that the MFVB posterior means are correct, but the marginal variances are underestimated if \( S \) is not diagonal. However, since the posterior means are correctly estimated, the LRVB approximation in Eq. (3.7) is in fact an equality. That is, for this model, \( \hat{\Sigma} = dm_t/dt_f = \Sigma \) exactly.

In order to prove this result, we will rely on the following lemma.

Lemma 3.2. Consider a target posterior distribution characterized by \( p(\theta | x) = \mathcal{N}(\theta | \mu, \Sigma) \), where \( \mu \) and \( \Sigma \) may depend on \( x \), and \( \Sigma \) is invertible. Let \( \theta = (\theta_1, \ldots, \theta_J) \), and consider a MFVB approximation to \( p(\theta | x) \) that factorizes as \( q(\theta) = \prod_j q(\theta_j) \). Then the variational posterior means are the true posterior means; i.e. \( m_j = \mu_j \) for all \( j \) between 1 and \( J \).

Proof. The derivation of MFVB for the multivariate normal can be found in Section 10.1.2 of Bishop [2006]; we highlight some key results here. Let \( \Lambda = \Sigma^{-1} \). Let the \( j \) index on a row or column correspond to \( \theta_j \), and let the \(-j\) index correspond to
\{\theta_i : i \in [J] \setminus j\}. E.g., for \( j = 1 \),

\[ \Lambda = \begin{bmatrix} \Lambda_{11} & \Lambda_{1,1} \\ \Lambda_{-1,1} & \Lambda_{-1,-1} \end{bmatrix}. \]

By the assumption that \( p(\theta|x) = \mathcal{N}(\theta|\mu, \Sigma) \), we have

\[
\log p(\theta_j|\theta_{i \in [J]\setminus j}, x) = -\frac{1}{2}(\theta_j - \mu_j)^T \Lambda_{jj}(\theta_j - \mu_j) + (\theta_j - \mu_j)^T \Lambda_{j,-j}(\theta_{-j} - \mu_{-j}) + \text{const}(3.9)
\]

where the final term is constant with respect to \( \theta_j \). It follows that

\[
\log q^*_j(\theta_j) = \mathbb{E}_{q^*_{i \in [J]\setminus j}} \log p(\theta, x) + \text{const} = -\frac{1}{2} \theta_j^T \Lambda_{jj} \theta_j + \theta_j \mu_j \Lambda_{jj} - \theta_j \Lambda_{j,-j}(\mathbb{E}_{q^*}\theta_{-j} - \mu_{-j}).
\]

So

\[ q^*_j(\theta_j) = \mathcal{N}(\theta_j|m_j, \Lambda_{jj}^{-1}), \]

with mean parameters

\[ m_j = \mathbb{E}_{q^*_j} \theta_j = \mu_j - \Lambda_{jj}^{-1} \Lambda_{j,-j} (m_{-j} - \mu_{-j}) \tag{3.10} \]

as well as an equation for \( \mathbb{E}_{q^*} \theta_j^T \theta_j \).

Note that \( \Lambda_{jj} \) must be invertible, for if it were not, \( \Sigma \) would not be invertible.

The solution \( m = \mu \) is a unique stable point for Eq. (3.10), since the fixed point
equations for each $j$ can be stacked and rearranged to give

$$m - \mu = - \begin{bmatrix} 0 & \Lambda_{11}^{-1} \Lambda_{12} & \cdots & \Lambda_{11}^{-1} \Lambda_{1(J-1)} & \Lambda_{11}^{-1} \Lambda_{1J} \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
\Lambda_{J1}^{-1} \Lambda_{J1} & \Lambda_{J1}^{-1} \Lambda_{J1J} & \cdots & \Lambda_{J1}^{-1} \Lambda_{J1(J-1)} & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & 0 & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & \Lambda_{J1}^{-1} \Lambda_{J1J} & 0 \\
0 & \cdots & \cdots & \Lambda_{J1J} & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & \Lambda_{J1}^{-1} \Lambda_{J1J} & 0 \\
\Lambda_{J1}^{-1} \Lambda_{J1J} & \Lambda_{J1J}^{-1} \Lambda_{J1(J-1)} & \cdots & \Lambda_{J1J}^{-1} \Lambda_{J1J} & 0 \end{bmatrix} (m - \mu)$$

$$= - \begin{bmatrix} 0 & \Lambda_{12} & \cdots & \Lambda_{1(J-1)} & \Lambda_{1J} \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
\Lambda_{J1} & \Lambda_{J2} & \cdots & \Lambda_{J(J-1)} & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & 0 & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & \Lambda_{JJ} & 0 \\
0 & \cdots & \cdots & \Lambda_{J1} \Lambda_{J2} & \cdots & \Lambda_{J(J-1)} \Lambda_{JJ} & 0 \end{bmatrix} \times \begin{bmatrix} 0 & \Lambda_{12} & \cdots & \Lambda_{1(J-1)} & \Lambda_{1J} \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
\Lambda_{J1} & \Lambda_{J2} & \cdots & \Lambda_{J(J-1)} & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & 0 & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & \Lambda_{JJ} & 0 \\
0 & \cdots & \cdots & \Lambda_{J1} \Lambda_{J2} & \cdots & \Lambda_{J(J-1)} \Lambda_{JJ} & 0 \end{bmatrix} (m - \mu)$$

$$0 = \Lambda (m - \mu) \Leftrightarrow \begin{bmatrix} 0 & \Lambda_{12} & \cdots & \Lambda_{1(J-1)} & \Lambda_{1J} \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
\Lambda_{J1} & \Lambda_{J2} & \cdots & \Lambda_{J(J-1)} & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & 0 & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & \Lambda_{JJ} & 0 \\
0 & \cdots & \cdots & \Lambda_{J1} \Lambda_{J2} & \cdots & \Lambda_{J(J-1)} \Lambda_{JJ} & 0 \end{bmatrix} (m - \mu) \Leftrightarrow \begin{bmatrix} 0 & \Lambda_{12} & \cdots & \Lambda_{1(J-1)} & \Lambda_{1J} \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
\Lambda_{J1} & \Lambda_{J2} & \cdots & \Lambda_{J(J-1)} & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & 0 & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & \Lambda_{JJ} & 0 \\
0 & \cdots & \cdots & \Lambda_{J1} \Lambda_{J2} & \cdots & \Lambda_{J(J-1)} \Lambda_{JJ} & 0 \end{bmatrix} (m - \mu) \Leftrightarrow $$

$$m = \Lambda (m - \mu) \Leftrightarrow m = \mu.$$

The last step follows from the assumption that $\Sigma$ (and hence $\Lambda$) is invertible. It follows that $\mu$ is the unique stable point of Eq. (3.10).

**Proposition 3.2.** Assume we are in the setting of Lemma 3.2, where additionally $\mu$ and $\Sigma$ are on the interior of the feasible parameter space. Then the LRVB covariance estimate exactly captures the true covariance, $\hat{\Sigma} = \Sigma$.

**Proof.** Consider the perturbation for LRVB defined in Eq. (3.3). By perturbing the log likelihood, we change both the true means $\mu_t$ and the variational solutions, $m_t$. The result is a valid density function since the original $\mu$ and $\Sigma$ are on the interior of the parameter space. By Lemma 3.2, the MFVB solutions are exactly the true
means, so \( m_{t,j} = \mu_{t,j} \), and the derivatives are the same as well. This means that the first term in Eq. (3.7) is not approximate, i.e.

\[
\frac{dm_t}{dT} = \frac{d}{dT} E_p \theta = \Sigma_t,
\]

It follows from the arguments above that the LRVB covariance matrix is exact, and \( \hat{\Sigma} = \Sigma \).

**Comparison with supplemented expectation-maximization**

The result in Section 3.2 about the multivariate normal distribution draws a connection between LRVB corrections and the “supplemented expectation-maximization” (SEM) method of Meng and Rubin [1991]. SEM is an asymptotically exact covariance correction for the EM algorithm that transforms the full-data Fisher information matrix into the observed-data Fisher information matrix using a correction that is formally similar to Eq. (3.7). In this section, we argue that this similarity is not a coincidence; in fact the SEM correction is an asymptotic version of LRVB with two variational blocks, one for the missing data and one for the unknown parameters.

Although LRVB as described here requires a prior (unlike SEM, which supplements the MLE), the two covariance corrections coincide when the full information likelihood is approximately log quadratic and proportional to the posterior, \( p(\theta|x) \). This might be expected to occur when we have a large number of independent data points informing each parameter—i.e., when a central limit theorem applies and the priors do not affect the posterior. In the full information likelihood, some terms may be viewed as missing data, whereas in the Bayesian model the same terms may be viewed as latent parameters, but this does not prevent us from formally comparing the two methods.

We can draw a term-by-term analogy with the equations in Meng and Rubin [1991]. We denote variables from the SEM paper with a superscript “SEM” to avoid confusion. MFVB does not differentiate between missing data and parameters to be estimated, so our \( \theta \) corresponds to \((\theta^{SEM}, Y^{SEM}_{mis})\) in Meng and Rubin [1991]. SEM is an asymptotic theory, so we may assume that \((\theta^{SEM}, Y^{SEM}_{mis})\) have a multivariate normal distribution, and that we are interested in the mean and covariance of \( \theta^{SEM} \).

In the E-step of Meng and Rubin [1991], we replace \( Y^{SEM}_{mis} \) with its conditional expectation given the data and other \( \theta^{SEM} \). This corresponds precisely to Eq. (3.10), taking \( \theta_j = Y^{SEM}_{mis} \). In the M-step, we find the maximum of the log likelihood with respect to \( \theta^{SEM} \), keeping \( Y^{SEM}_{mis} \) fixed at its expectation. Since the mode of a
multivariate normal distribution is also its mean, this, too, corresponds to Eq. (3.10), now taking \( \theta_j = \theta_{SEM} \).

It follows that the MFVB and EM fixed point equations are the same; i.e., our \( M \) is the same as their \( M_{SEM} \), and our \( \partial M / \partial m \) of Eq. (3.5) corresponds to the transpose of their \( DM_{SEM} \), defined in Eq. (2.2.1) of Meng and Rubin [1991]. Since the “complete information” corresponds to the variance of \( \theta_{SEM} \) with fixed values for \( Y_{OBS} \), this is the same as our \( \Sigma_{q*11} \), the variational covariance, whose inverse is \( I_{oc}^{-1} \). Taken all together, this means that equation (2.4.6) of Meng and Rubin [1991] can be re-written as our Eq. (3.7).

\[
V_{SEM} = I_{oc}^{-1} (I - DM_{SEM})^{-1} \implies \\
\Sigma = V \left( I - \left( \frac{\partial M}{\partial m^T} \right)^T \right)^{-1} = \left( I - \frac{\partial M}{\partial m^T} \right)^{-1} V.
\]

### Scaling the matrix inverse

Eq. (3.7) requires the inverse of a matrix as large as the parameter dimension of the posterior \( p(\theta|x) \), which may be computationally prohibitive. Suppose we are interested in the covariance of parameter sub-vector \( \alpha \), and let \( z \) denote the remaining parameters: \( \theta = (\alpha, z)^T \). We can partition \( \Sigma = (\Sigma_\alpha, \Sigma_{\alpha z}; \Sigma_{za}, \Sigma_z) \). Similar partitions exist for \( V \) and \( H \). If we assume a mean-field factorization \( q(\alpha, z) = q(\alpha)q(z) \), then \( V_{az} = 0 \). (The variational distributions may factor further as well.) We calculate the Schur complement of \( \Sigma \) in Eq. (3.7) with respect to its \( z \)th component to find that

\[
\hat{\Sigma}_\alpha = (I_\alpha - V_\alpha H_\alpha - V_\alpha H_{az} (I_z - V_z H_z)^{-1} V_z H_{za})^{-1} V_\alpha.
\]  

(3.11)

Here, \( I_\alpha \) and \( I_z \) refer to \( \alpha \)- and \( z \)-sized identity matrices, respectively. In cases where \( (I_z - V_z H_z)^{-1} \) can be efficiently calculated (e.g., all the experiments in Section 3.3; see Fig. (A.1) in Appendix A.1), Eq. (3.11) requires only an \( \alpha \)-sized inverse.

### 3.3 Experiments

We compare the covariance estimates from LRVB and MFVB in a range of models, including models both with and without conjugacy. We demonstrate the superiority of the LRVB estimate to MFVB in all models before focusing in on Gaussian mixture models for a more detailed scalability analysis.

For each model, we simulate datasets with a range of parameters. In the graphs, each point represents the outcome from a single simulation. The horizontal axis is
always the result from an MCMC procedure, which we take as the ground truth. As discussed in Section 3.1, the accuracy of the LRVB covariance for a sufficient statistic depends on the approximation \( m_t^* \approx \mathbb{E}_p \theta \). In the models to follow, we focus on regimes of moderate dependence where this is a reasonable assumption for most of the parameters (see Section 3.3 for an exception). Except where explicitly mentioned, the MFVB means of the parameters of interest coincided well with the MCMC means, so our key assumption in the LRVB derivations of Section 3.1 appears to hold.

**Normal-Poisson Model**

**Model.** First consider a Poisson generalized linear mixed model, exhibiting non-conjugacy. We observe Poisson draws \( y_n \) and a design vector \( x_n \), for \( n = 1, \ldots, N \). Implicitly below, we will everywhere condition on the \( x_n \), which we consider to be a fixed design matrix. The generative model is:

\[
\begin{align*}
    z_n | \beta, \tau &\sim \mathcal{N} (z_n | \beta x_n, \tau^{-1}) , \quad y_n | z_n \sim \text{Poisson} (y_n | \exp(z_n)) , \\
    \beta &\sim \mathcal{N} (\beta | 0, \sigma^2_\beta) , \quad \tau \sim \Gamma (\tau | \alpha_\tau, \beta_\tau).
\end{align*}
\] (3.12)

For MFVB, we factorize \( q (\beta, \tau, z) = q (\beta) q (\tau) \prod_{n=1}^N q (z_n) \). Inspection reveals that the optimal \( q (\beta) \) will be Gaussian, and the optimal \( q (\tau) \) will be gamma (see Appendix A.1). Since the optimal \( q (z_n) \) does not take a standard exponential family form, we restrict further to Gaussian \( q (z_n) \). There are product terms in \( L \) (for example, the term \( \mathbb{E}_{q_\alpha} [\tau] \mathbb{E}_{q_\alpha} [\beta] \mathbb{E}_{q_\alpha} [z_n] \)), so \( H \neq 0 \), and the mean field approximation does not hold; we expect LRVB to improve on the MFVB covariance estimate. A detailed description of how to calculate the LRVB estimate can be found in Appendix A.1.

**Results.** We simulated 100 datasets, each with 500 data points and a randomly chosen value for \( \mu \) and \( \tau \). We drew the design matrix \( x \) from a normal distribution and held it fixed throughout. We set prior hyperparameters \( \sigma^2_\beta = 10, \alpha_\tau = 1, \) and \( \beta_\tau = 1 \). To get the “ground truth” covariance matrix, we took 20000 draws from the posterior with the R \texttt{MCMCglmm} package [Hadfield, 2010], which used a combination of Gibbs and Metropolis Hastings sampling. Our LRVB estimates used the autodifferentiation software \texttt{JuMP} [Lubin and Dunning, 2015].

Results are shown in Fig. (3.1). Since \( \tau \) is high in many of the simulations, \( z \) and \( \beta \) are correlated, and MFVB underestimates the standard deviation of \( \beta \) and \( \tau \). LRVB matches the MCMC standard deviation for all \( \beta \), and matches for \( \tau \) in all but the most correlated simulations. When \( \tau \) gets very high, the MFVB assumption starts to bias the point estimates of \( \tau \), and the LRVB standard deviations start to
differ from MCMC. Even in that case, however, the LRVB standard deviations are much more accurate than the MFVB estimates, which underestimate the uncertainty dramatically. The final plot shows that LRVB estimates the covariances of $z$ with $\beta$, $\tau$, and $\log \tau$ reasonably well, while MFVB considers them independent.

Figure 3.1: Posterior mean and covariance estimates on normal-Poisson simulation data.

**Linear random effects**

**Model.** Next, we consider a simple random slope linear model, with full details in Appendix [A.2]. We observe scalars $y_n$ and $r_n$ and a vector $x_n$, for $n = 1, \ldots, N$. Implicitly below, we will everywhere condition on all the $x_n$ and $r_n$, which we consider to be fixed design matrices. In general, each random effect may appear in multiple observations, and the index $k(n)$ indicates which random effect, $z_k$, affects which observation, $y_n$. The full generative model is:

$$y_n | \beta, z, \tau \overset{\text{iid}}{\sim} \mathcal{N}(y_n | \beta^T x_n + r_n z_k(n), \tau^{-1}) \quad z_k | \nu \overset{\text{iid}}{\sim} \mathcal{N}(z_k | 0, \nu^{-1})$$

$$\beta \sim \mathcal{N}(\beta | 0, \Sigma_\beta) \quad \nu \sim \Gamma(\nu | \alpha_\nu, \beta_\nu) \quad \tau \sim \Gamma(\tau | \alpha_\tau, \beta_\tau)$$

We assume the mean-field factorization $q(\beta, \nu, z) = q(\beta)q(\nu)q(z)\prod_{k=1}^{K}q(z_k)$. Since this is a conjugate model, the optimal $q$ will be in the exponential family with no additional assumptions.

**Results.** We simulated 100 datasets of 300 datapoints each and 30 distinct random effects. We set prior hyperparameters to $\alpha_\nu = 2$, $\beta_\nu = 2$, $\alpha_\tau = 2$, $\beta_\tau = 2$, and $\Sigma_\beta = 0.1^{-1}I$. Our $x_n$ was 2-dimensional. As in Section 3.3, we implemented the variational solution using the autodifferentiation software JuMP [Lubin and Dunning, 2015]. The MCMC fit was performed with using MCMCglmm [Hadfield, 2010].

Intuitively, when the random effect explanatory variables $r_n$ are highly correlated with the fixed effects $x_n$, then the posteriors for $z$ and $\beta$ will also be correlated, leading to a violation of the mean field assumption and an underestimated MFVB covariance. In our simulation, we used $r_n = x_{1n} + \mathcal{N}(0, 0.4)$, so that $r_n$ is correlated...
with \(x_{1n}\) but not \(x_{2n}\). The result, as seen in Fig. (3.2), is that \(\beta_1\) is underestimated by MFVB, but \(\beta_2\) is not. The \(\nu\) parameter, in contrast, is not well-estimated by the MFVB approximation in many of the simulations. Since the LRVB depends on the approximation \(m^*_t \approx \mathbb{E}_p \theta\), its LRVB covariance is not accurate either (Fig. (3.2)). However, LRVB still improves on the MFVB standard deviation.

![Figure 3.2: Posterior mean and covariance estimates on linear random effects simulation data.](image)

### Mixture of Normals

**Model.** Mixture models constitute some of the most popular models for MFVB application \([\text{Blei et al.}, 2003, \text{Blei and Jordan}, 2006]\) and are often used as an example of where MFVB covariance estimates may go awry \([\text{Bishop}, 2006, \text{Turner and Sahani}, 2011]\). Thus, we will consider in detail a Gaussian mixture model (GMM) consisting of a \(K\)-component mixture of \(P\)-dimensional multivariate normals with unknown component means, covariances, and weights. In what follows, the weight \(\pi_k\) is the probability of the \(k\)th component, \(\mu_k\) is the \(P\)-dimensional mean of the \(k\)th component, and \(\Lambda_k\) is the \(P \times P\) precision matrix of the \(k\)th component (so \(\Lambda_k^{-1}\) is the covariance parameter). \(N\) is the number of data points, and \(x_n\) is the \(n\)th observed \(P\)-dimensional data point. We employ the standard trick of augmenting the data generating process with the latent indicator variables \(z_{nk}\), for \(n = 1, \ldots, N\) and \(k = 1, \ldots, K\), such that \(z_{nk} = 1\) implies \(x_n \sim \mathcal{N}(\mu_k, \Lambda_k^{-1})\). So the generative model is:

\[
P(z_{nk} = 1) = \pi_k, \quad p(x | \pi, \mu, \Lambda, z) = \prod_{n=1:N} \prod_{k=1:K} \mathcal{N}(x_n | \mu_k, \Lambda_k^{-1})^{z_{nk}}.
\]

We used diffuse conditionally conjugate priors (see Appendix A.3 for details). We make the variational assumption \(q(\mu, \pi, \Lambda, z) = \prod_{k=1}^K q(\mu_k) q(\Lambda_k) q(\pi_k) \prod_{n=1}^N q(z_n)\). We compare the accuracy and speed of our estimates to Gibbs sampling on the augmented model (Eq. (3.13)) using the function \texttt{rmixGibbs} from the R package \texttt{bayesm}. We implemented LRVB in C++, making extensive use of \texttt{RcppEigen} \([\text{Bates and Eddelbuettel}, 2013]\). We evaluate our results both on simulated data and on the MNIST data set \([\text{LeCun et al.}, 1998]\).
Results. For simulations, we generated $N = 10000$ data points from $K = 2$ multivariate normal components in $P = 2$ dimensions. MFVB is expected to underestimate the marginal variance of $\mu$, $\Lambda$, and $\log(\pi)$ when the components overlap since that induces correlation in the posteriors due to the uncertain classification of points between the clusters. We check the covariances estimated with Eq. (3.7) against a Gibbs sampler, which we treat as the ground truth.

We performed 198 simulations, each of which had at least 500 effective Gibbs samples in each variable—calculated with the R tool effectiveSize from the coda package [Plummer et al., 2006]. The first three plots show the diagonal standard deviations, and the third plot shows the off-diagonal covariances. Note that the off-diagonal covariance plot excludes the MFVB estimates since most of the values are zero. Fig. (3.3) shows that the raw MFVB covariance estimates are often quite different from the Gibbs sampler results, while the LRVB estimates match the Gibbs sampler closely.

For a real-world example, we fit a $K = 2$ GMM to the $N = 12665$ instances of handwritten 0s and 1s in the MNIST data set. We used PCA to reduce the pixel intensities to $P = 25$ dimensions. Full details are provided in Appendix A.4. In this MNIST analysis, the $\Lambda$ standard deviations were under-estimated by MFVB but correctly estimated by LRVB (Fig. (3.3)); the other parameter standard deviations were estimated correctly by both and are not shown.

Figure 3.3: Posterior mean and covariance estimates on GMM simulation and MNIST data.

---

2The likelihood described in Section 3.3 is symmetric under relabeling. When the component locations and shapes have a real-life interpretation, the researcher is generally interested in the uncertainty of $\mu$, $\Lambda$, and $\pi$ for a particular labeling, not the marginal uncertainty over all possible re-labelings. This poses a problem for standard MCMC methods, and we restrict our simulations to regimes where label switching did not occur in our Gibbs sampler. The MFVB solution conveniently avoids this problem since the mean field assumption prevents it from representing more than one mode of the joint posterior.
Scaling experiments

We here explore the computational scaling of LRVB in more depth for the finite Gaussian mixture model (Section 3.3). In the terms of Section 3.2, \( \alpha \) includes the sufficient statistics from \( \mu, \pi, \) and \( \Lambda \), and grows as \( O(KP^2) \). The sufficient statistics for the variational posterior of \( \mu \) contain the \( P \)-length vectors \( \mu_k \), for each \( k \), and the \( (P+1)P/2 \) second-order products in the covariance matrix \( \mu_k \mu_k^T \). Similarly, for each \( k \), the variational posterior of \( \Lambda \) involves the \( (P+1)P/2 \) sufficient statistics in the symmetric matrix \( \Lambda_k \) as well as the term \( \log |\Lambda_k| \). The sufficient statistics for the posterior of \( \pi_k \) are the \( K \) terms \( \log \pi_k \). So, minimally, Eq. (3.7) will require the inverse of a matrix of size \( O(KP^2) \). The sufficient statistics for \( z \) have dimension \( K \times N \). Though the number of parameters thus grows with the number of data points, \( H_z = 0 \) for the multivariate normal (see Appendix A.3), so we can apply Eq. (3.11) to replace the inverse of an \( O(KN) \)-sized matrix with multiplication by the same matrix. Since a matrix inverse is cubic in the size of the matrix, the worst-case scaling for LRVB is then \( O(K^2) \) in \( K \), \( O(P^6) \) in \( P \), and \( O(N) \) in \( N \).

In our simulations (Fig. (3.4)) we can see that, in practice, LRVB scales linearly\(^3\) in \( N \) and approximately cubically in \( P \) across the dimensions considered\(^4\). The \( P \) scaling is presumably better than the theoretical worst case of \( O(P^6) \) due to extra efficiency in the numerical linear algebra. Note that the vertical axis of the leftmost plot is on the log scale. At all the values of \( N, K \) and \( P \) considered here, LRVB was at least as fast as Gibbs sampling and often orders of magnitude faster.

---

\(^3\)Since \( \sum_{k=1}^K \pi_k = 1 \), using \( K \) sufficient statistics involves one redundant parameter. However, this does not violate any of the necessary assumptions for Eq. (3.7), and it considerably simplifies the calculations. Note that though the perturbation argument of Section 3.1 requires the parameters of \( p(\theta|x) \) to be in the interior of the feasible space, it does not require that the parameters of \( p(x|\theta) \) be interior.

\(^4\)The Gibbs sampling time was linearly rescaled to the amount of time necessary to achieve 1000 effective samples in the slowest-mixing component of any parameter. Interestingly, this rescaling leads to increasing efficiency in the Gibbs sampling at low \( P \) due to improved mixing, though the benefits cease to accrue at moderate dimensions.

\(^5\)For numeric stability we started the optimization procedures for MFVB at the true values, so the time to compute the optimum in our simulations was very fast and not representative of practice. On real data, the optimization time will depend on the quality of the starting point. Consequently, the times shown for LRVB are only the times to compute the LRVB estimate. The optimization times were on the same order.
Figure 3.4: Scaling of LRVB and Gibbs on simulation data in both log and linear scales. Before taking logs, the line in the two lefthand (N) graphs is $y \propto x$, and in the righthand (P) graph, it is $y \propto x^3$. 
Chapter 4

A Swiss Army infinitesimal jackknife

Statistical machine learning methods are increasingly deployed in real-world problem domains where they are the basis of decisions affecting individuals’ employment, savings, health, and safety. Unavoidable randomness in data collection necessitates understanding how our estimates, and resulting decisions, might have differed had we observed different data. Both cross validation (CV) and the bootstrap attempt to diagnose this variation and are widely used in classical data analysis. But these methods are often prohibitively slow for modern, massive datasets, as they require running a learning algorithm on many slightly different datasets. In this work, we propose to replace these many runs with a single perturbative approximation. We show that the computation of this approximation is far cheaper than the classical methods, and we provide theoretical conditions that establish its accuracy.

Many data analyses proceed by minimizing a loss function of exchangeable data. Examples include empirical loss minimization and M-estimation based on product likelihoods. Since we typically do not know the true distribution generating the data, it is common to approximate the dependence of our estimator on the data via the dependence of the estimator on the empirical distribution. In particular, we often form a new, proxy dataset using random or deterministic modifications of the empirical distribution, such as randomly removing \( k \) datapoints for leave-\( k \)-out CV. A proxy dataset obtained in this way can be represented as a weighting of the original data. From a set of such proxy datasets we can obtain estimates of uncertainty, including estimates of bias, variance, and prediction accuracy.

As data and models grow, the cost of repeatedly solving a large optimization problem for a number of different values of weights can become impractically large. Conversely, though, larger datasets often exhibit greater regularity; in particular, under fairly general conditions, limit laws based on independence imply that an optimum exhibits diminishing dependence on any fixed set of data points. We use this observation to derive a linear approximation to resampling that needs
to be calculated only once, but which nonetheless captures the variability inherent in the repeated computations of classical CV. Our method is an instance of the infinitesimal jackknife (IJ), a general methodology that was historically a predecessor to cross-validation and the bootstrap [Jaeckel, 1972, Efron, 1982]. Part of our argument is that variants of the IJ should be reconsidered for modern large-scale applications because, for smooth optimization problems, the IJ can be calculated automatically with modern automatic differentiation tools [Baydin et al., 2018].

By using this linear approximation, we incur the cost of forming and inverting a matrix of second derivatives with size equal to the dimension of the parameter space, but we avoid the cost of repeatedly re-optimizing the objective. As we demonstrate empirically, this tradeoff can be extremely favorable in many problems of interest.

Our approach aims to provide a felicitous union of two schools of thought. In statistics, the IJ is typically used to prove normality or consistency of other estimators [Fernholz, 1983, Shao, 1993, Shao and Tu, 2012]. However, the conditions that are required for these asymptotic analyses to hold are prohibitively restrictive for machine learning—specifically, they require objectives with bounded gradients. A number of recent papers in machine learning have provided related linear approximations for the special case of leave-one-out cross-validation [Koh and Liang, 2017, Rad and Maleki, 2018, Beirami et al., 2017], though their analyses lack the generality of the statistical perspective.

We combine these two approaches by modifying the proof of the Fréchet differentiability of M-estimators developed by Clarke [1983]. Specifically, we adapt the proof away from the question of Fréchet differentiability within the class of all empirical distributions to the narrower problem of approximating the exact re-weighting on a particular dataset with a potentially restricted set of weights. This limitation of what we expect from the approximation is crucial; it allows us to bound the error in terms of a complexity measure of the set of derivatives of the observed objective function, providing a basis for non-asymptotic applications in large-scale machine learning, even for objectives with unbounded derivatives. Together with modern automatic differentiation tools, these results extend the use of the IJ to a wider range of practical problems. Thus, our “Swiss Army infinitesimal jackknife,” like the famous Swiss Army knife, is a single tool with many different functions.
4.1 Methods and results

Problem definition

We consider the problem of estimating an unknown parameter \( \theta \in \Omega \subseteq \mathbb{R}^D \), with a compact \( \Omega \) and a dataset of size \( N \). Our analysis will proceed entirely in terms of a fixed dataset, though we will be careful to make assumptions that will plausibly hold for all \( N \) under suitably well-behaved random sampling. We define our estimate, \( \hat{\theta} \in \Omega \), as the root of a weighted estimating equation. For each \( n = 1, \ldots, N \), let \( g_n(\theta) \) be a function from \( \Omega \) to \( \mathbb{R}^D \). Let \( w_n \) be a real number, and let \( w \) be the vector collecting the \( w_n \). Then \( \hat{\theta} \) is defined as the quantity that satisfies

\[
\hat{\theta}(w) := \theta \text{ such that } \frac{1}{N} \sum_{n=1}^{N} w_n g_n(\theta) = 0. \tag{4.1}
\]

We will impose assumptions below that imply at least local uniqueness of \( \hat{\theta}(w) \); see the discussion following Assumption 4.2 in Section 4.1.

As an example, consider a family of continuously differentiable loss functions \( f(\cdot, \theta) \) parameterized by \( \theta \) and evaluated at data points \( x_n, n = 1, \ldots, N \). If we want to solve the optimization problem \( \hat{\theta} = \arg\min_{\theta \in \Omega} \frac{1}{N} \sum_{n=1}^{N} f(x_n, \theta) \), then we take \( g_n(\theta) = \partial f(x_n, \theta) / \partial \theta \) and \( w_n \equiv 1 \). By keeping our notation general, we will be able to analyze a more general class of problems, such as multi-stage optimization (see Section 4.5). However, to aid intuition, we will sometimes refer to the \( g_n(\theta) \) as “gradients” and their derivatives as “Hessians.”

When equation (4.1) is not degenerate (we articulate precise conditions below), \( \hat{\theta} \) is a function of the weights through solving the estimating equation, and we write \( \hat{\theta}(w) \) to emphasize this. We will focus on the case where we have solved equation (4.1) for the weight vector of all ones, \( 1_w := (1, \ldots, 1) \), which we denote \( \hat{\theta}_1 := \theta(1_w) \).

A re-sampling scheme can be specified by choosing a set \( W \subseteq \mathbb{R}^N \) of weight vectors. For example, to approximate leave-k-out CV, one repeatedly computes \( \hat{\theta}(w) \) where \( w \) has \( k \) randomly chosen zeros and all ones otherwise. Define \( W_k \) as the set of every possible leave-k-out weight vector. Showing that our approximation is good for all leave-k-out analyses with probability one is equivalent to showing that the approximation is good for all \( w \in W_k \).

In the case of the bootstrap, \( W \) contains a fixed number \( B \) of randomly chosen weight vectors, \( w_b^{*} \overset{\text{iid}}{\sim} \text{Multinomial}(N, N^{-1}) \) for \( b = 1, \ldots, B \), so that \( \sum_{n=1}^{N} w_{bn}^{*} = N \) for each \( b \). Note that while \( w_n \) or \( w^{*}_{bn} \) are scalars, \( w_b^{*} \) is a vector of length \( N \). The distribution of \( \hat{\theta}(w^{*}_b) - \hat{\theta}(1_w) \) is then used to estimate the sampling
variation of $\hat{\theta}_1$. Define this set $W_B^* = \{w_1^*, \ldots, w_B^*\}$. Note that $W_B^*$ is stochastic and is a subset of all weight vectors that sum to $N$.

In general, $W$ can be deterministic or stochastic, may contain integer or non-integer values, and may be determined independently of the data or jointly with it. As with the data, our results hold for a given $W$, but in a way that will allow natural high-probability extensions to stochastic $W$.

**Linear approximation**

The main problem we solve is the computational expense involved in evaluating $\hat{\theta}(w)$ for all $w \in W$. Our contribution is to use only quantities calculated from $\hat{\theta}_1$ to approximate $\hat{\theta}(w)$ for all $w \in W$, without re-solving equation (4.1). Our approximation is based on the derivative $\frac{d\hat{\theta}(w)}{dw}$, whose existence depends on the derivatives of $g_n(\theta)$, which we assume to exist, and which we denote as $h_n(\theta) := \frac{\partial g_n(\theta)}{\partial \theta}$. We use this notation because $h_n(\theta)$ would be the Hessian of a term of the objective in the case of an optimization problem. We make the following definition for brevity.

**Definition 4.1.** The fixed point equation and its derivative are given respectively by

$$G(\hat{\theta}, w) := \frac{1}{N} \sum_{n=1}^{N} w_n g_n(\theta)$$

$$H(\hat{\theta}, w) := \frac{1}{N} \sum_{n=1}^{N} w_n h_n(\theta).$$

Note that $G(\hat{\theta}(w), w) = 0$ because $\hat{\theta}(w)$ solves equation (4.1) for $w$. We define $H_1 := H(\hat{\theta}_1, 1_w)$ and define the weight difference as $\Delta w = w - 1_w \in \mathbb{R}^N$.

When $H_1$ is invertible, one can use the implicit function theorem and the chain rule to show that the derivative of $\hat{\theta}(w)$ with respect to $w$ is given by

$$\left. \frac{d\hat{\theta}(w)}{dw^T} \right|_{1_w} \Delta w = -H_1^{-1} \frac{1}{N} \sum_{n=1}^{N} g_n(\hat{\theta}_1) \Delta w$$

$$= -H_1^{-1} G(\hat{\theta}_1, \Delta w).$$

This derivative allows us to form a first-order approximation to $\hat{\theta}(w)$ at $\hat{\theta}_1$.

**Definition 4.2.** Our linear approximation to $\hat{\theta}(w)$ is given by

$$\hat{\theta}_H(w) := \hat{\theta}_1 - H_1^{-1} G(\hat{\theta}_1, \Delta w).$$
We use the subscript “IJ” for “infinitesimal jackknife,” which is the name for this estimate in the statistics literature [Jaeckel, 1972, Shao, 1993]. Because \( \hat{\theta}_{IJ} \) depends only on \( \hat{\theta}_1 \) and \( \Delta w \), and not on solutions at any other values of \( w \), there is no need to re-solve equation (4.1). Instead, to calculate \( \hat{\theta}_{IJ} \) one must solve a linear system involving \( H_1 \). Recalling that \( \theta \) is \( D \)-dimensional, the calculation of \( H_1^{-1} \) (or a factorization that supports efficient solution of linear systems) can be \( O(D^3) \). However, once \( H_1^{-1} \) is calculated or \( H_1 \) is factorized, calculating our approximation \( \hat{\theta}_{IJ}(w) \) for each new weight costs only as much as a single matrix-vector multiplication. Furthermore, \( H_1 \) often has a sparse structure allowing \( H_1^{-1} \) to be calculated more efficiently than a worst-case scenario (see Section 4.5 for an example). In more high-dimensional examples with dense Hessian matrices, such as neural networks, one may need to turn to approximations such as stochastic second-order methods [Koh and Liang, 2017] [Agarwal et al., 2017] and conjugate gradient [Wright and Nocedal, 1999]. Indeed, even in relatively small or sparse problems, the vast bulk of the computation required to calculate \( \hat{\theta}_{IJ} \) is in the computation of \( H_1^{-1} \). We leave the important question of approximate calculation of \( H_1^{-1} \) for future work.

Assumptions and results

We now state our key assumptions and results, which are sufficient conditions under which \( \hat{\theta}_{IJ}(w) \) will be a good approximation to \( \hat{\theta}(w) \). We defer most proofs to Section 4.6. We use \( \|\cdot\|_{op} \) to denote the matrix operator norm, \( \|\cdot\|_2 \) to denote the \( L_2 \) norm, and \( \|\cdot\|_1 \) to denote the \( L_1 \) norm. For quantities like \( g \) and \( h \), which have dimensions \( N \times D \) and \( N \times D \times D \) respectively, we apply the \( L_p \) norm to the vectorized version of arrays. For example, \( \frac{1}{\sqrt{N}} \|h(\theta)\|_2 = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{D} \sum_{j=1}^{D} |h_n(\theta)|^2_{ij}} \) which is the square root of a sample average over \( n \in [N] \).

We state all assumptions and results for a fixed \( N \), a given estimating equation vector \( g(\theta) \), and a fixed class of weights \( W \). Although our analysis proceeds with these quantities fixed, we are careful to make only assumptions that can plausibly hold for all \( N \) and/or for randomly chosen \( W \) under appropriate regularity conditions.

**Assumption 4.1 (Smoothness).** For all \( \theta \in \Omega_\theta \), each \( g_n(\theta) \) is continuously differentiable in \( \theta \).

The smoothness in Assumption 4.1 is necessary for a local approximation like Definition 4.2 to have any hope of being useful.

**Assumption 4.2 (Non-degeneracy).** For all \( \theta \in \Omega_\theta \), \( H(\theta, 1_w) \) is non-singular, with \( \sup_{\theta \in \Omega_\theta} \|H(\theta, 1_w)^{-1}\|_{op} \leq C_{ap} < \infty \).
Without Assumption 4.2, the derivative in Definition 4.2 would not exist. For an optimization problem, Definition 4.2 amounts to assuming that the Hessian is strongly positive definite, and, in general, assures that the solution \( \hat{\theta} \) is unique.

Under our assumptions, we will show later that, additionally, \( \hat{\theta}(w) \) is unique in a neighborhood of \( \theta_1 \); see Lemma 4.6 of Section 4.6. Furthermore, by fixing \( C_{op} \), if we want to apply Assumption 4.2 for \( N \to \infty \), we will require that \( H_1 \) remains strongly positive definite.

**Assumption 4.3** (Bounded averages). There exist finite constants \( C_g \) and \( C_h \) such that 
\[
\sup_{\theta \in \Omega_o} \frac{1}{\sqrt{N}} \| g(\theta) \|_2 \leq C_g < \infty \quad \text{and} \quad \sup_{\theta \in \Omega_o} \frac{1}{\sqrt{N}} \| h(\theta) \|_2 \leq C_h < \infty.
\]

Assumption 4.3 essentially states that the sample variances of the gradients and Hessians are uniformly bounded. Note that it does not require that these quantities are bounded term-wise. For example, we allow \( \sup_n \| g_n(\theta) \|_2 \xrightarrow{N \to \infty} \infty \), as long as \( \sup_n \frac{1}{\sqrt{N}} \| g_n(\theta) \|_2^2 \) remains bounded. This is a key advantage of the present work over many past applications of the IJ to M-estimation, which require \( \sup_n \| g_n(\theta) \|_2 \) to be uniformly bounded for all \( N \) [Shao and Tu 2012, Beirami et al. 2017].

In both machine learning and statistics, \( \sup_n \| g_n(\theta) \|_2^2 \) is rarely bounded, though \( \frac{1}{N} \| g(\theta) \|_2^2 \) often is. As a simple example, suppose that \( \theta \in \mathbb{R}^1 \), \( x_n \sim \mathcal{N}(0,1) \), and \( g_n = \theta - x_n \), as would arise from the squared error loss \( f_n(x_n, \theta) = \frac{1}{2} (\theta - x_n)^2 \). Fix a \( \theta \) and let \( N \to \infty \). Then \( \sup_n \| g_n(\theta) \|_2^2 \to \infty \) because \( \sup_n |x_n| \to \infty \), but \( \frac{1}{N} \| g(\theta) \|_2^2 \to \theta^2 + 1 \) by the law of large numbers.

**Assumption 4.4** (Local smoothness). There exists a \( \Delta_\theta > 0 \) and a finite constant \( L_h \) such that,
\[
\| \theta - \hat{\theta}_1 \|_2 \leq \Delta_\theta \quad \text{implies that} \quad \frac{\| h(\theta) - h(\hat{\theta}_1) \|_2}{\sqrt{N}} \leq L_h \| \theta - \hat{\theta}_1 \|_2.
\]

The constants defined in Assumption 4.4 are needed to calculate our error bounds explicitly.

Assumptions 4.1–4.4 are quite general and should be expected to hold for many reasonable problems, including holding uniformly asymptotically with high probability for many reasonable data-generating distributions, as the following lemma shows.

**Lemma 4.1** (The assumptions hold under uniform convergence). Let \( \Omega_o \) be a compact set, and let \( g_n(\theta) \) be twice continuously differentiable IID random functions for \( n \in [N] \). (The function is random but \( \theta \) is not—for example, \( \mathbb{E}[g_n(\theta)] \) is still a function of \( \theta \).) Define \( r_n(\theta) := \frac{\partial^2 g_n(\theta)}{\partial \theta \partial \theta} \), so \( r_n(\theta) \) is a \( D \times D \times D \) tensor.

Assume that we can exchange integration and differentiation, that \( \mathbb{E}[h_n(\theta)] \) is non-singular for all \( \theta \in \Omega_o \), and that all of \( \mathbb{E}\left[ \sup_{\theta \in \Omega_o} \| g_n(\theta) \|_2^2 \right] \), \( \mathbb{E}\left[ \sup_{\theta \in \Omega_o} \| h_n(\theta) \|_2^2 \right] \), and \( \mathbb{E}\left[ \sup_{\theta \in \Omega_o} \| r_n(\theta) \|_2^2 \right] \) are finite.
Then $\lim_{N \to \infty} P (\text{Assumptions 4.1–4.4 hold}) = 1$.

Lemma 4.1 follows from the uniform convergence results of Theorems 9.1 and 9.2 in Keener [2011]. See Section 4.6 for a detailed proof. A common example to which Lemma 4.1 would apply is where $x_n$ are well-behaved IID data and $g_n(\theta) = \gamma(x_n, \theta)$ for an appropriately smooth estimating function $\gamma(\cdot, \theta)$. See Keener [2011, Chapter 9] for more details and examples, including applications to maximum likelihood estimators on unbounded domains.

Assumptions 4.1–4.4 apply to the estimating equation. We also require a boundedness condition for $W$.

Assumption 4.5 (Bounded weight averages). The quantity $\frac{1}{\sqrt{N}} \|w\|_2$ is uniformly bounded for $w \in W$ by a finite constant $C_w$.

Our final requirement is considerably more restrictive, and contains the essence of whether or not $\hat{\theta}_U(w)$ will be a good approximation to $\hat{\theta}(w)$.

Condition 4.1 (Set complexity). There exists a $\delta \geq 0$ and a corresponding set $W_\delta \subseteq W$ such that

$$\sup_{w \in W_\delta} \sup_{\theta \in \Omega_\theta} \left\| \frac{1}{N} \sum_{n=1}^{N} (w_n - 1) g_n(\theta) \right\|_1 \leq \delta \quad \text{and}$$

$$\sup_{w \in W_\delta} \sup_{\theta \in \Omega_\theta} \left\| \frac{1}{N} \sum_{n=1}^{N} (w_n - 1) h_n(\theta) \right\|_1 \leq \delta.$$ 

Condition 4.1 is central to establishing when the approximation $\hat{\theta}_U(w)$ is accurate. For a given $\delta$, $W_\delta$ will be the class of weight vectors for which $\hat{\theta}_U(w)$ is accurate to within order $\delta$. Trivially, $1_w \in W_\delta$ for $\delta = 0$, so $W_\delta$ is always non-empty, even for arbitrarily small $\delta$. The trick will be to choose a small $\delta$ that still admits a large class $W_\delta$ of weight vectors. In Section 4.2 we will discuss Condition 4.1 in more depth, but it will help to first state our main theorem.

Definition 4.3. The following constants are given by quantities in Assumptions 4.1–4.5.

$$C_U := 1 + DC_w L_h C_{op}$$

$$\Delta_\delta := \min \left\{ \Delta_0 C_{op}^{-1}, \frac{1}{2} C_U^{-1} C_{op}^{-1} \right\}.$$ 

Note that, although the parameter dimension $D$ occurs explicitly only once in Definition 4.3, all of $C_w$, $C_{op}$, and $L_h$ in general might also contain dimension
dependence. Additionally, the bound $\delta$ in Condition 4.1, a measure of the set complexity of the parameters, will typically depend on dimension. However, the particular place where the parameter dimension enters will depend on the problem and asymptotic regime, and our goal is to provide an adaptable toolkit for a wide variety of problems.

We are now ready to state our main result.

**Theorem 4.1** (Error bound for the approximation). *Under Assumptions 4.1–4.5 and Condition 4.1*

$$\delta \leq \Delta_\delta \Rightarrow \max_{w \in W_\delta} \left\| \hat{\theta}_{IJ}(w) - \hat{\theta}(w) \right\|_2 \leq 2C_{op}^2C_{IJ}\delta^2.$$

We stress that Theorem 4.1 bounds only the difference between $\hat{\theta}_{IJ}(w)$ and $\hat{\theta}(w)$. Theorem 4.1 alone does not guarantee that $\hat{\theta}_{IJ}(w)$ converges to any hypothetical infinite population quantity. We see this as a strength, not a weakness. To begin with, convergence to an infinite population requires stronger assumptions. Contrast, for example, the Fréchet differentiability work of [Clarke, 1983], on which our work is based, with the stricter requirements in the proof of consistency in [Shao, 1993]. Second, machine learning problems may not naturally admit a well-defined infinite population, and the dataset at hand may be of primary interest. Finally, by analyzing a particular sample rather than a hypothetical infinite population, we can bound the error in terms of the quantities $C_{IJ}$ and $\Delta_\delta$, which can actually be calculated from the data at hand.

Still, Theorem 4.1 is useful to prove asymptotic results about the difference $\left\| \hat{\theta}_{IJ}(w) - \hat{\theta}(w) \right\|_2$. As an illustration, we now show that the uniform consistency of leave-$k$-out CV follows from Theorem 4.1 by a straightforward application of Hölder’s inequality.

**Corollary 4.1** (Consistency for leave-$k$-out CV). *Assume that Assumptions 4.1–4.5 hold uniformly for all $N$. Fix an integer $k$, and let

$$W_k := \{w : w_n = 0 \text{ in } k \text{ entries and 1 otherwise}\}.$$

Then, for all $N$, there exists a constant $C_K$ such that

$$\sup_{w \in W_k} \left\| \hat{\theta}_{IJ}(w) - \hat{\theta}(w) \right\|_2 \leq C_K \frac{\|g\|_\infty^2}{N^2} \leq C_K \max \left\{C_g, C_h\right\}^2 \frac{1}{N}.$$
CHAPTER 4. INFINITESIMAL JACKKNIFE

Proof. For \( w \in W_k \), \( \frac{\|\Delta w\|_2}{N} = \sqrt{\frac{K}{N}} \). Define \( C_{gh} := \max\{C_g, C_h\} \). By Assumption 4.3 \( \|g\|_2 / \sqrt{N} \leq C_{gh} \) and \( \|h\|_2 / \sqrt{N} \leq C_{gh} \) for all \( N \). By Hölder’s inequality,

\[
\sup_{w \in W} \sup_{\theta \in \Omega} \left\| \frac{1}{N} \sum_{n=1}^{N} (w_n - 1) g_n (\theta) \right\|_1 \\
\leq \sup_{w \in W} \|w - 1\|_1 \sup_{\theta \in \Omega} \frac{\|g\|_{\infty}}{N} = K \frac{\|g\|_{\infty}}{\sqrt{N}} \leq K C_{gh} \sqrt{N},
\]

with a similar bound for \( \|h\|_2 \). Consequently, for \( N \) large enough, Condition 4.1 is satisfied with \( W_\delta = W_k \) and either \( \delta = K \frac{\|g\|_{\infty}}{\sqrt{N}} \) or \( \delta = K C_{gh} \frac{\sqrt{N}}{\sqrt{N}} \). The result then follows from Theorem 4.1.

4.2 Examples

The moral of Theorem 4.1 is that, under Assumptions 4.1–4.5 and Condition 4.1, \( \|\hat{\theta}_{IJ} - \hat{\theta} (w)\| = O(\delta^2) \) for \( w \in W_\delta \). That is, if we can make \( \delta \) small enough, \( W_\delta \) big enough, and still satisfy Condition 4.1 then \( \hat{\theta}_{IJ} (w) \) is a good approximation to \( \hat{\theta} (w) \) for “most” \( w \), where “most” is defined as the size of \( W_\delta \). So it is worth taking a moment to develop some intuition for Condition 4.1. We have already seen in Corollary 4.1 that \( \hat{\theta}_{IJ} \) is, asymptotically, a good approximation for leave-k-out CV uniformly in \( W \). We now discuss some additional cases: first, a worst-case example for which \( \hat{\theta}_{IJ} \) is not expected to work, second the bootstrap, and finally we revisit leave-one-out cross validation in the context of these other two methods.

First, consider a pathological example. Let \( W_{full} \) be the set of all weight vectors that sum to \( N \). Let \( n^* = \max_{n \in [N]} \|g_n (\hat{\theta}_1)\|_1 \) be the index of the gradient term with the largest \( L_1 \) norm, and let \( w_{n^*} = N \) and \( w_n = 0 \) for \( n \neq n^* \). Then

\[
\sup_{\theta \in \Omega} \left\| \frac{1}{N} \sum_{n=1}^{N} (w_n - 1) g_n (\theta) \right\|_1 \\
= \sup_{\theta \in \Omega} \left\| g_n^* (\theta) - \frac{1}{N} \sum_{n=1}^{N} g_n (\theta) \right\|_1 \geq \left\| g_n^* (\hat{\theta}_1) \right\|_1.
\]

(The last inequality uses the fact that \( G (\hat{\theta}_1, 1_w) = 0 \).) In this case, unless the largest gradient, \( \left\| g_n^* (\hat{\theta}_1) \right\|_1 \), is small, Condition 4.1 will not be satisfied for small \( \delta \), and we would not expect \( \hat{\theta}_{IJ} \) to be a good estimate for \( \hat{\theta} (w) \) for all \( w \in W_{full} \).
The class $W_{\text{full}}$ is too expressive. In the language of Condition 4.1, for some small fixed $\delta$, $W_\delta$ will be some very restricted subset of $W_{\text{full}}$ in most realistic situations.

Now, suppose that we are using $B$ bootstrap weights,

$$w^*_b \overset{iid}{\sim} \text{Multinomial } (N, N^{-1}),$$

for $b = 1, \ldots, B$, and analyzing an optimization problem as defined in Section 4.1. For a given $w^*_b$, a dataset $x_1^*, \ldots, x_N^*$ formed by taking $w^*_{b,n}$ copies of datapoint $x_n$ is equivalent in distribution to $N$ IID samples with replacement from the empirical distribution on $(x_1, \ldots, x_N)$. In this notation, we then have

$$\frac{1}{N} \sum_{n=1}^{N} (w^*_b - 1) g_n(\theta) = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial f(\theta, x^*_n)}{\partial \theta} - \frac{1}{N} \sum_{n=1}^{N} \frac{\partial f(\theta, x_n)}{\partial \theta}.$$  

In this case, Condition 4.1 is a uniform bound on a centered empirical process of derivatives of the objective function. Note that estimating sample variances by applying the IJ with bootstrap weights is equivalent to the ordinary delta method based on an asymptotic normal approximation [Efron 1982, Chapter 21]. In order to provide an approximation to the bootstrap that retains benefits (such as the faster-than-normal convergence to the true sampling distribution described by [Hall 2013]), one must consider higher-ordered Taylor expansions of $\hat{\theta}(w)$. We leave this for future work.

Finally, let us return to leave-one-out CV. In this case, $w_n - 1$ is nonzero for exactly one entry. Again, we can choose to leave out the adversarially-chosen $n^*$ as in the first pathological example. However, unlike the pathological example, the leave-one-out CV weights are constrained to be closer to $1_w$—specifically, we set $w_{n^*} = 0$, and let $w$ be one elsewhere. Then Condition 4.1 requires $\sup_{\theta \in \Omega_\theta} \| \frac{1}{N} g_{n^*}(\theta) \|_1 \leq \delta$. In contrast to the pathological example, this supremum will get smaller as $N$ increases as long as $\| g_{n^*}(\theta) \|_1$ grows more slowly than $N$. For this reason, we expect leave-one-out (and, indeed, leave-$k$-out for fixed $k$) to be accurately approximated by $\hat{\theta}_{IJ}$ in many cases of interest, as stated in Corollary 4.1.

### 4.3 Related work

Although the idea of forming a linear approximation to the re-weighting of an M-estimator has a long history, we nevertheless contribute in a number of ways. By limiting ourselves to approximating the exact reweighting on a particular dataset,
we both loosen the strict requirements from the statistical literature and generalize the existing results from the machine learning literature.

The jackknife is often favored over the IJ in the statistics literature because of the former’s simple computational approach, as well as perceived difficulties in calculating the necessary derivatives when some of the parameters are implicitly defined via optimization [Shao and Tu, 2012, Chapter 2.1] (though exceptions exist; see, e.g., Wager et al. [2014]). The brute-force approach of the jackknife is, however, a liability in large-scale machine learning problems, which are generally extremely expensive to re-optimize. Furthermore, and critically, the complexity and tedium of calculating the necessary derivatives is entirely eliminated by modern automatic differentiation [Baydin et al., 2018, Maclaurin et al., 2015].

Our work is based on the proof of the Fréchet differentiability of M-estimators of Clarke [1983]. In classical statistics, Fréchet differentiability is typically used to describe the asymptotic behavior of functionals of the empirical distribution in terms of a functional [Mises, 1947, Fernholz, 1983]. Since Clarke [1983] was motivated by such asymptotic questions, he studied the Fréchet derivative evaluated at a continuous probability distribution for function classes that included delta functions. This focus led to the requirement of a bounded gradient. However, unbounded gradients are ubiquitous in both statistics and machine learning, and an essential contribution of the current paper is to remove the need for bounded gradients.

There exist proofs of the consistency of the (non-infinitesimal) jackknife that allow for unbounded gradients. For example, it is possible that the proofs of Reeds [1978], which require a smoothness assumption similar to our Assumption 4.4, could be adapted to the IJ. However, the results of Reeds [1978]—as well as those of Clarke [1983] and subsequent applications such as those of Shao and Tu [2012]—are asymptotic and applicable only to IID data. By providing finite sample results for a fixed dataset and weight set, we are able to provide a template for proving accuracy bounds for more generic probability distributions and re-weighting schemes.

A number of recent machine learning papers have derived approximate linear versions of leave-one-out estimators. Koh and Liang [2017] consider approximating the effect of leaving out one observation at a time to discover influential observations and construct adversarial examples, but provide little supporting theory. Beirami et al. [2017] provide rigorous proofs for an approximate leave-one-out CV estimator; however, their estimator requires computing a new inverse Hessian for each new weight at the cost of a considerable increase in computational complexity. Like the classical statistics literature, Beirami et al. [2017] assume that the gradients are bounded for all $N$. When $\|g\|_\infty^2$ in Corollary 4.1 is finite for all $N$, we achieve the same $N^{-2}$ rate claimed by Beirami et al. [2017] for leave-one-out CV although we use only a single matrix inverse. Rad and Maleki [2018] also approximate leave-
one-out CV, and prove tighter bounds for the error of their approximation than we do, but their work is customized to leave-one-out CV and makes much more restrictive assumptions (e.g., Gaussianity).

### 4.4 Simulated experiments

We begin the empirical demonstration of our method on two simple generalized linear models: logistic and Poisson regression. In each case, we generate a synthetic dataset \( Z = \{(x_n, y_n)\}_{n=1}^N \) from parameters \((\theta, b)\), where \(\theta \in \mathbb{R}^{100}\) is a vector of regression coefficients and \(b \in \mathbb{R}\) is a bias term. In each experiment, \(x_n \in \mathbb{R}^{100}\) is drawn from a multivariate Gaussian, and \(y_n\) is a scalar drawn from a Bernoulli distribution with the logit link or from a Poisson distribution with the exponential link.

For a ground truth, we generate a large test set with \(N = 100,000\) datapoints to measure the true generalization error. We show in Fig. (4.1) that, over 50 randomly generated datasets, our approximation consistently underestimates the actual error predicted by exact leave-one-out CV; however, the difference is small relative to the improvements they both make over the error evaluated on the training set.

Fig. (4.2) shows the relative timings of our approximation and exact leave-one-out CV on logistic regression with datasets of increasing size. The time to run our approximation is roughly an order of magnitude smaller.

### 4.5 Genomics experiments

We now consider a genomics application in which we use CV to choose the degree of a spline smoother when clustering time series of gene expression data. Code and instructions to reproduce our results can be found in the git repository rgitordan/AISTATS2019SwissArmyIJ. The application is also described in detail in Appendix A.7.

We use a publicly available data set of mice gene expression [Shoemaker et al., 2015] in which mice were infected with influenza virus, and gene expression was assessed several times after infection. The observed data consists of expression levels \(y_{gt}\) for genes \(g = 1, \ldots, n_g\) and time points \(t = 1, \ldots, n_t\). In our case \(n_g = 1000\) and \(n_t = 14\). Many genes behave the same way; thus, clustering the genes by the pattern of their behavior over time allows dimensionality reduction that

---

1Leave-one-out CV may not be the most appropriate estimator of generalization error in this setting [Rosset and Tibshirani, 2018], but this section is intended only to provide simple illustrative examples.
can facilitate interpretation. Consequently, we wish to first fit a smoothed regression line to each gene and then cluster the results. Following Luan and Li [2003], we model the time series as a gene-specific constant additive offset plus a B-spline basis of degree 3, and the task is to choose the B-spline basis degrees of freedom using cross-validation on the time points.

Our analysis runs in two stages—first, we regress the genes on the spline basis, and then we cluster a transformed version of the regression fits. By modeling in two stages, we both speed up the clustering and allow for the use of flexible transforms of the fits. We are interested in choosing the smoothing parameter using CV on the time points. Both the time points and the smoothing parameter enter the regression objective directly, but they affect the clustering objective only through the optimal regression parameters. Because the optimization proceeds in two stages, the fit is not the optimum of any single objective function. However, it can still be represented as an M-estimator (see Appendix A.7).

We implemented the model in `scipy` [Jones et al., 2001] and computed all deriva-
Figure 4.2: Simulated data: timing results.

In conjunction with autograd [Maclaurin et al., 2015], we note that the match between “exact” cross-validation (removing time points and re-optimizing) and the IJ was considerably improved by using a high-quality second-order optimization method. In particular, for these experiments, we employed the Newton conjugate-gradient trust region method [Wright and Nocedal, 1999, Chapter 7.1] as implemented by the method trust-ncg in scipy.optimize, preconditioned by the Cholesky decomposition of an inverse Hessian calculated at an initial approximate optimum. The Hessian used for the preconditioner was with respect to the clustering parameters only and so could be calculated quickly, in contrast to the $H_1$ matrix used for the IJ, which includes the regression parameters as well. We found that first-order or quasi-Newton methods (such as BFGS) often got stuck or terminated at points with fairly large gradients. At such points our method does not apply in theory nor, we found, very well in practice.

Fig. (4.3) shows that the IJ is a reasonably good approximation to the test set error. In particular, both the IJ and exact CV capture the increase in test error for $df = 8$, which is not present in the training error. Thus we see that, like exact CV, the IJ is able to prevent overfitting. Though the IJ underestimates exact CV, we note that it differs from exact CV by no more than exact CV itself differs from the true

\[\text{Fig. (4.3) shows that the IJ is a reasonably good approximation to the test set error. In particular, both the IJ and exact CV capture the increase in test error for } df = 8, \text{ which is not present in the training error. Thus we see that, like exact CV, the IJ is able to prevent overfitting. Though the IJ underestimates exact CV, we note that it differs from exact CV by no more than exact CV itself differs from the true}\]
quantity of interest, the test error.

The timing results for the genomics experiment are shown in Fig. (4.4). For this particular problem with approximately 39,000 parameters (the precise number depends on the degrees of freedom), finding the initial optimum takes about 42 seconds. The cost of finding the initial optimum is shared by exact CV and the IJ, and, as shown in Fig. (4.4), is a small proportion of both.

The principle time cost of the IJ is the computation of $H_1$. Computing and inverting a dense matrix of size 39,000 would be computationally prohibitive. But, for the regression objective, $H_1$ is extremely sparse and block diagonal, so computing $H_1$ in this case took only around 360 seconds. Inverting $H_1$ took negligible time. Once we have $H_1^{-1}$, obtaining the subsequent IJ approximations is nearly instantaneous.

The cost of refitting the model for exact CV varies by degrees of freedom (increasing degrees of freedom increases the number of parameters) and the number of left-out points (an increasing number of left-out datapoints increases the number
of refits). As can be seen in Fig. (4.4), for low degrees of freedom and few left-out points, the cost of re-optimizing is approximately the same as the cost of computing $H_1$. However, as the degrees of freedom and number of left-out points grow, the cost of exact CV increases to as much as an order of magnitude more than that of the IJ.

### 4.6 Detailed assumptions, lemmas, and proofs

#### Tools

We begin by stating two general propositions that will be useful. First, we show that a version of Cauchy-Schwartz can be applied to weighted sums of tensors.

**Proposition 4.2.** Tensor array version of Hölder’s inequality. Let $w$ be an array of scalars and let $\alpha = (\alpha_1, \ldots, \alpha_N)$ be an array of tensors, were each $\alpha_n$ is indexed by $i = 1, \ldots, D_A$ (i may be a multi-index—e.g., if $A$ is a $D \times D$ matrix, then $i = (j, k)$, for $j, k \in [D]$ and $D_A = D^2$). Let $p, q \in [1, \infty]$ be two numbers such
that $p^{-1} + q^{-1} = 1$. Then
\[
\left\| \frac{1}{N} \sum_{n=1}^{N} w_n a_n \right\|_1 \leq \frac{D_A^p}{N} \|w\|_p \|a\|_q .
\]

In particular, with $p = q = 2$,
\[
\left\| \frac{1}{N} \sum_{n=1}^{N} w_n a_n \right\|_1 \leq \sqrt{D_A} \|w\|_2 \|a\|_2 \sqrt{N} / \sqrt{N}.
\]

Proof. The conclusion follows from the ordinary Hölder’s inequality applied term-wise to $n$ and Jensen’s inequality applied to the indices $i$.

\[
\left\| \frac{1}{N} \sum_{n=1}^{N} w_n a_n \right\|_1 = \frac{D_A}{N} \sum_{i=1}^{D_A} \left| \sum_{n=1}^{N} w_n \right|_p^{\frac{1}{p}} \left( \sum_{n=1}^{N} |(a_n)_i|^q \right)^{\frac{1}{q}} \quad \text{(Hölder)}
\]
\[
= \frac{1}{N} \|w\|_p D_A \sum_{i=1}^{D_A} \left( \sum_{n=1}^{N} |(a_n)_i|^q \right)^{\frac{1}{q}}
\]
\[
\leq \frac{1}{N} \|w\|_p D_A \left( \frac{1}{D_A} \sum_{i=1}^{D_A} \sum_{n=1}^{N} |(a_n)_i|^q \right)^{\frac{1}{q}} \quad \text{(Jensen applied to $i$)}
\]
\[
= \frac{1}{N} \|w\|_p D_A \left( \frac{1}{D_A} \sum_{n=1}^{N} \|a_n\|_q^q \right)^{\frac{1}{q}}
\]
\[
= \frac{1}{N} \|w\|_p D_A^{1 - \frac{q}{2}} \|a\|_q
\]
\[
= \frac{D_A^{\frac{1}{p}}}{N} \|w\|_p \|a\|_q .
\]

Next, we prove a relationship between the term-wise difference between matrices and the difference between their operator norms. It is well-known that the minimum eigenvalue of a non-singular matrix is continuous in the entries of the matrix. In the next proposition, we quantify this continuity for the $L_1$ norm.
Proposition 4.3. Let $A$ and $B$ be two matrices. Let $\|A^{-1}\|_{op} \leq C_{op}$ for some finite $C_{op}$. Then

$$\|A - B\|_1 \leq \frac{1}{2}C_{op}^{-1} \implies \|B^{-1}\|_{op} \leq 2C_{op}.$$  

Proof. We will use [Schott 2016, Theorem 5.20] and the associated discussion, which states the following general result. Take any matrix norm $\|\cdot\|$ that satisfies $\|I\| = 1$, where $I$ is the identity matrix. Then if $\|A^{-1}\| \|A - B\| < 1$, then

$$\|A^{-1} - (A - B)^{-1}\| \leq \frac{\|A^{-1}\| \|A - B\|}{1 - \|A^{-1}\| \|A - B\|} \|A^{-1}\|. \quad (4.2)$$

We will apply equation (4.2) using the operator norm $\|\cdot\|_{op}$, for which $\|I\|_{op} = 1$. First, note that

$$\|A^{-1}\|_{op} \|A - B\|_{op} \leq \|A^{-1}\|_{op} \|A - B\|_1 \quad \text{(ordering of matrix norms)}$$

$$\leq \frac{1}{2}C_{op}C_{op}^{-1}$$

$$= \frac{1}{2},$$

so we can apply equation (4.2). Then

$$\|B^{-1}\|_{op} \leq \|B^{-1} - A^{-1}\|_{op} + \|A^{-1}\|_{op} \quad \text{(triangle inequality)}$$

$$\leq \frac{\|A^{-1}\|_{op} \|A - B\|_{op}}{1 - \|A^{-1}\|_{op} \|A - B\|_{op}} \|A^{-1}\|_{op} + \|A^{-1}\|_{op} \quad \text{(Equation 4.2)}$$

$$\leq \left( \frac{1}{2} + 1 \right) \|A^{-1}\|$$

$$\leq 2C_{op}. \quad \square$$

Lemmas

We now prove some useful consequences of our assumptions. The proof roughly proceeds for all $w \in W_{\delta}$ by the following steps:

1. When $\delta$ is small we can make $\|\hat{\theta}(w) - \hat{\theta}_1\|_2$ small. (Lemma 4.2 below.)

2. When $\|\theta - \hat{\theta}_1\|_2$ is small, then the derivatives $H(\theta, w)$ are close to their optimal value, $H(\hat{\theta}_1, 1_w)$. (Lemma 4.3 and Lemma 4.4 below.)
3. When the derivatives are close to their optimal values, then $H(\theta, w)$ is uniformly non-singular. (Lemma 4.5 below.)

4. When the derivatives are close to their optimal values and $H(\theta, w)$ is uniformly non-singular we can control the error in $\hat{\theta}_W - \hat{\theta}(w)$ in terms of $\delta$. (Theorem 4.5 below.)

We begin by showing that the difference between $\hat{\theta}(w)$ and $\hat{\theta}_1$ for $w \in W_\delta$ can be made small by making $\delta$ from Condition 4.1 small.

**Lemma 4.2.** Small parameter changes. Under Assumptions 4.1—4.3 and Condition 4.1,

$$\text{for all } w \in W_\delta, \quad \|\hat{\theta}(w) - \hat{\theta}_1\|_2 \leq C_{op} \delta.$$  

**Proof.** By a first-order Taylor expansion in $\theta$, for some $\tilde{\theta}$ such that $\|\tilde{\theta} - \hat{\theta}_1\|_2 \leq \|\hat{\theta}(w) - \hat{\theta}_1\|_2$,

$$G\left(\hat{\theta}(w), 1_w\right) = G\left(\hat{\theta}_1, 1_w\right) + H\left(\hat{\theta}, 1_w\right)\left(\hat{\theta}(w) - \hat{\theta}_1\right).$$

By Assumption 4.2, $H(\tilde{\theta}, 1_w)$ is non-singular. A little manipulation, together with the fact that $G\left(\hat{\theta}(w), w\right) = G\left(\hat{\theta}_1, 1_w\right) = 0$ gives

$$G\left(\hat{\theta}(w), 1_w\right) - G\left(\hat{\theta}(w), w\right) = H\left(\hat{\theta}, 1_w\right)\left(\hat{\theta}(w) - \hat{\theta}_1\right) \Rightarrow \hat{\theta}(w) - \hat{\theta}_1 = H\left(\hat{\theta}, 1_w\right)^{-1}\left(G\left(\hat{\theta}(w), 1_w\right) - G\left(\hat{\theta}(w), w\right)\right).$$
Applying Condition 4.1 and Assumption 4.2

\[
\| \hat{\theta}(w) - \hat{\theta}_1 \|_2 = \left\| H \left( \hat{\theta}, 1_w \right)^{-1} \left( G \left( \hat{\theta}(w), 1_w \right) - G \left( \hat{\theta}(w), w \right) \right) \right\|_2 \\
\leq \left\| H \left( \hat{\theta}, 1_w \right)^{-1} \right\|_{\text{op}} \left\| \left( G \left( \hat{\theta}(w), 1_w \right) - G \left( \hat{\theta}(w), w \right) \right) \right\|_2 \\
\leq \sup_{\theta \in \Omega} \left\| H \left( \theta, 1_w \right)^{-1} \right\|_{\text{op}} \left\| \left( G \left( \hat{\theta}(w), 1_w \right) - G \left( \hat{\theta}(w), w \right) \right) \right\|_2 \\
\leq C_{\text{op}} \left\| G \left( \hat{\theta}(w), 1_w \right) - G \left( \hat{\theta}(w), w \right) \right\|_2 \text{ (Assumption 4.2)} \\
\leq C_{\text{op}} \left\| G \left( \hat{\theta}(w), 1_w \right) - G \left( \hat{\theta}(w), w \right) \right\|_1 \text{ (relation between norms)} \\
\leq C_{\text{op}} \sup_{\theta \in \Omega} \left\| G \left( \theta, 1_w \right) - G \left( \theta, w \right) \right\|_1 \\
\leq C_{\text{op}} \delta. \text{ (Condition 4.1)}.
\]

Because we will refer to it repeatedly, we give the set of \( \theta \) defined in Lemma 4.2 a name.

**Definition 4.4.** For a given \( \delta \), define the region around \( \hat{\theta}_1 \) given by Lemma 4.2 as

\[
B_{C_{\text{op}}\delta} := \left\{ \theta : \| \theta - \hat{\theta}_1 \|_2 \leq C_{\text{op}}\delta \right\} \cap \Omega_{\theta}.
\]

In other words, Lemma 4.2 states that Condition 4.1 implies \( \hat{\theta}(w) \in B_{C_{\text{op}}\delta} \) when \( w \in W_\delta \).

Next, we show that closeness in \( \theta \) will mean closeness in \( H(\theta, w) \).

**Lemma 4.3.** Boundedness and continuity. Under Assumptions 4.1–4.5 and Condition 4.1

for all \( \theta \in B_{\Delta \theta} \),

\[
\sup_{w \in W} \left\| H(\theta, w) - H(\hat{\theta}_1, w) \right\|_1 \leq DC_wL_h \left\| \theta - \hat{\theta}_1 \right\|_2.
\]
Proof. For $\theta \in B_{\Delta_\theta}$,

$$\sup_{w \in W} \left\| H(\theta, w) - H(\hat{\theta}_1, w) \right\|_1$$

$$= \sup_{w \in W} \left\| \frac{1}{N} \sum_{n=1}^{N} w_n \left( h_n(\theta) - h_n(\hat{\theta}_1) \right) \right\|_1 \quad \text{(by definition)}$$

$$\leq D \sup_{w \in W} \frac{\|w\|_2}{\sqrt{N}} \frac{\|h(\theta) - h(\hat{\theta}_1)\|_2}{\sqrt{N}} \quad \text{(Proposition 4.2)}$$

$$\leq DC_w \frac{\|h(\theta) - h(\hat{\theta}_1)\|_2}{\sqrt{N}} \quad \text{(Assumption 4.5)}$$

$$\leq DC_w L_h \left\| \theta - \hat{\theta}_1 \right\|_2 \quad \text{(Assumption 4.4 and $\theta \in B_{\Delta_\theta}$).}$$

We now combine Lemma 4.2 and Lemma 4.3 to show that $H(\theta, w)$ is close to its value at the solution $H(\hat{\theta}_1, 1_w)$ for sufficiently small $\delta$ and for all $\theta \in B_{C_{\text{op}}\delta}$.

**Lemma 4.4.** Bounds for difference in parameters. Under Assumptions 4.1–4.5 and Condition 4.1 if $\delta \leq \Delta_\theta C_{\text{op}}^{-1}$, then

$$\sup_{\theta \in B_{C_{\text{op}}\delta}} \sup_{w \in W_\delta} \left\| H(\theta, w) - H(\hat{\theta}_1, 1_w) \right\|_1 \leq (1 + DC_w L_h C_{\text{op}}) \delta.$$ 

**Proof.** By Lemma 4.2, $\delta \leq \Delta_\theta C_{\text{op}}^{-1}$ implies that $C_{\text{op}}\delta \leq \Delta_\theta$ and so $B_{C_{\text{op}}\delta} \subseteq B_{\Delta_\theta}$. Consequently, we can apply Lemma 4.3

$$\sup_{\theta \in B_{C_{\text{op}}\delta}} \sup_{w \in W_\delta} \left\| H(\theta, w) - H(\hat{\theta}_1, w) \right\|_1 \leq \sup_{\theta \in B_{\Delta_\theta}} \sup_{w \in W_\delta} \left\| H(\theta, w) - H(\hat{\theta}_1, w) \right\|_1$$

$$\leq DC_w L_h \left\| \theta - \hat{\theta}_1 \right\|_2 \quad \text{(Lemma 4.3)}$$

$$\leq DC_w L_h C_{\text{op}}\delta \quad \text{(because $\theta \in B_{C_{\text{op}}\delta}$).}$$
CHAPTER 4. INFINITESIMAL JACKKNIFE

Next, we can use this to write
\[
\sup_{\theta \in B_{\delta}^{\cop}} \sup_{w \in W_{\delta}} \left\| H(\theta, w) - H(\hat{\theta}_1, 1_w) \right\|_1
\]
\[
= \sup_{\theta \in B_{\delta}^{\cop}} \sup_{w \in W_{\delta}} \left\| H(\theta, w) - H(\theta, 1_w) + H(\theta, 1_w) - H(\hat{\theta}_1, 1_w) \right\|_1
\]
\[
\leq \sup_{\theta \in B_{\delta}^{\cop}} \sup_{w \in W_{\delta}} \left\| H(\theta, 1_w) - H(\hat{\theta}_1, 1_w) \right\|_1
\]
\[
\leq \sup_{\theta \in \Omega_{\delta}} \sup_{w \in W_{\delta}} \left\| H(\theta, w) - H(\theta, 1_w) \right\|_1 + \sup_{\theta \in B_{\delta}^{\cop}} \sup_{w \in W_{\delta}} \left\| H(\theta, 1_w) - H(\hat{\theta}_1, 1_w) \right\|_1
\]
\[
\leq \delta + \sup_{\theta \in B_{\delta}^{\cop}} \sup_{w \in W_{\delta}} \left\| H(\theta, 1_w) - H(\hat{\theta}_1, 1_w) \right\|_1 \quad \text{(Condition 4.1)}
\]
\[
\leq \delta + DC_{\delta} L_h C_{\delta} \delta.
\]

The constant that appears multiplying \( \delta \) at the end of the proof of Lemma 4.4 will appear often in what follows, so we give it the special name \( C_{IJ} \) in Definition 4.3.

Note that Lemma 4.4 places a condition on how small \( \delta \) must be in order for our regularity conditions to apply. Lemma 4.2 will guarantee that \( \hat{\theta}(w) \in B_{\delta}^{\cop} \), but if we are not able to make \( \delta \) arbitrarily small in Condition 4.1, then we are not guaranteed to ensure that \( B_{\delta}^{\cop} \subseteq B_{\delta}^{\theta} \), will not be able to assume Lipschitz continuity, and none of our results will apply.

Next, using Lemma 4.4, we can extend the operator bound on \( H^{-1}_1 \) from Assumption 4.2 to \( H(\theta, w)^{-1} \) for all \( w \in W_{\delta} \), not only for \( w = 1_w \).

**Lemma 4.5. Uniform invertibility of the Hessian.** Under Assumptions 4.1–4.5 and Condition 4.1 if \( \delta \leq \min \left\{ \Delta_{\theta} C_{\delta}^{-1}, \frac{1}{2} C_{IJ}^{-1} C_{\delta}^{-1} \right\} \), then
\[
\sup_{\theta \in B_{\delta}^{\cop}} \sup_{w \in W_{\delta}} \left\| H(\theta, w)^{-1} \right\|_{\op} \leq 2C_{\delta}.
\]

**Proof.** By Assumption 4.2 \( \left\| H(\hat{\theta}_1, 1_w)^{-1} \right\|_{\op} \leq C_{\delta} \). So by Proposition 4.3, it will suffice to select \( \delta \) so that
\[
\sup_{\theta \in B_{\delta}^{\cop}} \sup_{w \in W_{\delta}} \left\| H(\theta, w) - H(\hat{\theta}_1, 1_w) \right\|_1 \leq \frac{1}{2} C_{\delta}^{-1}. \quad (4.3)
\]
When we can apply Lemma 4.4, we have
\[
\sup_{\theta \in B_{C_{op}^{-1}}} \sup_{w \in W} \left\| H(\theta, w) - H(\hat{\theta}_1, 1_w) \right\|_1 \leq C_1 \delta.
\]

So \( H(\theta, w) \) will satisfy equation (4.3) if we can apply Lemma 4.4 and if
\[
\delta \leq \frac{1}{2} C_{op}^{-1} C_{U}^{-1}.
\]

To apply Lemma 4.4 we additionally require that \( \delta \leq \Delta_{\theta} C_{op}^{-1} \). By taking \( \delta \leq \min \{ \Delta_{\theta} C_{op}^{-1}, \frac{1}{2} C_{op}^{-1} C_{U}^{-1} \} \), we satisfy equation (4.3) and the result follows.

At last, the upper bound on \( \delta \) will be sufficient to control the error terms in our approximation. For compactness, we give it the name \( \Delta_{\delta} \) in Definition 4.3.

Finally, we state a result that will allow us to define derivatives of \( \hat{\theta}(w) \) with respect to \( w \).

**Lemma 4.6.** Inverse function theorem. Under Assumptions 4.1–4.5 and Condition 4.1, and for \( \delta \leq \Delta_{\delta} \), there exists a continuous, differentiable function of \( w \), \( \hat{\theta}(w) \), such that, for all \( w \in W \), \( G(\hat{\theta}(w), w) = 0 \).

*Proof.* This follows from Lemma 4.5 and the implicit function theorem.

By definition, \( \hat{\theta}(1_w) = \hat{\theta}_1 \).

**Bounding the errors in a Taylor expansion**

We are now in a position to use Assumptions 4.1–4.5 and Condition 4.1 to bound the error terms in a first-order Taylor expansion of \( \hat{\theta}(w) \). We begin by simply calculating the derivative \( d\hat{\theta}(w) / dw \).

**Proposition 4.4.** For any \( w \in W \) for which \( H(\hat{\theta}(w), w) \) is invertible, and for any vector \( a \in \mathbb{R}^N \),
\[
\frac{d\hat{\theta}(w)}{dw} \bigg|_{w=1} a = -H(\hat{\theta}(w), w)^{-1} G(\hat{\theta}(w), a).
\]
CHAPTER 4. INFINITESIMAL JACKKNIFE

Proof. Because \( G(\hat{\theta}(w), w) = 0 \) for all \( w \in W \), by direct calculation,

\[
0 = \frac{d}{dw^T} G(\hat{\theta}(w), w) |_{w=a} \\
= \left( \frac{\partial G}{\partial \hat{\theta}^T} \frac{d\hat{\theta}}{dw^T} + \frac{\partial G}{\partial w^T} \right) |_{w=a} \\
= H(\hat{\theta}(w), w) \frac{d\hat{\theta}}{dw^T} |_{w=a} + \left( \frac{\partial}{\partial w^T} \frac{1}{N} \sum_{n=1}^{N} w_n g_n(\theta) \right) |_{w=a} \\
= H(\hat{\theta}(w), w) \frac{d\hat{\theta}}{dw^T} |_{w=a} + \frac{1}{N} \sum_{n=1}^{N} g_n(\hat{\theta}(w)) a \\
= H(\hat{\theta}(w), w) \frac{d\hat{\theta}}{dw^T} |_{w=a} + G(\hat{\theta}(w), a) .
\]

Because \( H(\hat{\theta}(w), w) \) is invertible by assumption, the result follows. \( \square \)

Definition 4.5. Define

\[
\hat{\theta}_{IJ}(w) := \hat{\theta}_1 + \frac{d\hat{\theta}(w)}{dw^T} |_{w=1} (w - 1) \\
= \hat{\theta}_1 - H^{-1}_1 G(\hat{\theta}_1, w) \quad (\text{because } G(\hat{\theta}_1, 1) = 0) 
\]

\( \hat{\theta}_{II}(w) \) in Definition 4.5 is the first term in a Taylor series expansion of \( \hat{\theta}(w) \) as a function of \( w \). We want to bound the error, \( \hat{\theta}_{II}(w) - \hat{\theta}(w) \).

Theorem 4.5. Under Assumptions 4.4–4.5 and Condition 4.1 when \( \delta \leq \Delta_{\delta} \),

\[
\sup_{w \in W_{\delta}} \left\| \hat{\theta}_{II}(w) - \hat{\theta}(w) \right\|_2 \leq 2C_{op}^2 C_{II} \delta^2.
\]

Proof. By a one-term Taylor series expansion of \( G(\hat{\theta}(w), w) = 0 \) in \( \theta \) around \( \hat{\theta}_1 \), we have, for some \( \bar{\theta} \) such that \( \left\| \bar{\theta} - \hat{\theta}_1 \right\|_2 \leq \left\| \theta(w) - \hat{\theta}_1 \right\|_2 \),

\[
0 = G(\bar{\theta}(w), w) = G(\hat{\theta}_1, w) + H(\bar{\theta}, w) \left( \hat{\theta}(w) - \hat{\theta}_1 \right).
\]

Because \( \delta \in W_{\delta} \), Lemma 4.2 implies that \( \hat{\theta}(w) \in B_{C_{op}\delta} \). Because \( \left\| \hat{\theta} - \hat{\theta}_1 \right\|_2 \leq \left\| \hat{\theta}(w) - \hat{\theta}_1 \right\|_2 \), \( \bar{\theta} \in B_{C_{op}\delta} \) as well. Because \( \bar{\theta} \in B_{C_{op}\delta} \), Lemma 4.5 implies that
$H\left(\hat{\theta}, w\right)$ is invertible, so we can solve for $\hat{\theta}(w) - \hat{\theta}_1$.

$$\hat{\theta}(w) - \hat{\theta}_1 = -H\left(\hat{\theta}, w\right)^{-1}G\left(\hat{\theta}_1, w\right)$$

$$= \left( -H\left(\hat{\theta}, w\right)^{-1} + H\left(\hat{\theta}_1, 1_w\right)^{-1} - H\left(\hat{\theta}_1, 1_w\right)^{-1} \right) G\left(\hat{\theta}_1, w\right)$$

$$= \left( H\left(\hat{\theta}_1, 1_w\right)^{-1} - H\left(\hat{\theta}, w\right)^{-1} \right) G\left(\hat{\theta}_1, w\right) + \hat{\theta}_1(w) - \hat{\theta}_1.$$

Eliminating $\hat{\theta}_1$ and taking the supremum of both sides we have that

$$\sup_{w \in W_\delta} \left\| \hat{\theta}_1(w) - \hat{\theta}(w) \right\|_2$$

$$= \sup_{w \in W_\delta} \left\| \left( H\left(\hat{\theta}_1, 1_w\right)^{-1} - H\left(\hat{\theta}, w\right)^{-1} \right) G\left(\hat{\theta}_1, w\right) \right\|_2$$

$$= \sup_{w \in W_\delta} \left\| H\left(\hat{\theta}, w\right)^{-1} \left( H\left(\tilde{\theta}, w\right) - H\left(\hat{\theta}_1, 1_w\right) \right) H\left(\hat{\theta}_1, 1_w\right)^{-1} G\left(\hat{\theta}_1, w\right) \right\|_2$$

$$\leq 2C_{op} \sup_{w \in W_\delta} \left\| H\left(\tilde{\theta}, w\right) - H\left(\hat{\theta}_1, 1_w\right) \right\|_{op} \left\| H\left(\hat{\theta}_1, 1_w\right)^{-1} G\left(\hat{\theta}_1, w\right) \right\|_2$$

(Lemma 4.5)

$$\leq 2C_{op} \sup_{w \in W_\delta} \left\| H\left(\tilde{\theta}, w\right) - H\left(\hat{\theta}_1, 1_w\right) \right\|_2 \left\| H\left(\hat{\theta}_1, 1_w\right)^{-1} G\left(\hat{\theta}_1, w\right) \right\|_2$$

$$\leq 2C_{op} \sup_{w \in W_\delta} \left\| H\left(\tilde{\theta}, w\right) - H\left(\hat{\theta}_1, 1_w\right) \right\|_2 \left\| H\left(\hat{\theta}_1, 1_w\right)^{-1} G\left(\hat{\theta}_1, w\right) \right\|_2$$

(ordering of matrix norms)

$$\leq 2C_{op}^2 C_{II} \delta \sup_{w \in W_\delta} \left\| H\left(\hat{\theta}_1, 1_w\right)^{-1} G\left(\hat{\theta}_1, w\right) \right\|_2$$

(Lemma 4.4)

$$\leq 2C_{op}^2 C_{II} \delta \sup_{w \in W_\delta} \left\| G\left(\hat{\theta}_1, w\right) \right\|_2$$

(Assumption 4.2)

$$= 2C_{op}^2 C_{II} \delta \sup_{w \in W_\delta} \left\| G\left(\hat{\theta}_1, w\right) - G\left(\hat{\theta}_1, 1_w\right) \right\|_2$$

(because $G\left(\hat{\theta}_1, 1_w\right) = 0$)

$$\leq 2C_{op}^2 C_{II} \delta^2$$ (Condition 4.1).

\[\square\]

Use cases

First, let us state a simple condition under which Assumptions 4.1–4.4 hold. It will help to have a lemma for the Lipschitz continuity.
Lemma 4.7. Derivative Cauchy Schwartz. Let \( a(\theta) = (a_1(\theta), ..., a_N(\theta)) \) be an array of tensors with multi-index \( i \in [D_A] \), and let \( \frac{\partial a(\theta)}{\partial \theta} = \left( \frac{\partial}{\partial \theta} a_1(\theta), ..., \frac{\partial}{\partial \theta} a_N(\theta) \right) \) be an array of tensors of size \( D \times D_A \). Then

\[
\left\| \frac{\partial}{\partial \theta} \| a(\theta) \|_2 \right\|^2 \leq D_A \left\| \frac{\partial a}{\partial \theta} \right\|^2.
\]

Proof. By direct calculation,

\[
\left\| \frac{\partial}{\partial \theta} \| a(\theta) \|_2 \right\|^2 = \sum_{r=1}^{D} \left( \sum_{n=1}^{D_A} \sum_{i=1}^{N} a_{n,i}(\theta)^2 \right)^2
\]

\[
= \sum_{r=1}^{D} \left( \sum_{n=1}^{D_A} \sum_{i=1}^{N} 2a_{n,i}(\theta) \frac{\partial a_{n,i}(\theta)}{\partial \theta_r} \right)^2
\]

\[
\leq \sum_{r=1}^{D} \left( \sum_{n=1}^{D_A} \sum_{i=1}^{N} a_{n,i}(\theta)^2 \right)^{\frac{1}{2}} \left( \sum_{n=1}^{D_A} \left( \frac{\partial a_{n,i}(\theta)}{\partial \theta_r} \right)^2 \right)^{\frac{1}{2}} \right)^2
\]

\[
\leq \sum_{r=1}^{D} \left( 2D_A^2 \left( \frac{1}{D_A} \sum_{i=1}^{D_A} \sum_{n=1}^{N} a_{n,i}(\theta)^2 \right)^{\frac{1}{2}} \left( \frac{1}{D_A} \sum_{n=1}^{D_A} \left( \frac{\partial a_{n,i}(\theta)}{\partial \theta_r} \right)^2 \right) \right)^{\frac{1}{2}} \right)^2
\]

\[
= 4D_A^2 \| a \|^2 \sum_{r=1}^{D} \left\| \frac{\partial a_r}{\partial \theta} \right\|^2
\]

\[
= 4D_A^2 \| a \|^2 \left\| \frac{\partial a}{\partial \theta} \right\|^2.
\]

By the chain rule,

\[
\left\| \frac{\partial}{\partial \theta} \| a(\theta) \|_2 \right\|^2 = \frac{1}{4} \left\| \frac{\partial a}{\partial \theta} \right\|^2 \leq D_A^2 \left\| \frac{\partial a}{\partial \theta} \right\|^2.
\]

\[\square\]

Lemma 4.8. Let \( a(\theta) \in \mathbb{R}^{D \times D} \) be a continuously differentiable random matrix with a \( D \times D \times D \) derivative tensor. (Note that the function, not \( \theta \), is random. For example, \( \mathbb{E}[a(\theta)] \) is still a function of \( \theta \).) Suppose that \( \mathbb{E}[\| a(\theta) \|_2] \) is finite for all
\[ \theta \in \Omega_\theta. \text{ Then, for all } \theta_1, \theta_2 \in \Omega_\theta, \]

\[ |\mathbb{E}[\| a(\theta_1) \|_2] - \mathbb{E}[\| a(\theta_2) \|_2]| \leq \sqrt{\mathbb{E}\left[ \sup_{\theta \in \Omega_\theta} \left\| \frac{\partial a(\theta)}{\partial \theta} \right\|_2^2 \right]} \| \theta_1 - \theta_2 \|_2. \]

**Proof.** For any tensor \( a \) with multi-index \( i \),

\[
\left\| \frac{\partial}{\partial \theta} \| a \|_2 \right\|_2^2 = \sum_{r=1}^D \left( \frac{\partial}{\partial \theta_r} \| a \|_2 \right)^2 \\
= \sum_{r=1}^D \left( \frac{\partial}{\partial \theta_r} \sum_{i=1}^{D_A} a_i^2 \right) \\
= \sum_{r=1}^D \left( 2 \sum_{i=1}^{D_A} a_i \frac{\partial a_i}{\partial \theta_r} \right) \\
\leq 4 \sum_{r=1}^D \sum_{i=1}^{D_A} a_i^2 \sum_{i=1}^{D_A} \left( \frac{\partial a_i}{\partial \theta_r} \right)^2 \quad \text{(Cauchy-Schwartz)} \\
= 4 \sum_{i=1}^{D_A} a_i^2 \sum_{r=1}^D \sum_{i=1}^{D_A} \left( \frac{\partial a_i}{\partial \theta_r} \right)^2 \\
= 4 \| a \|_2^2 \left\| \frac{\partial a}{\partial \theta} \right\|_2^2.
\]

Consequently,

\[
\left\| \frac{\partial}{\partial \theta} \| a(\theta) \|_2 \right\|_2^2 = \left\| \frac{1}{2 \| a(\theta) \|_2} \frac{\partial}{\partial \theta} \| a(\theta) \|_2 \right\|_2^2 \\
= \frac{1}{4 \| a(\theta) \|_2^2} \left\| \frac{\partial}{\partial \theta} \| a(\theta) \|_2 \right\|_2^2 \\
\leq \frac{4 \| a(\theta) \|_2^2}{4 \| a(\theta) \|_2^2} \left\| \frac{\partial}{\partial \theta} a(\theta) \right\|_2^2 \\
= \left\| \frac{\partial a(\theta)}{\partial \theta} \right\|_2^2.
So for any \( \theta_1, \theta_2 \in \Omega \),
\[
\begin{align*}
&|E[\|a(\theta_1)\|_2] - E[\|a(\theta_2)\|_2]| \\
&\leq E[\|a(\theta_1)\|_2^2 - \|a(\theta_2)\|_2^2] \\
&\leq E\left[\left(\sup_{\theta \in \Omega} \left\| \frac{\partial}{\partial \theta} \|a(\theta)\|_2 \right\|_2\right)\right] \|\theta_1 - \theta_2\|_2 \quad (\theta \text{ is not random}) \\
&\leq E\left[\left(\sup_{\theta \in \Omega} \left\| \frac{\partial a(\theta)}{\partial \theta} \right\|_2\right)^2\right] \|\theta_1 - \theta_2\|_2 \\
&\leq \sqrt{E\left[\sup_{\theta \in \Omega} \left\| \frac{\partial a(\theta)}{\partial \theta} \right\|_2^2\right]} \|\theta_1 - \theta_2\|_2.
\end{align*}
\]

The result follows. Note that the bound still holds (though vacuously) if
\[
E\left[\sup_{\theta \in \Omega} \left\| \frac{\partial a(\theta)}{\partial \theta} \right\|_2^2\right]
\]
is infinite. \qed

**Proposition 4.6.** Let \( \Omega \) be a compact set. Let \( g_n(\theta) \) be twice continuously differentiable IID random functions. Define
\[
h_n(\theta) := \frac{\partial g_n(\theta)}{\partial \theta}, \quad r_n(\theta) := \frac{\partial^2 g_n(\theta)}{\partial \theta \partial \theta},
\]
where \( r_n(\theta) \) is a \( D \times D \times D \) tensor. Assume that
\begin{itemize}
  \item 1a) \( E\left[\sup_{\theta \in \Omega} \left\| g_n(\theta) \right\|_2^2\right] < \infty; \)
  \item 1b) \( E\left[\sup_{\theta \in \Omega} \left\| h_n(\theta) \right\|_2^2\right] < \infty; \)
  \item 1c) \( E\left[\sup_{\theta \in \Omega} \left\| r_n(\theta) \right\|_2^2\right] < \infty; \)
  \item 2) \( E[h_n(\theta)] \) is non-singular for all \( \theta \in \Omega; \)
  \item 3) We can exchange expectation and differentiation.
\end{itemize}
Then \( \lim_{N \to \infty} P(\text{Assumptions 4.1}-4.4 \text{ hold}) = 1. \)

**Proof.** The proof follows from Theorems 9.1 and 9.2 of Keener [2011]. We will first show that the expected values of the needed functions satisfy Assumptions 4.1-4.4, and then that the sample versions converge uniformly.

By Jensen’s inequality,
\[
E\left[\sup_{\theta \in \Omega} \left\| g_n(\theta) \right\|_2\right] = E\left[\sqrt{\sup_{\theta \in \Omega} \left\| g_n(\theta) \right\|_2^2}\right] \leq \sqrt{E\left[\sup_{\theta \in \Omega} \left\| g_n(\theta) \right\|_2^2\right]}.
\]
Also, for the \( i \)th component of \( g_n(\theta) \)
\[
\mathbb{E} \left[ \sup_{\theta \in \Omega_\theta} |g_{n,i}(\theta)| \right] \leq \mathbb{E} \left[ \sup_{\theta \in \Omega_\theta} \|g_n(\theta)\|_\infty \right] \leq \mathbb{E} \left[ \sup_{\theta \in \Omega_\theta} \|g_n(\theta)\|_2 \right].
\]

By Theorem 9.1 of Keener [2011], \( \mathbb{E} \left[ \|g_n(\theta)\|_2^2 \right] \), \( \mathbb{E} \left[ \|g_n(\theta)\|_2 \right] \), and \( \mathbb{E} \left[ g_n(\theta) \right] \) are continuous functions of \( \theta \), and because \( \Omega_\theta \) is compact, they are each bounded. Similar reasoning applies to \( h_n(\theta) \) and \( r_n(\theta) \). Consequently we can define
\[
\sup_{\theta \in \Omega_\theta} \mathbb{E} \left[ \|g_n(\theta)\|_2^2 \right] =: Q_g^2 < \infty \]
\[
\sup_{\theta \in \Omega_\theta} \mathbb{E} \left[ \|h_n(\theta)\|_2^2 \right] =: Q_h^2 < \infty.
\]

Below, these constants will be used to satisfy Assumption 4.1 and Assumption 4.3 with high probability.

Because \( \Omega_\theta \) is compact, \( \mathbb{E} \left[ h_n(\theta) \right] \) is continuous, \( \mathbb{E} \left[ h_n(\theta) \right] \) is non-singular, and the operator norm is a continuous function of \( \mathbb{E} \left[ h_n(\theta) \right] \), we can also define
\[
\sup_{\theta \in \Omega_\theta} \|\mathbb{E} \left[ h_n(\theta) \right]^{-1}\|_{op} =: Q_{op} < \infty.
\]

Below, this constant can be used to satisfy Assumption 4.2 with high probability.

Finally, we turn to the Lipschitz condition. Lemma 4.8 implies that
\[
|\mathbb{E} \left[ \|h_n(\theta_1)\|_2 \right] - \mathbb{E} \left[ \|h_n(\theta_2)\|_2 \right]| \leq \sqrt{\mathbb{E} \left[ \sup_{\theta \in \Omega_\theta} \|r_n(\theta)\|_2^2 \right]} \|\theta_1 - \theta_2\|_2.
\]

Define
\[
\Lambda_h = \sqrt{\mathbb{E} \left[ \sup_{\theta \in \Omega_\theta} \|r_n(\theta)\|_2^2 \right]},
\]
so that we have shown that \( \mathbb{E} \left[ \|h_n(\theta)\|_2 \right] \) is Lipschitz in \( \Omega_\theta \) with constant \( \Lambda_h \), which is finite by assumption.

We have now shown, essentially, that the expected versions of the quantities we wish to control satisfy Assumptions 4.1–4.4 with \( N = 1 \). We now need to show that the sample versions satisfy Assumptions 4.1–4.4 with high probability, which will follow from the fact that the sample versions converge uniformly to their expectations by Theorem 9.2 of Keener [2011].
First, observe that Assumption 4.1 holds with probability one by assumption. For the remaining assumption choose an $\epsilon > 0$, and define
\[
\begin{align*}
C_g &:= \sqrt{Q^2_g + \epsilon} \\
C_h &:= \sqrt{Q^2_h + \epsilon} \\
C_{op} &:= 2Q_{op} \\
L_h &:= \sqrt{D^4\Lambda^2_h + \epsilon}.
\end{align*}
\]
By Keener [2011] Theorem 9.2,
\[
\sup_{\theta \in \Omega} \left| \frac{1}{N} \sum_{n=1}^{N} \| g_n (\theta) \|^2_2 - \mathbb{E} \left[ \| g_n (\theta) \|^2_2 \right] \right| \xrightarrow{p} 0.
\]
Because
\[
\sup_{\theta \in \Omega} \left| \frac{1}{N} \sum_{n=1}^{N} \| g_n (\theta) \|^2_2 \right| > Q^2_g + \epsilon \geq \sup_{\theta \in \Theta} \mathbb{E} \left[ \| g_n (\theta) \|^2_2 \right] + \epsilon \Rightarrow \\
\sup_{\theta \in \Theta} \left| \frac{1}{N} \sum_{n=1}^{N} \| g_n (\theta) \|^2_2 - \mathbb{E} \left[ \| g_n (\theta) \|^2_2 \right] \right| > \epsilon,
\]
we have
\[
P \left( \sup_{\theta \in \Theta} \left| \frac{1}{N} \sum_{n=1}^{N} \| g_n (\theta) \|^2_2 \right| \geq Q^2_g + \epsilon \right) \leq \\
P \left( \sup_{\theta \in \Theta} \left| \frac{1}{N} \sum_{n=1}^{N} \| g_n (\theta) \|^2_2 - \mathbb{E} \left[ \| g_n (\theta) \|^2_2 \right] \right| \leq \epsilon \right),
\]
so
\[
P \left( \sup_{\theta \in \Theta} \left| \frac{1}{N} \sum_{n=1}^{N} \| g_n (\theta) \|^2_2 \right| \geq C^2_g \right) \xrightarrow{N \to \infty} 0.
\]
An analogous argument holds for $\frac{1}{N} \| h_n (\theta) \|^2_2$. Consequently,
\[
P (\text{Assumption 4.3 holds}) \xrightarrow{N \to \infty} 1.
\]
We now consider Assumption 4.2. Again, by Keener [2011] Theorem 9.2 applied to each element of the matrix $h_n(\theta)$, using a union bound over each of the $D^2$ entries,

$$\sup_{\theta \in \Omega} \left\| \frac{1}{N} \sum_{n=1}^{N} h_n(\theta) - \mathbb{E}[h_n(\theta)] \right\|_1 \xrightarrow{p \to 0} 0.$$  

By the converse of Proposition 4.3 because $\|\mathbb{E}[h_n(\theta)]^{-1}\|_{op} \leq Q_{op}$,

$$\left\| \left( \frac{1}{N} \sum_{n=1}^{N} h_n(\theta) \right)^{-1} \right\|_{op} \geq 2Q_{op} = C_{op} \Rightarrow$$

$$\left\| \frac{1}{N} \sum_{n=1}^{N} h_n(\theta) - \mathbb{E}[h_n(\theta)] \right\|_1 > \frac{1}{2} Q_{op}^{-1}.$$  

Consequently,

$$P\left( \left\| \left( \frac{1}{N} \sum_{n=1}^{N} h_n(\theta) \right)^{-1} \right\|_{op} \geq C_{op} \right) \leq$$

$$P\left( \left\| \frac{1}{N} \sum_{n=1}^{N} h_n(\theta) - \mathbb{E}[h_n(\theta)] \right\|_1 \xrightarrow{p \to 0} 0, \right.$$  

and $P$ (Assumption 4.2 holds) $\xrightarrow{N \to \infty} 1$.

Finally, applying Lemma 4.8 to $\frac{1}{\sqrt{N}} \|h(\theta_2)\|_2$,

$$\left| \frac{1}{\sqrt{N}} \|h(\theta_1)\|_2 - \frac{1}{\sqrt{N}} \|h(\theta_2)\|_2 \right| \leq \sup_{\theta \in \Omega_{\theta}} \left\| \frac{\partial}{\partial \theta} \frac{1}{\sqrt{N}} \|h(\theta)\|_2 \right\|_2 \|\theta_1 - \theta_2\|_2$$

$$\leq \frac{D^2}{\sqrt{N}} \sup_{\theta \in \Omega_{\theta}} \|r(\theta)\|_2 \|\theta_1 - \theta_2\|_2$$

$$= D^2 \sqrt{\sup_{\theta \in \Omega_{\theta}} \frac{1}{N} \|r(\theta)\|_2^2 \|\theta_1 - \theta_2\|_2}.$$  

Consequently,
\[
\left| \frac{1}{\sqrt{N}} \| h(\theta_1) \|_2 - \frac{1}{\sqrt{N}} \| h(\theta_2) \|_2 \right| \geq L_h \| \theta_1 - \theta_2 \|_2 \implies \\
D^2 \sqrt{\sup_{\theta \in \Theta} \frac{1}{N} \| r(\theta) \|_2^2} \geq L_h \Rightarrow \\
\sup_{\theta \in \Theta} \frac{1}{N} \| r(\theta) \|_2^2 - \sup_{\theta \in \Theta} \mathbb{E} \left[ \| r_n(\theta) \|_2^2 \right] \geq \frac{L_h^2}{D^4} - \sup_{\theta \in \Theta} \mathbb{E} \left[ \| r_n(\theta) \|_2^2 \right] \Rightarrow \\
\sup_{\theta \in \Theta} \left( \frac{1}{N} \| r(\theta) \|_2^2 - \mathbb{E} \left[ \| r_n(\theta) \|_2^2 \right] \right) \geq \frac{L_h^2}{D^4} - \Lambda_h^2 = \epsilon.
\]

However, again by Keener [2011] Theorem 9.2,
\[
\sup_{\theta \in \Theta} \left( \frac{1}{N} \| r(\theta) \|_2^2 - \mathbb{E} \left[ \| r_n(\theta) \|_2^2 \right] \right) \xrightarrow{p \to 0, N \to \infty} 0,
\]
so \( P \) (Assumption 4.4 holds) \( \xrightarrow{N \to \infty} 1 \).
Chapter 5

Approximate measures of cluster stability for Bayesian nonparametrics

Clustering is the canonical unsupervised learning problem, in which we aim to find an assignment of data points to groups, or clusters, that represent meaningful latent structure in a data set. Bayesian nonparametric (BNP) models form a particularly popular set of Bayesian models for clustering due to their flexibility and coherent assessment of uncertainty. As with any Bayesian model of moderate complexity, typically the Bayesian posterior cannot be computed exactly for BNP clustering problems, and an approximation must be employed. Mean-field variational Bayes (MFVB) forms a posterior approximation by solving an optimization problem and is widely used due to its speed [Blei and Jordan, 2006]. An exact BNP posterior might, at least in theory, vary dramatically when presented with different data. Certainly we expect small, rare clusters—which are ubiquitous in BNP—to vary substantially based on the observed data. When reporting the summaries of the clustering for the purposes of scientific inquiry, it behooves us to understand how stable, or alternatively how sensitive, this report is relative to the data [Yu, 2013].

If one were to use the bootstrap to assess stability in this analysis pipeline, it would require a new run of MFVB for each simulated data set. This time cost is often prohibitively expensive, especially for exploratory data analyses. We instead propose to provide a fast, automatic approximation to a full bootstrap analysis based on the infinitesimal jackknife [Jaeckel, 1972; Efron, 1982], which can be seen as a linear approximation to the global stability measure provided by the full bootstrap. This locality can buy drastic time savings, with the infinitesimal jackknife sometimes running orders of magnitude faster than the bootstrap. We here demonstrate how to apply this idea to a data analysis pipeline consisting of an MFVB approxi-
CHAPTER 5. CLUSTER STABILITY

mation to a BNP clustering posterior. We show that the necessary calculations can be done nearly automatically, without tedious derivations by a practitioner, using modern automatic differentiation software [Maclaurin et al., 2015]. This automa-
tion suggests a generality to our methods beyond BNP clustering.

In the remainder, we describe the BNP model and MFVB inference in more
detail in Section 5.1. We review summaries for assessing the output of our cluster-
ing, across which we can in turn assess stability, in Section 5.5. We describe our
new stability assessment procedure in Section 5.6. And we demonstrate our abil-
ty to quickly and accurately quantify stability in Section 5.7 on an application to
clustering time-course gene expression data [Shoemaker et al., 2015, Luan and Li,
2003].

5.1 Data, model, and inference

Clustering procedures typically estimate which data points are clustered together,
a quantity of primary importance in many analyses. It can be used to reduce the
dimensionality, or to facilitate the interpretation of complex data sets. For exam-
ple, genomics experiment often assess cell activity genome-wide, but many genes
behave the same way. Clustering them thus allows dimensionality reduction that
can facilitate interpretation. Finding robust and stable clusters is thus crucial for
appropriate downstream analysis.

Because the differences and evolution over time of gene expression yields im-
portant insight on gene regulation of the cell-cycle, or on how cells react to toxins,
drugs or viruses, we focus on the specific task of clustering time course gene-
expression data. We use a publicly available data set of mice gene expression
[Shoemaker et al., 2015], composed 14 time points after mice are infected with
the influenza virus. See Section 5.2 for more details.

The observed data consists of expression levels $y_{gt}$ for genes $g = 1, ..., n_g$ and
time points $t = 1, ..., n_t$ (see Fig. (5.1) for a single-gene time course data). As
described by Luan and Li [2003], we model the time series as a gene-specific con-
stant additive offset plus a B-spline basis of degree 3 and 7 degrees of freedom. We
denote the basis matrix by $X$ (see Fig. (5.1) in Section 5.2).

Let $b_g$ denote the additive offset for gene $g$, and $y_g$ the vector of observations
$(y_{g1}, ..., y_{gT})^T$. Denote the variance of the errors as $\sigma^2$ and let $I_T$ be the $n_t \times n_t$
identity matrix. We model each gene’s B-spline coefficients, $\beta_g$, using a a stick-
breaking representation of a Bayesian nonparametric (BNP) Dirichlet process mix-
ture model [Ferguson, 1973, Sethuraman, 1994]. Excluding priors, the generative
model is then:
\[ \nu_k | \alpha \sim \text{Beta}(1, \alpha) \]
\[ \pi_k(\nu) := \nu_k \prod_{j=1}^{k-1} (1 - \nu_j) \]
\[ \beta_k \sim \mathcal{G}_0 \]
\[ z_g|\nu \sim \text{Mult}(\pi(\nu)) \]
\[ \beta_g|z_g = \sum_{k=1}^{\infty} \beta_k z_{gk} \]
\[ y_g|X, \beta_g, b_g, \sigma^2 \sim \mathcal{N}(X \beta_g + b_g, I_T \sigma^2) . \] (5.1)

See Section 5.3 for details of the priors.

For brevity, use the single vector \( \theta \) to represent all the unknown parameters \( \nu, \beta_k, z_{gk}, \sigma^2, \) and \( b_g \), for all \( k \) and \( g = 1, \ldots, n_g \). We are interested in the posterior \( p(\theta|y) \), which is intractable. To approximate \( p(\theta|y) \), we form a variational approximation to \( p(\theta|y) \), denoted \( q^*(\theta) \) and parameterized by a real-valued parameter \( \eta \), using a truncated representation of the BNP prior with \( K = 30 \) components, which was large enough that more than half of the clusters were essentially unoccupied \[\text{[Blei and Jordan, 2006].}\] The variational distribution is chosen as a local minimum of the KL divergence from the true posterior:

\[ q^*(\theta) := q(\theta|\eta^*) \text{ where } \eta^* := \arg\min KL(q(\theta|\eta)||p(\theta|Y)) . \] (5.2)

See Section 5.3 for details of the variational approximation. Ideally, we would like a global minimum of Eq. (5.2), but due to the non-convexity of the problem, we can only guarantee finding a local minimum. Importantly for the assessment of co-clustering, knowledge of \( \eta^* \) allows us to approximate the posterior probability \( \zeta_{gk}(\eta^*) := E_{q^*}[\mathbb{I}_{z_{gk}}] \), the posterior probability of gene \( g \) belonging to cluster \( k \). We write \( \zeta \) without subscripts to refer to the \( n_g \times K \) matrix with entries \( \zeta_{gk} \).

Finally, we introduce some additional notation related to data sensitivity that will be useful to describe the bootstrap and the infinitesimal jackknife in Section 5.6. To assess data sensitivity, we augment our model with scalar per-gene weights, \( w_g \geq 0 \), where \( W = (w_1, \ldots, w_{n_g})^\top \), where we define the weighted likelihood and corresponding optimal variational parameter:

\[ \log p(Y|\theta, W) = \sum_{g=1}^{n_g} w_g \log p(y_{g} | \theta) \Rightarrow \]
\[ \eta^*(W) := \arg\min KL(q(\theta; \eta)||p(\theta|Y, W)) . \] (5.3)
Defining $W_1 := (1, ..., 1)^\top$ we recover the original variational posterior $\eta^* = \eta^*(W_1)$. By setting $W$ to other integer-valued vectors, we can produce the effect of removing or repeating datapoints, since $p(Y|\theta)$ is exchangeable in $y_g$. In particular, by drawing $n_b$ bootstrap weights $W_b \sim \text{Multinomial}(n_b, n_b^{-1})$, for $b = 1, ..., n_b$, the bootstrap distribution of a function $\phi(\zeta(\eta^*))$ can be approximated with the draws $\phi(\zeta(\eta^*(W)))$. In the remainder of the paper, in a slight abuse of notation, we will write $\phi(W)$ in place of $\phi(\zeta(\eta^*(W)))$ below when the meaning is clear from the context.

5.2 Data description and processing

We use the publicly available mice micro array data set [Shoemaker et al., 2015]. Mice were infected with different influenza viruses, and gene expression was assessed at 14 time points after infection. We focus on the influenza virus “A/California/04/2009,” a mildly pathogenic virus from the 2009 pandemic season. We normalize the data as described in [Shoemaker et al., 2015]. We then apply the differential analysis tool EDGE between the influenza infected mice and control mice [Storey et al., 2005]. EDGE yields for each gene a p-value assessing how differently the genes behave between the two conditions. We then rank the genes from most significantly differentially expressed, to least significantly expressed and perform all downstream analysis on the top 1000 genes.

The observations are unevenly spaced, with more frequent observations at the beginning. As shown Fig. (5.1), each gene also has multiple measurements at each time point (called biological replicates). By modeling gene expression as a smooth function, via a B-spline basis, we naturally model the time aspect of the data, as well as provide an easy framework for including biological replicates in the clustering. The reader may observe that the sparse observations at later times leads to apparent non-smoothness in the fitted time series at late times, though the B-splines enforce smoothness in actual calendar time as desired.

![Figure 5.1: Data and splines](image-url)
5.3 Variational inference

We used the following priors:

\[ \alpha = 2 \]

\[ \beta_{ki} \overset{iid}{\sim} \mathcal{N}(0.38, 0.10^{-1}) \]

\[ b_g \overset{iid}{\sim} \mathcal{N}(0, 0.10^{-1}) \]

\[ \tau := \sigma^{-2} \overset{iid}{\sim} \text{Gamma}(0.10, 10.00) \]

The variational approximation was

\[ q(\theta|\eta) = \delta(\beta) \delta(\tau) \prod_{k=1}^{K} \left\{ q(\nu_k) \prod_{g} q(z_{gk}) q(b_g|z_{gk} = 1) \right\} \]

where \( \delta(\cdot) \) denotes a point mass at a parameterized location \cite{Neal:1998}, \( q(\nu_k) \) is a beta distribution, \( q(z_{gk}) \) is a multinomial distribution, \( q(b_g|z_{gk} = 1) \) is a normal distribution, and \( \eta \) denotes the vector of parameters for all these distributions. With this approximation, we seek \( \eta^*: \arg\min[\eta] KL(q(\theta|\eta)||p(\theta|Y)) \). See Appendix 5.4 for details of the optimization.

5.4 Optimization

Note that by parameterizing \( q(b_g, z_g) = \prod_{k=1}^{K} q(b_g|z_{gk} = 1) q(z_g) \), the updates for \( q(b_g, z_g) \) have a closed form given \( q(\beta, \tau, \nu) \). Denote the parameters for \( q(b_g, z_g) \) as \( \eta_{local} \) and the parameters for \( q(\beta, \tau, \nu) \) as \( \eta_{global} \), and write

\[ \hat{\eta}_{local}(\eta_{global}) := \arg\min[\eta_{local}] KL(\eta_{global}, \eta_{local}) \]

we can write the optimization problem Eq. (5.2) as a function of \( \eta_{global} \) only:

\[ \eta_{global} = \arg\min[\eta_{global}] KL(\eta_{global}, \hat{\eta}_{local}(\eta_{global})) \]  \hspace{1cm} (5.4)

This is valuable because the size of \( \eta_{local} \) grows with the number of genes, but the size of \( \eta_{global} \) does not. In addition to speeding up optimization, Eq. (5.4) can be easily differentiated using autograd to calculate the sensitivity matrix \( S \) in Eq. (5.7).

\footnote{Technically, a true point mass does not have a well-defined KL divergence with respect to the Lebesgue measure on \( \beta \) and \( \tau \). But \( \delta(\beta; \eta_{\beta}) \) can be thought of as a density with constant entropy, and where \( E_{\delta}[\beta] \approx \eta_{\beta} \). Such a distribution can be approximated arbitrarily closely with a multivariate normal distribution with vanishing variance, for example.}
To solve Eq. (5.4), we use a combination of Newton and quasi-Newton methods. We first choose an initialization by fitting individual B-splines to each gene expression, and use K-means to cluster the coefficients; the centroids were used to initialize the variational means for $\beta_k$. From this initialization, we ran BFGS for 300 iterations; at the point where BFGS terminated, we computed the Hessian of the KL objective, Eq. (5.4). This Hessian was used as a preconditioner for the final Newton trust region steps, which was iterated to convergence. Hessians were computed using autograd [Maclaurin et al., 2015], while BFGS and the newton trust-region routines were done with the BFGS and trust-ncg methods of scipy-optimize [Jones et al., 2001], respectively.

5.5 Clustering stability measures

To quantify the stability of a clustering procedure, we must first define measures of similarity between different clustering outputs. In particular, we will consider the similarity between the clustering $\zeta = \zeta(W_i)$, which is clustering at the optimum $\eta^*$, and $\tilde{\zeta} := \zeta(W_b)$ at bootstrap weights $W_b$.

In this work, we focus on two standard clustering stability measures: the Fowlkes-Mallows index and the Normalized mutual information. Both yield scores ranging between 0 and 1, where the higher the scores, the more similar the clusterings are. Note that while we focus on these measures, the procedure described below can be applied to any similarity measures $\phi(\tilde{\zeta})$. As mentioned above, we adapt the stability measures to be a function of $\zeta$ and $\tilde{\zeta}$. We here describe in more details those similarity measures and how we adapted them for our use case.

First, we will take a closer look at the Fowlkes-Mallows index [Fowlkes and Mallows, 1983]. Ignoring for the moment the variational distribution, consider a general clustering algorithm that outputs binary indicators $z_{gk}$ for gene $g$ belonging to cluster $k$. Suppose two different runs of the algorithm (e.g. runs with two different initializations) give two different outputs $z_{gk}$ and $\tilde{z}_{gk}$. Then the Fowlkes-Mallows similarity index is defined as

$$FM = \frac{\sum_{g_1g_2} C_{g_1g_2} \tilde{C}_{g_1g_2}}{\sqrt{(\sum_{g_1g_2} C_{g_1g_2}^2) \cdot (\sum_{g_1g_2} \tilde{C}_{g_1g_2}^2)}}$$  \hspace{1cm} (5.5)

where $C_{g_1g_2} := \sum_{k=1}^{K} z_{g_1k} z_{g_2k}$ is the indicator that genes $g_1$ and $g_2$ are clustered together under the first clustering; and $\tilde{C}_{g_1g_2}$ denotes the same quantity under the second clustering. The numerator in Eq. (5.5) then counts the number of gene
pairs that were co-clustered by both two clustering results, and the denominator normalizes the index to be between 0 and 1; hence, values closer to 1 suggest a more similar clustering.

We modify this definition slightly for our case since we have more than just binary indicators: we have posterior probabilities for $z_{gk}$ approximated by the variational distribution. This then gives the probability of co-clustering under the variational distribution, $E_{q^*} [C_{g1g2}]$. Having two different clustering results now corresponds to having two different variational distributions for $z$. To measure clustering similarity here, we simply replace $C_{g1g2}$ and $\tilde{C}_{g1g2}$ in Eq. (5.5) with $E_{q^*} [C_{g1g2}]$ and $E_{\tilde{q}^*} [C_{g1g2}]$, their expectations under two different variational distributions.

Now, let’s turn to the normalized mutual information score. Let $q$ and $\tilde{q}$ be two different variational distributions, with $E_{q^*} [z_{gk}] := \zeta_{gk}$ and $E_{\tilde{q}^*} [z_{gk}] := \tilde{\zeta}_{gkl}$. Suppose we consider the distribution on labels induced by drawing a random gene $g$, and then drawing the labels $k_1 \mid g \sim q(z_g)$ and $k_2 \mid z_g \sim \tilde{q}(z_g)$. Then define $P(k_1) = \frac{1}{n_g} \sum_g \zeta_{gk}$, the probability of cluster $k_1$ under the first variational distribution, and $\tilde{P}(k_2) = \frac{1}{n_g} \sum_g \tilde{\zeta}_{gk}$, the probability of cluster $k_2$ under the second variational distribution; also let $P(k_1,k_2) = \frac{1}{n_g} \sum_g \zeta_{gk_1}\tilde{\zeta}_{gk_2}$, the joint cluster probabilities. Then the normalized mutual information score for clustering similarity is given by

$$NMI = \frac{\sum_{k_1k_2} P(k_1,k_2) \log\left(\frac{P(k_1,k_2)}{P(k_1)P(k_2)}\right)}{\sqrt{\left(\sum_k P(k) \log P(k)\right) \cdot \left(\sum_k \tilde{P}(k) \log \tilde{P}(k)\right)}} \quad (5.6)$$

The numerator is the mutual information between the two clustering outputs defined by the variational distributions $q$ and $\tilde{q}$, with a larger mutual information representing more similar clusterings; the denominator then normalizes such that the score is between 0 and 1.

### 5.6 Data sensitivity

We now derive a local approximation to the bootstrap using the weight notation from Section 5.1. Noting that Eq. (5.3) is well-defined even for non-integer values of $W$, and observing that $KL(q(\theta; \eta) || p(\theta | Y, W))$ is smooth in both $\eta$ and $W$, it follows that $\eta^*(W)$ is smooth in $W$ in a neighborhood of $W_1$. Using the results from Giordano et al. [2018, Appendix D], and adopting the shorthand notation $KL(\eta, W) := KL(q(\theta; \eta) || p(\theta | Y, W))$, we can then calculate a “weight
CHAPTER 5. CLUSTER STABILITY

sensitivity matrix” \( S \) as

\[
S := \left. \frac{d\eta^* (W)}{dW} \right|_{W=W_1} = - \left. \left( \frac{\partial^2 KL (\eta, W)}{\partial \eta \partial \eta} \right)^{-1} \frac{\partial^2 KL (\eta, W)}{\partial \eta \partial W} \right|_{W=W_1}.
\] (5.7)

Although Eq. (5.7) would be tedious to calculate by hand, it can be calculated exactly using automatic differentiation in just a few lines of code (see Section 5.4 for more details).

Using \( S \), and a single-term Taylor expansion, we can approximate \( \eta^* (W) \) and, in turn, a clustering metric \( \phi (W) \):

\[
\eta^* (W) \approx \eta^*_{\text{Lin}} (W) := \eta^* + S (W - W_1) \quad (5.8)
\]

\[
\phi (W) = \phi (\zeta (\eta^* (W))) \approx \phi_{\text{Lin}} (W) := \phi (\zeta (\eta^*_{\text{Lin}} (W))). \quad (5.9)
\]

Note that the quantities \( \zeta \), which are probabilities and must lie between 0 and 1, can be expected to be extremely non-linear functions of \( \eta^* \), but they can be calculated quickly for any given \( \eta^* \). We take advantage of this fact to make a linear approximation only on \( \eta^* \) rather than calculating \( \frac{d\phi}{dW} \) directly.

This is nearly equivalent to the “infinitesimal jackknife” of Jaeckel [1972] (see also Efron [1982, Chapter 6]), where \( \eta^* \) is thought of as a statistic depending on the data \( y_g \). The only difference is that we linearize \( \eta^* \) rather than the full statistic \( \phi \). In order to avoid confusion with the jackknife estimator of variance, we will refer to \( \phi_{\text{Lin}} (W_b) \) as the “linear bootstrap” in Section 5.7 below. Note that the right-hand side of Eq. (5.8) involves only a matrix multiplication once \( S \) has been calculated, but evaluating \( \eta^* (W) \) exactly for \( W \neq W_1 \) typically involves re-solving the optimization problem Eq. (5.3). So although \( \phi_{\text{Lin}} (W_b) \) is only an approximation, it can generally be calculated much more quickly than \( \phi (W) \) (as is shown in Table 5.1 on page 122 below).

5.7 Results

We optimized Eq. (5.3) in Python using the trust-ncg method of scipy-optimize [Jones et al., 2001] using an initialization based on K-means. We calculated the necessary derivatives for the optimization and for Eq. (5.7) using the automatic differentiation library autograd [Maclaurin et al., 2015]. See Section 5.4 for details.

We first found a high-quality optimum for the original dataset (that is, at \( W = W_1 \)) by choosing lowest KL divergence achieved amongst 200 random restarts.\(^2\)

\(^2\)Our results are not quite as good if we take \( \eta^* \) to be an optimum chosen after only 10 rather than 200 initializations – see Section 5.8 for more details and discussion.
We take this optimum to be $\eta^*$, the value at which we calculate the sensitivity $S$ in Eq. (5.7). Then, for $n_b = 200$ different bootstrap weights $W_b$, we calculate three different estimates of $\eta^* (W_b)$: “warm starts,” $\eta_{Warm}^* (W_b)$, which optimize $\eta^* (W_b)$ starting at $\eta^*$; “cold starts,” $\eta_{Cold}^* (W_b)$ which optimize $\eta^* (W_b)$ taking the best of ten new random K-means initializations, and the linear bootstrap estimates, which are $\eta_{Lin}^* (W_b)$ of Eq. (5.8). For each of these three optima, we compare the bootstrap distribution of the stability measures of Section 5.5. The median times to calculate each of these measures are given in Table 5.1 on page 122.

Table 5.1: Median times (in seconds) to compute each bootstrap sample (or related quantities)

<table>
<thead>
<tr>
<th>$\eta^*$ (200 inits)</th>
<th>$\eta_{Cold}^*$ (10 inits)</th>
<th>$\eta_{Warm}^*$ (1 init)</th>
<th>$\eta_{Lin}^*$ (given $S$)</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time:</td>
<td>16088</td>
<td>931</td>
<td>53</td>
<td>0.0003</td>
</tr>
</tbody>
</table>

Fig. (5.1) shows the distribution of $\phi_{MI} (W_b)$ and $\phi_{FM} (W_b)$ for the three measures. Although the bootstrap based on is $\eta_{Lin}^* (W_b)$ biased slightly upwards relative to both of the actual bootstraps, it is a good approximation to the warm start bootstrap.

Figure 5.2: Cluster quality

Finally, we look at the bootstrap standard deviation of the elements of the matrix $\zeta (W_b)$. Fig. (5.3) shows the relationship between the co-clustering standard deviation as measured by $\eta_{Warm}^* (W_b)$ on the x-axis and $\eta_{Lin}^* (W_b)$ or $\eta_{Cold}^* (W_b)$ on the y-axes. Each point in the graph corresponds to a single value of $W_b$, so each graph contains $B = 200$ points. Because the vast majority pairs have very small standard deviation in both measures of the graph, we condition on at least one standard deviation being larger than 0.03. For both the cold start and the linear bootstrap, most of the mass lies on the diagonal, indicating a good qualitative correspondence with the warm start, though there is more frequent extreme deviation in the linear bootstrap.
5.8 Local optima

Many unsupervised clustering problems exhibit multiple local optima in the objective function, even for permutation-invariant quantities like co-clustering measures, and the problem described in the present work is no exception. Measures of uncertainty which are based on local information (like the infinitesimal jackknife) cannot be expected to capture the frequentist variability due to different initializations leading to substantively different local optima. The fact that the cold starts have lower-quality co-clustering than the warm starts in Fig. (5.2) indicates that there exist different local optima relatively far from $\eta^*$. In this section, we briefly discuss two additional observations concerning local optima.

One might first ask whether the local optima found by the cold start are much worse than those found by the warm start. The distribution of KL divergences across the bootstrap samples is shown in Fig. (5.4). Each point in Fig. (5.4) corresponds to two different estimates at the same weights $W_b$, so there are $B = 200$ points in each graph. The linear response KL divergence, which is not evaluated at an actual optimum, is larger than the corresponding optimal value, as expected. Note that the cold start KL divergence is not actually noticeably worse than the warm start KL divergence, suggesting that there may be meaningful frequentist variability due to local optima that is not captured by either $\eta_{\text{Warm}}(W_b)$ nor $\eta_{\text{Lin}}(W_b)$.

Finally, we note that the results in Section 5.7 depend in part on the fact that we are re-starting the optimization in our bootstrap samples at a high-quality optimum, $\eta^*$, chosen as the best out of 200 random restarts. If, instead, we set $\eta^*$ to be the best optimum found after only 10 random restarts, the results are not quite as good, as seen in Fig. (5.5). This is probably due both to the base set of cluster assignments, $\zeta$ in Eq. (5.5), is not as high-quality an optimum, and to the fact that optima near $\eta^*$, being of lower quality, is chosen less often during the bootstrap procedure.
Figure 5.4: Distribution of KL divergence relative to the warm start

Figure 5.5: Results with an initial optimum based on only 10 random restarts rather than 200
Chapter 6

Evaluating sensitivity to the stick breaking prior in Bayesian nonparametrics

A central question in many probabilistic clustering problems is how many distinct clusters are present in a particular dataset. A Bayesian nonparametric (BNP) model addresses this question by placing a generative process on cluster assignment, making the number of distinct clusters present amenable to Bayesian inference. However, like all Bayesian approaches, BNP requires the specification of a prior, and this prior may favor a greater or lesser number of distinct clusters. In practice, it is important to quantitatively establish that the prior is not too informative, particularly when—as is often the case in BNP—the particular form of the prior is chosen for mathematical convenience rather than because of a considered subjective belief.

We derive local sensitivity measures for a truncated variational Bayes (VB) approximation based on the Kullback-Leibler (KL) divergence. Local sensitivity measures approximate the nonlinear dependence of a VB optimum on prior parameters using a local Taylor series approximation [Gustafson, 1996b; Giordano et al., 2018]. Using a stick-breaking representation of a Dirichlet process, we consider perturbations both to the scalar concentration parameter and to the functional form of the stick-breaking distribution. As far as the authors are aware, ours is the first analysis of the local sensitivity of BNP posteriors when using a VB approximation.

Unlike previous work on local Bayesian sensitivity for BNP [Basu, 2000], we pay special attention to the ability of our sensitivity measures to extrapolate to different priors, rather than treating the sensitivity as a measure of robustness per se. Extrapolation motivates the use of multiplicative perturbations to the functional form of the prior for VB, as the KL divergence is then linear in the perturbation. Additionally, we linearly approximate only the optimization of the global parameters.
In VB, this step is typically the most computationally intensive part of inference. Meanwhile, we retain the non-linearities of easily computed quantities as functions of the global parameters, such as local parameters and the number of distinct clusters.

We apply our methods to estimate sensitivity of the expected number of distinct clusters present the Iris dataset [Anderson, 1936, Fisher, 1936] to the BNP prior specification. We evaluate the accuracy of our approximations by comparing to the much more expensive process of re-fitting the model.

6.1 Model and inference

Data and model. We use the Iris dataset [Anderson, 1936, Fisher, 1936], which contains 150 observations of three different types of iris flowers. We use measurements of their sepal length, sepal width, petal length, and petal width to cluster the data with the goal of recovering the three species. Let \( y_n \in \mathbb{R}^4 \) be these four measurements for flower \( n \).

In the spirit of BNP, let us suppose that there there are an infinite number of distinct species of iris in the world, indexed by \( k = 1, 2, 3, \ldots \), only some finite number of which are present in our observed dataset. Let \( z_n \) denote the index of the species (i.e. the cluster) to which flower \( n \) belongs, i.e., \( z_n = k \) for exactly one \( k \). Each cluster has mean \( \mu_k \in \mathbb{R}^4 \) and covariance \( \Sigma_k \in \mathbb{R}^{4 \times 4} \), and we write the collections as \( \mu = (\mu_1, \mu_2, \ldots) \) and \( \Sigma = (\Sigma_1, \Sigma_2, \ldots) \). Our data-generating process given the model parameters is then

\[
y_n | z_n, \mu, \Sigma \sim \mathcal{N} \left( y_n \bigg| \sum_{k=1}^{\infty} I\{z_n = k\} \mu_k, \sum_{k=1}^{\infty} I\{z_n = k\} \Sigma_k \right), \quad n = 1, \ldots, N.
\]

For \( \mu \) and \( \Sigma \), we use dispersed IID conjugate priors. For the prior on the cluster memberships \( z_n \), we use a stick-breaking representation of a BNP Dirichlet process prior [McCloskey, 1965, Ferguson, 1973, Patil and Taillie, 1977, Sethuraman, 1994]. Specifically, we define latent stick-breaking proportions \( \nu = (\nu_1, \nu_2, \ldots) \), a concentration parameter \( \alpha > 0 \), and base stick-breaking distribution \( p_0(\nu_k | \alpha) = \text{Beta} \left( \nu_k | 1, \alpha \right) \). The prior on the cluster assignments \( z_n \) for \( n = 1, \ldots, 150 \) is then
given by

\[ \nu | \alpha \sim \prod_{k=1}^{\infty} p_0(\nu_k | \alpha), \quad \text{with} \]

\[ \pi_k | \nu := \nu_k \prod_{j=1}^{k-1} (1 - \nu_j) \quad \text{and} \]

\[ z_n \sim \text{Categorical}(\pi). \quad (6.1) \]

The concentration parameter \( \alpha \) and stick-breaking prior \( p_0 \) thus determine our prior belief about the number of clusters present. This prior specification and the observed data combine to inform our posterior belief about the posterior number of clusters. We will be examining the sensitivity of our posterior belief to our choices for \( \alpha \) and \( p_0 \).

**Variational approximation.** It is difficult to calculate the posterior \( p(\nu, \mu, \Sigma, z | y) \), both because the normalizing constant is intractable and because there are an infinite number of latent components in a true BNP representation. In order to perform approximate inference, we use a truncated VB approximation using \( K = 30 \) components \[\text{[Blei and Jordan, 2006]}\] and assert that the later components are essentially unoccupied in the posterior. For compactness of notation, let \( \theta = (\nu, \mu, \Sigma) \) denote the collection of “global” parameters, i.e., parameters whose values affect the data-generating process of every observation \( y_n \). Let \( \delta(\cdot) \) denote a delta function. We define a class of approximating distributions for VB as

\[ Q := \left\{ q : q(\theta, z) = \left( \prod_{k=1}^{K} q(\nu_k) \delta(\mu_k) \delta(\Sigma_k) \right) \left( \prod_{n=1}^{150} q(z_n) \right), \right. \]

where

\[ q(\nu_k) = \text{Lognormal}(\nu_k) \quad \text{and} \quad q(z_n) = \text{Categorical}(z_n). \]

The family \( Q \) is parameterized by a finite-dimensional vector containing (1) the locations of the delta functions and (2) the parameters for the lognormal distributions, \( \eta_\theta \), and the parameters for the categorical distributions, which we denote by \( \eta_z \). We write the combined parameters as \( \eta = (\eta_\theta, \eta_z) \). That is, \( \eta \) is defined such that \( Q = \{ q : q(\theta, z) = q(\theta, z | \eta) = q(\theta | \eta_\theta) q(z | \eta_z) \} \). The variational approximation is then given by \( \eta^* = \arg\min_{\eta} KL(q(\theta, z | \eta) \| p(\theta, z | y)) \).

1^We use the lognormal distribution rather than the conjugate beta distribution because the log-normal makes numerical integration easier when re-optimizing using non-conjugate \( p_0 \). Were one to rely on our sensitivity measures and not re-optimize, there would be no need for numerical integration, and the more convenient beta variational approximation could be used.
It will be important later to note that it is easy to calculate the optimal \( \eta \) for a given \( \eta_\theta \) because the model is conditionally conjugate, i.e., \( p(z|\theta, y) \) is categorical, and so is \( q(z|\eta_\theta) \). Specifically, there exists an easily-calculated, closed form for \( \eta_\theta^* (\eta_\theta) = \arg\min_{\eta_\theta} KL (q(\theta|\eta_\theta)q(z|\eta_\theta)||p(\theta, z|y)) \).

**Target posterior quantity.** We are interested in the inferred number of clusters present in the observed data. This quantity can be expressed as an expectation with respect to \( q(z|\eta_\theta) \), and therefore as a function of \( \eta_\theta^* \) via the relation \( \eta_\theta^* (\eta_\theta) \):

\[
\begin{align*}
g(\eta_\theta^*) & := \mathbb{E}_{q(\theta, z|\eta_\theta^*)} [\# \{ \text{distinct clusters} \}] \\
& = \mathbb{E}_{q(z|\eta_\theta^*)} \left[ \sum_{k=1}^{K} \left( 1 - \prod_{n=1}^{N} I \{ z_n \neq k \} \right) \right].
\end{align*}
\]

For a given optimal set of global variational parameters, \( g(\eta_\theta^*) \) can be computed with Monte-Carlo draws of the cluster indicators, \( z \sim iid q(z|\eta_\theta^*) \). We will denote Monte-Carlo expectations by \( \hat{\mathbb{E}}[\cdot] \).

### 6.2 Prior work

Our approach has similarities to Basu [2000] who also considers local robustness in Bayesian nonparametric models. They derive the analytic derivative of some posterior quantity with respect to the DP prior parameter \( \alpha \). MCMC samples are then needed to evaluate the derivative. They also consider the robustness with respect to the functional form of the prior on the cluster means. By viewing a posterior quantity as a functional of the functional form of the prior, they evaluate Gateaux derivatives in various directions to other priors.

In our approach, we analyze the sensitivity of the variational distribution, and compute derivatives automatically from the KL objective. Moreover, we use the derivatives not only for sensitivity, but for extrapolating posterior quantities as they vary over prior parameters.

### 6.3 Hyperparameter sensitivity

**General hyperparameter sensitivity.** We wish to approximate the sensitivity of \( g(\eta_\theta^*) \) to perturbations of the value of \( \alpha \) and to the functional form of \( p_0 \). To do this, we will call on a general result for the sensitivity of VB optima to vectors of real-valued hyperparameters. Suppose the exact posterior is parameterized by a real-valued hyperparameter \( \epsilon \), i.e., the posterior is given by \( p(\theta, z|y, \epsilon) \). In the
present work, \( \epsilon \) will parameterize perturbations to the prior, as we will describe in more detail shortly. Then the optimal variational approximation is also a function of \( \epsilon \) through the minimization of the KL divergence. We can define

\[
KL(\eta_\theta, \epsilon) := KL \left( q(\theta, z|\eta_\theta, \eta^*_z(\eta_\theta)) \big|\big| p(\theta, z|y, \epsilon) \right)
\]

\[
\eta^*_\theta(\epsilon) = \arg\min_{\eta_\theta} KL(\eta_\theta, \epsilon).
\] (6.3)

In general, the dependence of \( \eta^*_\theta(\epsilon) \) on \( \epsilon \) is complex and nonlinear, but under mild regularity conditions—which are satisfied in the present case—we may approximate it with a first-order Taylor series. Giordano et al. [2018, Theorem 2] gives these conditions as well as a closed form expression for this Taylor series. Without loss of generality, let \( \epsilon = 0 \) represent the unperturbed posterior, so that \( p(\theta, z|y, \epsilon = 0) = p(\theta, z|y) \). Define the Hessian

\[
H := \frac{\partial^2 KL(\eta_\theta, \epsilon)}{\partial \eta_\theta \partial \eta^T_\theta} \bigg|_{\eta_\theta = \eta^*_\theta, \epsilon = 0}
\]

\[
f_{\eta} := \frac{\partial^2 E_{q(\theta, z|\eta_\theta, \eta^*_z(\eta_\theta))} \left[ \log p(y, \theta, z|\epsilon) \right]}{\partial \eta_\theta \partial \epsilon^T} \bigg|_{\eta_\theta = \eta^*_\theta, \epsilon = 0}.
\]

Then

\[
\eta^*_\theta(\epsilon) - \eta^*_\theta(0) \approx \left. \frac{d\eta^*_\theta(\epsilon)}{d\epsilon^T} \right|_{\epsilon = 0} \epsilon = -H^{-1} f_{\eta} \epsilon.
\] (6.4)

Note that \( H \) and \( f_{\eta} \) can be easily evaluated using automatic differentiation without any need to re-optimize for different \( \epsilon \) [Maclaurin et al., 2015]. Furthermore, the Hessian \( H \) needs to be factorized (e.g. with a Cholesky decomposition) or inverted only once and then re-used to approximate \( \eta^*_\theta(\epsilon) \) for many different perturbations.

**Allowing nonlinearity in computationally easy steps.** Note that the complete mapping \( \epsilon \mapsto \hat{E}_{q(z|\eta^*_z)} \{\#\{\text{distinct clusters}\}\} \) is, in general, composed of many highly nonlinear steps:

\[
\epsilon \mapsto \eta^*_\theta(\epsilon) \mapsto \eta^*_z(\eta^*_\theta) \mapsto \text{Draws from } z \sim q(z|\eta^*_z) \mapsto \hat{E}_z \{\#\{\text{distinct clusters}\}\}.
\]

However, only the first step, \( \epsilon \mapsto \eta^*_\theta(\epsilon) \), is computationally intensive (re-solving the optimization problem in Eq. (6.3) with a new \( \epsilon \)), and it is precisely this first step which we approximate linearly using Eq. (6.4), i.e., with \( \epsilon \mapsto \eta^*_\theta(0) - H^{-1} f_{\eta} \epsilon \). Consequently, our approximations retain the nonlinearity in the mapping \( \eta^*_\theta \mapsto \hat{E}_z \{\#\{\text{distinct clusters}\}\} \).

Furthermore, we attempt to improve the linearity of the dependence of \( \eta^*_\theta \) on \( \epsilon \) by using an unconstrained parameterization for \( \eta_\theta \) following [Stan Team, 2015] and Kucukelbir et al. [2015].
Sensitivity to $\alpha$. Let $\alpha_0$ be a base value of $\alpha$ at which we optimize for $\eta^\ast$. By taking $\epsilon = \alpha - \alpha_0$, and 

$$f^\alpha_\eta := \frac{\partial^2 E_{q(\theta,z|\eta_0,\eta^\ast_\theta)} \left[ \log p(\nu|\alpha) \right]}{\partial \eta_0 \partial \alpha^T} \bigg|_{\eta_0 = \eta^\ast_\theta, \alpha = \alpha_0},$$

we can approximate, using Eq. (6.4),

$$\eta^{LIN}_0(\alpha) := \eta^\ast_\theta - H^{-1} f^\alpha_\eta (\alpha - \alpha_0) \approx \eta^\ast_\theta (\alpha).$$

We can then approximate $g(\eta^\ast_\theta (\alpha)) \approx g \left( \eta^{LIN}_0 (\alpha) \right)$.

On the Iris data, we evaluated the expected number of clusters for a range of $\alpha$ between 0.5 and 15. Then we chose three $\alpha_0$ values, 3, 8, and 13, and constructed the linear approximation centered at each of these values of $\alpha_0$. We note that the linear approximation is more accurate when extrapolating from more clusters to fewer clusters, as can be seen from the fact that the linear approximation in the rightmost panel of Figure 6.1 is accurate across the entire range of $\alpha$, whereas the leftmost panel is not.

![Figure 6.1: Comparison of the expected number of clusters computed by re-optimizing versus the linear approximation. The blue vertical line indicates the location of $\alpha_0$.](image)

Sensitivity to functional perturbations. In order to measure sensitivity to changing the functional form of the prior on the sticks, we define a parametrized class of multiplicative perturbations to the base density $p_0$ and apply Eq. (6.4). Specifically, fix a multiplicative perturbation $\phi(\cdot) : [0,1] \rightarrow (0, \infty)$ (recall that the stick lengths $\nu_k$ lie in $[0,1]$). Fix some $\delta \in [0,1]$. We then define a $\delta$-contaminated prior $p_c$ by

$$p_c(\nu_k|\delta, \phi) := \frac{p_0(\nu_k)\phi(\nu_k)^\delta}{\int_0^1 p_0(\nu_k')\phi(\nu_k')^\delta d\nu_k'}. \quad (6.5)$$
The contaminating prior $p_c$ is defined so that $\delta \in [0, 1]$ interpolates multiplicatively between the original prior, $p_0$, and a prior proportional to $\phi(\nu_k)p_0$. For example, we might consider a different prior for the sticks, say $p_1(\nu_k)$. Letting $\phi(\nu_k) = p_1(\nu_k)/p_0(\nu_k)$, we recover $p_0$ at $\delta = 0$ and swap the original prior for the new prior by taking $\delta \rightarrow 1$.

For a fixed $\phi$, we can use Eq. (6.4) by taking $\epsilon = \delta$ and

$$f^{k,\phi}_\eta := \frac{\partial^2 \mathbb{E}_{q(\theta, z|\eta_\theta, \eta_z(\eta_\theta))} \left[ \sum_{k=1}^K \log p_c(\nu_k|\delta, \phi) \right]}{\partial \eta_\theta \partial \delta} \bigg|_{\eta_\theta = \eta_\theta^*, \delta = 0} = \frac{\partial \mathbb{E}_{q(\theta, z|\eta_\theta, \eta_z(\eta_\theta))} \left[ \sum_{k=1}^K \log \phi(\nu_k) \right]}{\partial \eta_\theta} \bigg|_{\eta_\theta = \eta_\theta^*}.$$  

Because we have used a multiplicative perturbation, $f^{k,\phi}_\eta$ is linear in $\delta$, which we might expect to improve the fidelity of a linear approximation. Indeed, for the purposes of extrapolating to different priors when using VB based on KL divergence, this fact appears to recommend multiplicative perturbations amongst the class of nonlinear perturbations considered by Gustafson [1996b].

![Figure 6.2](image-url)

Figure 6.2: Left column: the original prior $p_0$ in orange, the perturbed prior $p_c$ in green. Right: linearly approximated vs. re-fitted expected number of clusters after the perturbation.

In Figure 6.2 we show results for the functional perturbation $\phi(\nu_k) = 1 - e^{\nu_k}$. We find that the linear approximation in this case was able to capture the direction of the perturbation (the expected number of clusters increased under the perturbation), although as $\delta \rightarrow 1$ the quality of the approximation degraded.

### 6.4 Further results

We consider a modification to the expected number of posterior clusters defined in Eq. (6.2). We wish to count only the clusters with at least $t$ observations, rather than
the total number of distinct clusters. Hence, our posterior target quantity becomes,

\[ g_t(\eta^*_\theta) := \mathbb{E}_{q(z|\eta^*_\theta)} \left[ \# \{ \text{clusters with at least } t \text{ data points} \} \right] \tag{6.6} \]

\[ = \mathbb{E}_{q(z|\eta^*_\theta)} \left[ \sum_{k=1}^K \mathbb{I} \left( \sum_{n=1}^N \mathbb{I} \{ z_n = k \} \right) \geq t \right] \tag{6.7} \]

Note that \( t = 0 \) reduces to the original target posterior defined in the Eq. (6.2).

Eq. (6.7) defines an *in-sample* quantity, that is, the expected number of clusters we expect to see in the current Iris dataset. We also consider a posterior predictive quantity, or the number of clusters we expect to see in a *new* dataset of 150 iris flowers, given our posterior knowledge about the stick-breaking process. This is an expectation over the variational distribution of the sticks \( \nu \), defined as

\[ g_{t,\text{pred}}(\eta^*_\theta) := \mathbb{E}_{q(\nu|\eta^*_\theta)} \left[ \# \{ \text{clusters in new data set with at least } t \text{ data points} \} \right] \tag{6.8} \]

\[ = \mathbb{E}_{q(\nu|\eta^*_\theta)} \left[ \sum_{k=1}^K \left( 1 - \sum_{i=0}^t \binom{n}{i} \pi_k^i (1 - \pi_k)^{n-i} \right) \right] \tag{6.9} \]

where \( \pi_k \) are the cluster probabilities induced by the sticks \( \nu \).

Like before, and as described in Section 6.3, only the dependence of \( \eta^*_\theta \) on the prior perturbation is approximated linearly. Given an \( \eta^*_\theta \) value, these expectations are computed with Monte-Carlo samples from the variational distribution \( q(z|\eta^*_\theta) \) for the in-sample expectation. Samples are drawn from \( q(\nu|\eta^*_\theta) \) in the predictive expectation.

Figure 6.3 shows both the in-sample and predictive expected number of *distinct* clusters (i.e. \( t = 0 \)). The linear approximation does equally well for both the in-sample and the predictive quantity. It works best when we set \( \alpha_0 = 13 \)—that is, when we approximate from having more clusters to fewer clusters.

Next, Figure 6.4 shows both the in-sample and predicted expected number of clusters with at least three observations (\( t = 3 \)). Again the linear approximation is best when we start at \( \alpha_0 = 13 \) and extrapolate to fewer clusters.

We next consider functional perturbations to the prior on the sticks. Figure 6.5 shows the effect of our choice of \( \phi \) on the expected number of distinct clusters (\( t = 0 \)). Both the in-sample and the predictive quantities are displayed. The approximation is most accurate at small \( \epsilon \), though the predictive quantity for the first perturbation was fairly accurate for the entire range of \( \epsilon \in [0, 1] \).

Finally, we consider the same functional perturbation to the stick priors, but with the threshold for counting a cluster at \( t = 3 \). Figure 6.5 displays the comparison of
Figure 6.3: The in-sample expected number of distinct clusters (Top), and the predictive expected number of distinct clusters (Bottom). Comparison of these values computed by re-optimizing versus the linear approximation. The blue vertical line indicates the location of $\alpha_0$.

the linear approximation against the refitted values, for both the in-sample and the predictive quantities. In this case, the choice of perturbation did not significantly move the number of thresholded clusters in the re-fitted values.
Figure 6.4: The in-sample expected number of distinct clusters with at least three observations (Top), and the corresponding predictive quantity (Bottom). Comparison of these values computed by re-optimizing versus the linear approximation. The blue vertical line indicates the location of $\alpha_0$. 
Figure 6.5: The effect of prior perturbation on the expected number of distinct clusters ($t = 0$). Left column: the original prior $p_0$ in red, the perturbed prior $p_c$ in blue. Middle: linearly approximated vs. re-fitted in-sample expected number of clusters. Right: linearly approximated vs. re-fitted predictive expected number of clusters.
CHAPTER 6. STICK BREAKING

Figure 6.6: The effect of prior perturbation on the expected number of clusters with at least three data points ($t = 3$). Left column: the original prior $p_0$ in red, the perturbed prior $p_c$ in blue. Middle: linearly approximated vs. re-fitted in-sample expected number of clusters. Right: linearly approximated vs. re-fitted predictive expected number of clusters.
Chapter 7

Selected future work

In this section, I will briefly discuss a few immediate extensions of the ideas introduced in this thesis.

7.1 Delta function approximations: Empirical Bayes and the EM algorithm

The linear response work above has been focused on calculating linear response covariances for variational Bayes. The same idea can be fruitfully applied to calculating corrected covariances for empirical Bayes algorithms when a Dirac delta function is used as an approximating measure for some components of the posterior. Consistency of the resulting covariance estimates can be proven when a Bayesian central limit theorem applies. Furthermore, by viewing the expectation-maximization (EM) algorithm as an empirical Bayes approximation, classical EM covariance matrix estimates can be seen as a special case of LRVB covariances. This observation expands on and generalizes an observation made briefly in Chapter 3 connecting the “supplemented expectation-maximization” (SEM) algorithm to LRVB [Meng and Rubin 1991].

Suppose we wish to approximate a posterior distribution on two unknowns, \( \theta \) and \( z \), given data \( x \): \( p(\theta, z|x) \). Suppose we believe that the variance of \( \theta \) is small enough that, for some \( \hat{\theta} \),

\[
p(\theta, z|x) \approx p(z|x, \hat{\theta}).
\]  

Such a situation may occur, for example, when a Bernstein von-Mises theorem applies to \( p(\theta|x) \), in which case \( \hat{\theta} \) would be taken as the MAP estimate \( \hat{\theta} = \arg\max_{\theta} \log p(\theta, x) \) [Van der Vaart 2000]. The approximation Eq. (7.1) is employed in empirical Bayes approximations [Carlin and Louis 2010] and in the

Fixing \( \theta \) at the MAP in the posterior can be formalized as a variational approximation. Let \( \delta(\cdot) \) denote the Dirac delta measure, and let \( Q = \{ q(z|\zeta) : \zeta \in \Omega_\zeta \} \) denote a family of approximating distributions that contains \( p(z|x, \theta) \) for all \( \theta \) in a neighborhood of \( \hat{\theta} \). Then define the variational approximation

\[
\hat{\theta}, \hat{\zeta} = \arg\min_{\vartheta, \zeta} \mathbb{E}_{q(z|\zeta)\delta(\theta - \vartheta)} [\log q(z|\theta) - \log p(\theta, z|x)].
\] (7.2)

Eq. (7.2) is similar to a KL divergence, except that it includes no entropy term for the degenerate \( \delta(\theta - \vartheta) \) distribution. As desired, Eq. (7.2) clearly has, as its optimum, the distribution \( \delta(\theta - \hat{\theta})p(z|x, \hat{\theta}) \), which is precisely the approximation used in empirical Bayes and EM.

The advantage of writing empirical Bayes and EM approximations as a single variational problem is that one can immediately use LRVB to calculate covariance approximations that take into account potential posterior variability in \( \theta \).

When applied to the EM algorithm, LRVB recovers existing covariance approximations for the posterior covariance of \( \theta \) (as discussed briefly in Chapter 3 above), as well as providing improved covariance estimates for \( p(z|x) \). Arguably, the LRVB perspective is also a unification and simplification of the variety of existing EM covariance estimation techniques (see McLachlan and Krishnan [2007] for a recent survey).

As far as the author is aware, this idea has not been previously applied to empirical Bayes. Note that the estimation of \( p(z|x, \hat{\theta}) \) by optimization is not a strict requirement. Suppose we are instead able to produce draws from \( p(z|x, \hat{\theta}) \), e.g., with MCMC. One could then either calculate the necessary derivatives for LRVB using the equivalence of covariances and derivatives, or use importance sampling with draws from a normal approximation to \( p(\theta|x) \).

### 7.2 Fréchet differentiability of variational objectives

In the functional sensitivity work in Chapter 6, we motivate using the Banach space of log posteriors with an intuitive appeal to the linearity of KL divergence in this space. It turns out that there is a deeper reason for this choice of space: among the class of nonlinear perturbations considered by Gustafson [1996b], KL divergence is Fréchet differentiable only in the vector space of log posteriors equipped with the infinity norm. This result stands in stark contrast to the full Bayesian posterior, and may well have important consequences for the asymptotic performance of VB versions of worst-case functional perturbations of Gustafson [1996b].
It suffices to sketch the fact that KL divergence is discontinuous in $L_p$ norms for $p < \infty$. The proof of discontinuity follows from an intuitively obvious property of KL divergence. Choose some set assigned non-zero measure by $q$. No matter how small the measure of this set is, the KL divergence $KL(q||p)$ can be made arbitrarily large in magnitude by forcing $p$ close to zero within this set. Thus, two distributions $p_1$ and $p_2$ that differ only on a set of very small measure may have very different KL divergences $KL(q||p_1)$ and $KL(q||p_2)$. But because the set on which they differ has small measure, $p_1$ and $p_2$ will be close in any $L_p$ norm for finite $p$.

I will now formalize this intuition. Let $\theta$ be a scalar in $[0, 1]$. Let $q(\theta)$ be a fixed variational approximation and $p(\theta)$ be a target distribution (e.g., a posterior), each with a continuous density defined with respect to the Lebesgue measure on $[0, 1]$. Assume that $KL(q||p) < \infty$. Let $\theta_0$ be a point where both $q(\theta)$ and $p(\theta)$ are bounded strictly away from zero. Let an alternative “contamination” target distribution $p_c(\theta|\epsilon, \delta)$ be defined as

$$
p_c(\theta|\epsilon, \delta) = \begin{cases} 
\delta : & \|\theta - \theta_0\|_2 < \epsilon \\
\frac{p(\theta)}{C(\epsilon, \delta)} & \text{otherwise}
\end{cases}
$$

where $C(\epsilon, \delta)$ is a constant chosen so that $p_c(\theta|\epsilon, \delta)$ is normalized. Because the density $p(\theta)$ is continuous, $p_c(\theta|\epsilon, \delta)$ exists for all finite $\epsilon$.

Let $\lambda$ be a base measure that assigns nonzero mass to a neighborhood of $\theta_0$, and, for $\rho < \infty$, define an $L_\rho$ norm with respect to $\lambda$ as

$$
\|u(\theta)\|_\rho := \left(\int |u(\theta)|^\rho \lambda(d\theta)\right)^{1/\rho}.
$$

We assume that $\|p(\theta)\|_\rho < \infty$. For a fixed $\rho$, and any $n = 1, \ldots$, define the sequences $\delta_n = \exp(-n)$ and $\epsilon_n = n^{-1/2}$. Because both $\delta_n$ and $\epsilon_n$ go to zero as $n \to \infty$, $C(\epsilon_n, \delta_n) \to 1$ as $n \to \infty$. Then

$$
\|p(\theta) - p_c(\theta|\epsilon_n, \delta_n)\|_\rho^\rho = \int |p(\theta) - p_c(\theta|\epsilon_n, \delta_n)|^\rho \lambda(d\theta)
$$

$$
= \int_{\|\theta - \theta_0\|_2 < \epsilon} |p(\theta) - p_c(\theta|\epsilon_n, \delta_n)|^\rho \lambda(d\theta) + \\
\int_{\|\theta - \theta_0\|_2 \geq \epsilon} |p(\theta) - p_c(\theta|\epsilon_n, \delta_n)|^\rho \lambda(d\theta)
$$

$$
= \int_{\|\theta - \theta_0\|_2 < \epsilon} |p(\theta) - \delta_n|^\rho \lambda(d\theta) + \\
|1 - C(\epsilon_n, \delta_n)|^\rho \int_{\|\theta - \theta_0\|_2 \geq \epsilon} p(\theta)^\rho \lambda(d\theta)
$$
Because $C(\epsilon_n, \delta_n) \to 1$ as $n \to \infty$, the second term vanishes as $n$ gets large, and because $\epsilon_n$ and $\delta_n$ both go to zero as $n$ gets large, the first term vanishes as $n$ gets large. So as $n$ grows, the $L_\rho$ distance between $p(\theta)$ and $p_c(\theta|\epsilon_n, \delta_n)$ goes to zero.

Now, consider the KL divergence between $q$ and $p_c(\theta|\epsilon_n, \delta_n)$.

$$KL(q(\theta)||p_c(\theta|\epsilon_n, \delta_n)) = \int \log q(\theta)q(\theta)d\theta - \int \log p_c(\theta|\epsilon_n, \delta_n)q(\theta)d\theta.$$

The first term is bounded by assumption. The second term is

$$\int \log p_c(\theta|\epsilon_n, \delta_n)q(\theta)d\theta = \log \delta_n \int _{\|\theta - \theta_0\|_2 < \epsilon} q(\theta)d\theta + \int _{\|\theta - \theta_0\|_2 \geq \epsilon} (\log p(\theta) - \log C(\epsilon_n, \delta_n)) q(\theta)d\theta.$$

The second term is bounded by assumption. By continuity of $q(\theta)$, the first term is approximately

$$q(\theta_0)\epsilon_n \log \delta_n = q(\theta_0)\frac{-n}{\sqrt{n}} = -q(\theta_0)\sqrt{n}.$$

Consequently,

$$\lim_{n \to \infty} |KL(q(\theta)||p_c(\theta|\epsilon_n, \delta_n)) - KL(q(\theta)||p(\theta))| = \infty.$$

Thus, the map $p \mapsto KL(q||p)$ is discontinuous in the $L_\rho$ metric at $p(\theta)$. In general, a $q$ chosen to optimize $KL(q||p)$ will inherit this discontinuous dependence on $p$.

One way to avoid this problem is simply to use the $\rho = \infty$ norm,

$$\|u(\theta)\|_\infty := \text{esssup}_\theta |\log u(\theta)|.$$

In this norm, $p_c(\theta|\epsilon_n, \delta_n)$ is no longer close to $p(\theta)$. This is the norm we choose in Chapter 6. A formal proof of Fréchet differentiability of KL divergence and, in turn, of the variational approximation is then mostly bookkeeping.

Note that Gustafson [1996b] allows only pointwise non-negative additive perturbations. To define a perturbation that ablates mass in a certain region using Gustafson’s formalism, one adds mass everywhere else and re-normalizes. Naturally, this is formally considered to be a large perturbation in Gustafson’s metric.

1Technically, this is a closed subspace of a Banach space, and Gustafson [1996b] is considering derivatives evaluated on the boundary. Most functional analysis literature only considers differentiability in the interior of a Banach space (e.g., Zeidler [1986]) This technical point goes unmentioned, but seems unlikely to cause any real difficulties.

2I believe Gustafson [1996b] uses this definition to simplify the computation of worst-case perturbations rather than out of consideration of Fréchet differentiability.
As a consequence, Gustafson does not consider $p_c(\theta|\epsilon_n,\delta_n)$ to be “close” to $p(\theta)$ either, even though he uses $L_p$ norms. Thus, by following [Gustafson 1996b] and allowing only pointwise non-negative perturbations, one can avoid the above difficulties in the discontinuity of KL divergence, but at the cost of choosing a notion of “distance” that does not correspond well to intuition. Of course, one might argue that the $\| \cdot \|_\infty$ norm does not correspond well to intuition, either. It is interesting to ask whether one can find a metric that corresponds better with intuition and yet renders KL divergence continuous, perhaps by imposing smoothness requirements on the space of perturbations.

Finally, I speculate that these are not idle theoretical concerns. One of the key benefits of Gustafson’s theory of local robustness is the ability to calculate worst-case functional perturbations. As the amount of data grows, in $L_p$ norms these worst-case perturbations tend to have both negative and positive lobes, and to become very spiky and local. In other words, they approach the direction in which KL divergence is discontinuous. It is intuitively plausible that the techniques of [Gustafson 1996b], applied naively to VB, could give quite misleading measures of sensitivity.

### 7.3 Higher order sensitivity approximations

I will briefly discuss extending the results of Chapter 4 to higher-order approximations. The results of Chapter 4 can be extended to generic hyperparameters (other than re-weighting vectors) with essentially no modification, so the discussion in this section applies equally to the prior and functional sensitivity results of Chapter 6. Nevertheless, for clarity and continuity, I will continue to discuss the hyperparameters $w$, and, in general, to use the notation of Chapter 4.

By continuing the Taylor expansion of $G(\theta, w)$ and requiring an additional set complexity condition on higher order derivatives of a form analogous to Condition 4.1,

$$\sup_{w \in \mathcal{W}_\delta} \sup_{\theta \in \Omega_\delta} \left\| \frac{1}{N} \sum_{n=1}^{N} (w_n - 1) \frac{\partial^k g_n(\theta)}{\partial \theta^k} \right\|_1 \leq \delta,$$

one can construct accuracy bounds on higher-order Taylor series approximations to $\hat{\theta}(w)$ using a proof closely related to that of the first-order approximation. It turns out that, for a condition of the form Eq. (7.3) for all derivatives up to order $k \leq k_{\text{max}}$, the accuracy of the resulting $k_{\text{max}}$th order infinitesimal jackknife is proportional to $\delta^{k_{\text{max}}+1}$. Note that the results of Chapter 4 are achieved by taking $k_{\text{max}} = 1$. 

The possibility of higher-order approximations opens up the possibility of usable local approximations to bootstrap distributions. Let us consider bootstrap weights \( w \). It turns out that the linear approximation applied with bootstrap weights recovers only the asymptotic normal approximation covariances. The bootstrap is typically used in order to achieve accuracy rates that are faster than the normal approximation by a factor of \( \frac{1}{\sqrt{N}} \) [Hall 2013, Chapter 3]. Intuitively, because Eq. (7.3) is approximately a sample covariance, one might expect it to hold for each derivative with \( \delta = \frac{1}{\sqrt{N}} \) given a sufficiently smooth objective and well-behaved parameter set. The bootstrap weights have size \( C_w \approx \sqrt{N} \). One can see that, with \( k_{\text{max}} = 1 \) (as in Chapter 4), the linear approximation approaches the bootstrap estimator \( \hat{\theta}(w) \) at a rate of only \( N^{-1/2} \). However, by taking \( k_{\text{max}} = 3 \) (a fourth-order approximation), the Taylor series approximation error in the approximation of \( \hat{\theta}(w) \) would shrink at a rate \( N^{-3/2} \), which is faster than the rate at which the bootstrap \( \hat{\theta}(w) \) approaches the truth. Consequently, accuracy results for the bootstrap would also hold for a fourth-order approximation. It is interesting to ask whether third-order approximation, which matches but does not exceed the rate of the bootstrap, could also provide asymptotic benefits. This argument has left many details to be filled in, but the intuition seems clear enough.

At first, it might seem that a prohibitive practical difficulty with higher-order approximations is the calculation of higher-order tensors. To calculate a \( k \)th order approximation requires calculation of \( k + 1 \)th order derivative arrays. The size of these arrays scale as \( D^{k+1} \), which can be prohibitive for even moderately sized applications. However, this difficulty can be entirely avoided by using forward rather than reverse mode automatic differentiation. A working implementation can be found in the \texttt{ParametricSensitivityTaylorExpansion} class of my \texttt{vittles} Python package.
Appendix A

Supplementary Model Details

A.1 Normal-Poisson details

In this section, we use this model to provide a detailed, step-by-step description of a simple LRVB analysis.

The full joint distribution for the model in Eq. (3.12) is

\[
\log p(y, z, \beta, \tau) = \sum_{n=1}^{N} \left( -\frac{1}{2} \tau z_n^2 + x_n \tau \beta z_n - \frac{1}{2} x_n^2 \tau \beta^2 - \frac{1}{2} \log \tau \right)
+ \sum_{n=1}^{N} \left( -\exp(z_n) + z_n y_n \right) - \frac{1}{2} \sigma_\beta^2 \beta^2 + (\alpha_\tau - 1) \log \tau - \beta_\tau + \text{Constant}
\]

We find a mean-field approximation under the factorization \( q(\beta, \tau, z) = q(\beta) q(\tau) \prod_{n=1}^{N} q(z_n) \). By inspection, the log joint is quadratic in \( \beta \), so the optimal \( q(\beta) \) will be Gaussian [Bishop, 2006]. Similarly, the log joint is a function of \( \tau \) only via \( \tau \) and \( \log \tau \), so the optimal \( q(\tau) \) will be gamma. However, the joint does not take a standard exponential family form in \( z_n \):

\[
\log p(z_n | y, \beta, \tau) = (x_n \tau \beta + y_n) z_n - \frac{1}{2} \tau z_n^2 - \exp(z_n) + \text{Constant}
\]

The difficulty is with the term \( \exp(z_n) \). So we make the further restriction that

\[
q(z_n) = \mathcal{N}(\cdot) = q\left(z_n; \mathbb{E}[z_n], \mathbb{E}[z_n^2]\right).
\]

Fortunately, the troublesome term has an analytic expectation, as a function of the mean parameters, under this variational posterior:

\[
\mathbb{E}_{q_\alpha} \left[ \exp(z_n) \right] = \exp \left( \mathbb{E}_{q_\alpha} [z_n] + \frac{1}{2} \left( \mathbb{E}_{q_\alpha} [z_n^2] - \mathbb{E}_{q_\alpha} [z_n]^2 \right) \right).
\]
We can now write the variational distribution in terms of the following mean parameters:

\[ m = (\mathbb{E}_{q_{\alpha}}[\beta], \mathbb{E}_{q_{\alpha}}[\beta^2], \mathbb{E}_{q_{\alpha}}[\tau], \mathbb{E}_{q_{\alpha}}[\log \tau], \mathbb{E}_{q_{\alpha}}[z_1], \mathbb{E}_{q_{\alpha}}[z_1^2], ..., \mathbb{E}_{q_{\alpha}}[z_N], \mathbb{E}_{q_{\alpha}}[z_N^2])^T. \]

Calculating the LRVB covariance consists of roughly four steps:

1. finding the MFVB optimum \( q^* \),
2. computing the covariance \( V \) of \( q^* \),
3. computing \( H \), the Hessian of \( L(m) \), for \( q^* \), and
4. computing the matrix inverse and solving \( (I - VH)^{-1}V \).

For step (1), the LRVB correction is agnostic as to how the optimum is found. In our experiments below, we follow a standard coordinate ascent procedure for MFVB [Bishop, 2006]. We analytically update \( q(\beta) \) and \( q(\tau) \). Given \( q(\beta) \) and \( q(\tau) \), finding the optimal \( q(z) \) becomes \( N \) separate two-dimensional optimization problems; there is one dimension for each of the mean parameters \( \mathbb{E}_{q_{\alpha}}[z_n] \) and \( \mathbb{E}_{q_{\alpha}}[z_n^2] \). In our examples, we solved these problems sequentially using IPOPT [Wächter and Biegler, 2006].

To compute \( V \) for step (2), we note that by the mean-field assumption, \( \beta, \tau, \) and \( z_n \) are independent, so \( V \) is block diagonal. Since we have chosen convenient variational distributions, the mean parameters have known covariance matrices. For example, from standard properties of the normal distribution, \( \text{Cov}(\beta, \beta^2) = 2\mathbb{E}_{q_{\alpha}}[\beta](\mathbb{E}_{q_{\alpha}}[\beta^2] - \mathbb{E}_{q_{\alpha}}[\beta]^2) \).

For step (3), the mean parameters for \( \beta \) and \( \tau \) co-occur with each other and with all the \( z_n \), so these four rows of \( H \) are expected to be dense. However, the mean parameters for \( z_n \) never occur with each other, so the bulk of \( H \)—the \( 2N \times 2N \) block corresponding to the mean parameters of \( z \)—will be block diagonal (Fig. (A.1b)). The Hessian of \( L(m) \) can be calculated analytically, but we used the autodifferentiation software JuMP [Lubin and Dunning, 2015].

Finally, for step (4), we use the technique in Section 3.2 to exploit the sparsity of \( V \) and \( H \) (Fig. (A.1c)) in calculating \( (I - VH)^{-1} \).
A.2 Random effects model details

As introduced in Section 3.3, our model is:

\[ y_n | \beta, z, \tau \overset{\text{indep}}{\sim} N (\beta^T x_n + r_n z_{k(n)}, \tau^{-1}) \]
\[ z_k | \nu \overset{iid}{\sim} N (0, \nu^{-1}) \]

With the priors:

\[ \beta \sim N (0, \Sigma_\beta) \]
\[ \nu \sim \Gamma (\alpha_\nu, \beta_\nu) \]
\[ \tau \sim \Gamma (\alpha_\tau, \beta_\tau) \]

We will make the following mean field assumption:

\[ q (\beta, z, \tau, \nu) = q (\nu) q (\tau) q (\beta) \prod_{k=1}^K q (z_k) \]

We have \( n \in \{1, ..., N\} \), and \( k \in \{1, ..., K\} \), and \( k(n) \) matches an observation \( n \) to a random effect \( k \), allowing repeated observations of a random effect. The full joint log likelihood is:

\[
\log p (y_n | z_{k(n)}, \tau, \beta) = -\frac{\tau}{2} (y_n - \beta^T x_n - r_n z_{k(n)})^2 + \frac{1}{2} \log \tau + \text{Constant}
\]

\[
\log p (z_k | \nu) = -\frac{\nu}{2} z_k^2 + \frac{1}{2} \log \nu + \text{Constant}
\]

\[
\log p (\beta) = -\frac{1}{2} \text{trace} (\Sigma_\beta^{-1} \beta^T) + \text{Constant}
\]

\[
\log p (\tau) = (\alpha_\tau - 1) \log \tau - \beta_\tau \tau + \text{Constant}
\]

\[
\log p (\nu) = (\alpha_\nu - 1) \log \nu - \beta_\nu \nu + \text{Constant}
\]

\[
\log p (y, \tau, \beta, z) = \sum_{n=1}^N \log p (y_n | z_{k(n)}, \tau, \beta) + \sum_{k=1}^K \log p (z_k | \nu) + \log p (\beta) + \log p (\nu) + \log p (\tau)
\]
Expanding the first term of the conditional likelihood of \( y_n \) gives

\[
-\frac{\tau}{2} \left( y_n - \beta^T x_n - r_n z_{k(n)} \right)^2
\]

\[
= -\frac{\tau}{2} (y_n^2 - 2y_n x_n^T \beta - 2y_n r_n z_{n(k)} + \text{trace} \left( x_n x_n^T \beta \beta^T \right) + r_n^2 z_{k(n)}^2 + 2r_n x_n^T \beta z_{k(n)})
\]

By grouping terms, we can see that the mean parameters will be

\[
q(\beta) = q(\beta; \mathbb{E}_{q_\alpha}[\beta], \mathbb{E}_{q_\alpha}[\beta \beta^T])
\]

\[
q(z_k) = q(z_k; \mathbb{E}_{q_\alpha}[z_k], \mathbb{E}_{q_\alpha}[z_k^2])
\]

\[
q(\tau) = q(\tau; \mathbb{E}_{q_\alpha}[\tau], \mathbb{E}_{q_\alpha}[\log \tau])
\]

\[
q(\nu) = q(\nu; \mathbb{E}_{q_\alpha}[\nu], \mathbb{E}_{q_\alpha}[\log \nu])
\]

It follows that the optimal variational distributions are \( q(\beta) = \text{multivariate normal} \), \( q(z_k) = \text{univariate normal} \), and \( q(\tau) \) and \( q(\nu) \) will be gamma. We performed standard coordinate ascent on these distributions \[Bishop\ 2006\].

As in Section 3.3, we implemented this model in the autodifferentiation software JuMP \[Lubin and Dunning\ 2015\]. This means conjugate coordinate updates were easy, since the natural parameters corresponding to a mean parameters are the first derivatives of the log likelihood with respect to the mean parameters. For example, denoting the log likelihood at step \( s \) by \( L_s \), the update for \( q_{s+1}(z_k) \) will be:

\[
\log q_{s+1}(z_k) = \frac{\partial \mathbb{E}_{q_\alpha}[L_s]}{\partial \mathbb{E}_{q_\alpha}[z_k]} z_k + \frac{\partial \mathbb{E}_{q_\alpha}[L_s]}{\partial \mathbb{E}_{q_\alpha}[z_k^2]} z_k^2 + \text{Constant}
\]

Given the partial derivatives of \( L_s \) with respect to the mean parameters, the updated mean parameters for \( z_k \) can be read off directly using standard properties of the normal distribution.

The variational covariance matrices are all standard. We can see that \( H \) will have nonzero terms in general (for example, the three-way interaction \( \mathbb{E}_{q_\alpha}[\tau] \mathbb{E}_{q_\alpha}[z_{k(n)}] \mathbb{E}_{q_\alpha}[z_k] \)), and that LRVB will be different from MFVB. As usual in our models, \( H \) is sparse, and we can easily apply the technique in section Section 3.2 to get the covariance matrix excluding the random effects, \( z \).

### A.3 Multivariate Normal mixture details

In this section we derive the basic formulas needed to calculate Eq. (3.7) for a finite mixture of normals, which is the model used in Section 3.3. We will follow the notation introduced in Section 3.3.
APPENDIX A. SUPPLEMENTARY MODEL DETAILS

Let each observation, \( x_n \), be a \( P \times 1 \) vector. We will denote the \( P \)th component of the \( n \)th observation \( x_n \), with a similar pattern for \( z \) and \( \mu \). We will denote the \( p, q \)th entry in the matrix \( \Lambda_k \) as \( \Lambda_{k,pq} \). The data generating process is as follows:

\[
P(x|\mu, \pi, \Lambda) = \prod_{n=1}^{N} P(x_n|z_n, \mu, \Lambda) \prod_{k=1}^{K} P(z_{nk}|\pi_k)
\]

\[
\log P(x_n|z_n, \mu, \Lambda) = \sum_{n=1}^{N} z_{nk} \log \phi_k(x_n) + \text{Constant}
\]

\[
\log \phi_k(x) = -\frac{1}{2} (x - \mu_k)^T \Lambda_k (x - \mu_k) + \frac{1}{2} \log |\Lambda_k| + \text{Constant}
\]

\[
\log P(z_{nk}|\pi_k) = \sum_{k=1}^{K} z_{nk} \log \pi_k + \text{Constant}
\]

It follows that the log posterior is given by

\[
\log P(z, \mu, \pi, \Lambda|x) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \left( \log \pi_k - \frac{1}{2} (x_n - \mu_k)^T \Lambda_k (x_n - \mu_k) + \frac{1}{2} \log |\Lambda_k| \right) + \sum_{k=1}^{K} \log p(\mu_k) + \sum_{k=1}^{K} \log p(\Lambda_k) + \log p(\pi) + \text{Constant}
\]

We used a multivariate normal prior for \( \mu_k \), a Wishart prior for \( \Lambda_k \), and a Dirichlet prior for \( \pi \). In the simulations described in Section 3.3, we used the following prior parameters for the VB model:

\[
p(\mu_k) = \mathcal{N}(0_P, \text{diag}_P(0.01)^{-1})
\]

\[
p(\Lambda_k) = \text{Wishart}((\text{diag}_P(0.01)), 1)
\]

\[
p(\pi) = \text{Dirichlet}(5K)
\]

Here, \( \text{diag}_P(a) \) is a \( P \)-dimensional diagonal matrix with \( a \) on the diagonal, and \( 0_P \) is a length \( P \) vector of the value 0, with a similar definition for \( 5K \). Unfortunately, the function we used for the MCMC calculations, \texttt{rnmixGibbs} in the package \texttt{bayesm}, uses a different form for the \( \mu_k \) prior. Specifically, \texttt{rnmixGibbs} uses the prior

\[
p_{MCMC}(\mu_k|\Lambda_k) = \mathcal{N}(0, a^{-1} \Lambda_k^{-1})
\]

where \( a \) is a scalar. There is no way to exactly match \( p_{MCMC}(\mu_k) \) to \( p(\mu_k) \), so we simply set \( a = 0.01 \). Since our datasets are all reasonably large, the prior was
dominated by the likelihood, and we found the results extremely insensitive to the prior on $\mu_k$, so this discrepancy is of no practical importance.

The parameters $\mu_k$, $\Lambda_k$, $\pi$, and $z_n$ will each be given their own variational distribution. For $q_{\mu_k}$ we will use a multivariate normal distribution; for $q_{\Lambda_k}$ we will use a Wishart distribution; for $q_{\pi}$ we will use a Dirichlet distribution; for $q_{z_n}$ we will use a Multinoulli (a single multinomial draw). These are all the optimal variational choices given the mean field assumption and the conditional conjugacy in the model.

The sufficient statistics for $\mu_k$ are all terms of the form $\mu_{kp}$ and $\mu_{kp}\mu_{kq}$. Consequently, the sub-vector of $\theta$ corresponding to $\mu_k$ is

$$
\theta_{\mu_k} = \left( \begin{array}{c}
\mu_k1 \\
\vdots \\
\mu_{kp} \\
\mu_k1\mu_k1 \\
\mu_k1\mu_k2 \\
\vdots \\
\mu_kP\mu_kP
\end{array} \right)
$$

We will only save one copy of $\mu_{kp}\mu_{kq}$ and $\mu_{kq}\mu_{kp}$, so $\theta_{\mu_k}$ has length $P + \frac{1}{2} (P + 1) P$. For all the parameters, we denote the complete stacked vector without a $k$ subscript:

$$
\theta_\mu = \left( \begin{array}{c}
\theta_{\mu_1} \\
\vdots \\
\theta_{\mu_K}
\end{array} \right)
$$

The sufficient statistics for $\Lambda_k$ are all the terms $\Lambda_{kpq}$ and the term $\log |\Lambda_k|$. Again, since $\Lambda$ is symmetric, we do not keep redundant terms, so $\theta_{\Lambda_k}$ has length $1 + \frac{1}{2} (P + 1) P$. The sufficient statistic for $\pi$ is the $K$-vector $(\log \pi_1, \ldots, \log \pi_K)$. The sufficient statistics for $z$ are simply the $N \times K$ values $z_{nk}$ themselves.

In terms of Section 3.2, we have

$$
\alpha = \left( \begin{array}{c}
\theta_\mu \\
\theta_\Lambda \\
\theta_\pi
\end{array} \right)
$$

$$
z = \left( \begin{array}{c}
\theta_z
\end{array} \right)
$$

That is, we are primarily interested in the covariance of the sufficient statistics of $\mu$, $\Lambda$, and $\pi$. The latent variables $z$ are nuisance parameters.
APPENDIX A. SUPPLEMENTARY MODEL DETAILS

To put the log likelihood in terms useful for LRVB, we must express it in terms of the sufficient statistics, taking into account the fact the $\theta$ vector does not store redundant terms (e.g. it will only keep $\Lambda_{ab}$ for $a < b$ since $\Lambda$ is symmetric).

\[-\frac{1}{2} \left( x_n - \mu_k \right)^T \Lambda_k \left( x_n - \mu_k \right) \]

\[= -\frac{1}{2} \text{trace} \left( \Lambda_k (x_n - \mu_k) (x_n - \mu_k)^T \right) \]

\[= -\frac{1}{2} \sum \sum (\Lambda_{k,ab} (x_{n,a} - \mu_{k,a}) (x_{n,b} - \mu_{k,b})) \]

\[= -\frac{1}{2} \sum \sum (\Lambda_{k,ab} x_{n,a} \mu_{k,a} - \Lambda_{k,ab} x_{n,b} \mu_{k,a} + \Lambda_{k,ab} x_{n,a} x_{n,b}) \]

\[= -\frac{1}{2} \sum \Lambda_{k,aa} \left( \mu_k^2 \right)^a + \sum \Lambda_{k,aa} x_{n,a} \mu_{k,a} - \frac{1}{2} \sum \Lambda_{k,aa} (x_n^2)^a - \]

\[\frac{1}{2} \sum \sum \Lambda_{k,ab} x_{n,a} \mu_{k,b} + \sum \Lambda_{k,ab} x_{n,b} \mu_{k,b} - \frac{1}{2} \sum \Lambda_{k,ab} x_{n,a} x_{n,b} \]

\[= -\frac{1}{2} \sum \Lambda_{k,aa} \left( \mu_k^2 \right)^a + \sum \Lambda_{k,aa} x_{n,a} \mu_{k,a} - \frac{1}{2} \sum \Lambda_{k,aa} (x_n^2)^a - \]

\[\sum \Lambda_{k,ab} x_{n,a} \mu_{k,b} + \sum \Lambda_{k,ab} (x_{n,b} \mu_{k,b} + x_{n,a} \mu_{k,a}) - \sum \Lambda_{k,ab} x_{n,a} x_{n,b} \]

The MFVB updates and covariances in $V$ are all given by properties of standard distributions. To compute the LRVB corrections, it only remains to calculate the Hessian, $H$. These terms can be read directly off the posterior. First we calculate derivatives with respect to components of $\mu$.

\[\frac{\partial^2 H}{\partial \mu_{k,a} \partial \Lambda_{k,ab}} = \sum_i z_{nk} x_{n,b} \]

\[\frac{\partial^2 H}{\partial (\mu_{k,a} \mu_{k,b}) \partial \Lambda_{k,ab}} = -\left( \frac{1}{2} \right) 1(a=b) \sum_n z_{nk} \]

\[\frac{\partial^2 H}{\partial \mu_{k,a} \partial z_{nk}} = \sum_b \Lambda_{k,ab} x_{n,b} \]

\[\frac{\partial^2 H}{\partial (\mu_{k,a} \mu_{k,b}) \partial z_{nk}} = -\left( \frac{1}{2} \right) 1(a=b) \Lambda_{k,ab} \]
All other \( \mu \) derivatives are zero. For \( \Lambda \),

\[
\frac{\partial^2 H}{\partial \Lambda_{k,ab} \partial z_{nk}} = -\left(\frac{1}{2}\right)^{(a=b)} \left( x_{n,a} x_{n,b} - \mu_{k,a} x_{n,b} - \mu_{k,b} x_{n,a} + \mu_{k,a} \mu_{k,b} \right)
\]

\[
\frac{\partial^2 H}{\partial \log |\Lambda_k| \partial z_{nk}} = \frac{1}{2}
\]

The remaining \( \Lambda \) derivatives are zero. The only nonzero second derivatives for \( \log \pi \) are to \( Z \) and are given by

\[
\frac{\partial^2 H}{\partial \log \pi_k \partial z_{nk}} = 1
\]

Note in particular that \( H_{zz} = 0 \), allowing efficient calculation of Eq. (3.11).

### A.4 MNIST details

For a real-world example, we applied LRVB to the unsupervised classification of two digits from the MNIST dataset of handwritten digits. We first preprocess the MNIST dataset by performing principle component analysis on the training data’s centered pixel intensities and keeping the top 25 components. For evaluation, the test data is projected onto the same 25-dimensional subspace found using the training data.

We then treat the problem of separating handwritten 0s from 1s as an unsupervised clustering problem. We limit the dataset to instances labeled as 0 or 1, resulting in 12665 training and 2115 test points. We fit the training data as a mixture of multivariate Gaussians. Here, \( K = 2 \), \( P = 25 \), and \( N = 12665 \). Then, keeping the \( \mu \), \( \Lambda \), and \( \pi \) parameters fixed, we calculate the expectations of the latent variables \( z \) in Eq. (3.13) for the test set. We assign test set data point \( x_n \) to whichever component has maximum a posteriori expectation. We count successful classifications as test set points that match their cluster’s majority label and errors as test set points that are different from their cluster’s majority label. By this measure, our test set error rate was 0.08. We stress that we intend only to demonstrate the feasibility of LRVB on a large, real-world dataset rather than to propose practical methods for modeling MNIST.

### A.5 LKJ priors for covariance matrices

In this section we briefly derive closed-form expressions for using an LKJ prior with a Wishart variational approximation.
**Proposition A.1.** Let $\Sigma$ be a $K \times K$ positive definite covariance matrix. Define the $K \times K$ matrix $M$ such that

$$M_{ij} = \begin{cases} \sqrt{\Sigma_{ij}} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Define the correlation matrix $R$ as

$$R = M^{-1} \Sigma M^{-1}.$$ 

Define the LKJ prior on $R$ with concentration parameter $\xi$ ([Lewandowski et al., 2009]):

$$p_{\text{LKJ}}(R|\xi) \propto |R|^{\xi - 1}.$$ 

Let $q(\Sigma|V^{-1}, \nu)$ be an inverse Wishart distribution with matrix parameter $V^{-1}$ and $\nu$ degrees of freedom. Then

$$\mathbb{E}_q[\log |R|] = \log |V^{-1}| - \psi_K \left( \frac{\nu}{2} \right) - K \sum_{k=1}^{K} \log \left( (V^{-1})_{kk} \right) + K \psi \left( \frac{\nu - K + 1}{2} \right) + \text{Constant}$$

$$\mathbb{E}_q[\log p_{\text{LKJ}}(R|\xi)] = (\xi - 1) \mathbb{E}_q[\log |R|] + \text{Constant},$$

where Constant does not depend on $V$ or $\nu$. Here, $\psi_K$ is the multivariate digamma function.

**Proof.** First note that

$$\log |\Sigma| = 2 \log |M| + \log |R|$$

$$= 2 \sum_{k=1}^{K} \log \sqrt{\Sigma_{kk}} + \log |R|$$

$$= \sum_{k=1}^{K} \log \Sigma_{kk} + \log |R| \Rightarrow$$

$$\log |R| = \log |\Sigma| - \sum_{k=1}^{K} \log \Sigma_{kk}. \quad (A.1)$$

By Eq. B.81 in Bishop [2006], a property of the inverse Wishart distribution is the following relation.

$$E_q[\log |\Sigma|] = \log |V^{-1}| - \psi_K \left( \frac{\nu}{2} \right) - K \log 2, \quad (A.2)$$
where $\psi_K$ is the multivariate digamma function. By the marginalization property of the inverse Wishart distribution,

$$
\Sigma_{kk} \sim \text{InverseWishart} \left( (V^{-1})_{kk}, \nu - K + 1 \right) \Rightarrow 
E_q [\log \Sigma_{kk}] = \log ((V^{-1})_{kk}) - \psi \left( \frac{\nu - K + 1}{2} \right) - \log 2. \quad (A.3)
$$

Plugging Eq. (A.2) and Eq. (A.3) into Eq. (A.1) gives the desired result. □

### A.6 Logistic GLMM model details

In this section we include extra details about the model and analysis of Section 2.4. We will continue to use the notation defined therein. We use $Constant$ to denote any constants that do not depend on the prior parameters, parameters, or data. The log likelihood is

$$
\log p(y_{it}|u_t, \beta) = y_{it} \log \left( \frac{p_{it}}{1 - p_{it}} \right) + \log (1 - p_{it}) \\
= y_{it} \rho + \log (1 - p_{it}) + \text{Constant}
$$

$$
\log p(u|\mu, \tau) = -\frac{1}{2} \tau \sum_{t=1}^{T} (u_t - \mu)^2 - \frac{1}{2} T \log \tau \\
= -\frac{1}{2} \tau \sum_{t=1}^{T} (u_t^2 - \mu u_t + \mu^2) - \frac{1}{2} T \log \tau + \text{Constant}
$$

$$
\log p(\mu, \tau, \beta) = -\frac{1}{2} \sigma_\mu^{-2} (\mu^2 + 2 \mu \mu_0) + \\
(1 - \alpha_\tau) \tau + \beta_\tau \log \tau + \\
-\frac{1}{2} \left( \text{trace} \left( \Sigma^{-1}_\beta \beta^T \right) + 2 \text{trace} \left( \Sigma^{-1}_\beta \beta_0 \beta^T \right) \right). \quad (A.4)
$$

The prior parameters were taken to be

$$
\mu_0 = 0.000 \\
\sigma_\mu^{-2} = 0.010 \\
\beta_0 = 0.000 \\
\sigma_\beta^{-2} = 0.100 \\
\alpha_\tau = 3.000 \\
\beta_\tau = 3.000.
$$
Under the variational approximation, $\rho_{it}$ is normally distributed given $x_{it}$, with

$$
\rho_{it} = x_{it}^T \beta + u_t \\
\mathbb{E}_q[\rho_{it}] = x_{it}^T \mathbb{E}_q[\beta] + \mathbb{E}_q[u_t] \\
\text{Var}_q(\rho_{it}) = \mathbb{E}_q[\beta^T x_{it} x_{it}^T \beta] - \mathbb{E}_q[\beta] x_{it} x_{it}^T \mathbb{E}_q[\beta] + \text{Var}_q(u_t) \\
= \mathbb{E}_q[\text{tr}(\beta^T x_{it} x_{it}^T \beta)] - \text{tr}(\mathbb{E}_q[\beta] x_{it} x_{it}^T \mathbb{E}_q[\beta]) + \text{Var}_q(u_t) \\
= \text{tr}(x_{it} x_{it}^T \left(\mathbb{E}_q[\beta \beta^T] - \mathbb{E}_q[\beta] \mathbb{E}_q[\beta]^T\right)) + \text{Var}_q(u_t).
$$

We can thus use $n_{MC} = 4$ points of Gauss-Hermite quadrature to numerically estimate $\mathbb{E}_q[\log\left(1 - \frac{e^{\rho_{it}}}{1 + e^{\rho_{it}}}\right)]$:

$$
\rho_{it,s} := \sqrt{\text{Var}_q(\rho_{it})} z_s + \mathbb{E}_q[\rho_{it}] \\
\mathbb{E}_q\left[\log\left(1 - \frac{e^{\rho_{it,s}}}{1 + e^{\rho_{it,s}}}\right)\right] \approx \frac{1}{n_{MC}} \sum_{s=1}^{n_{MC}} \log\left(1 - \frac{e^{\rho_{it,s}}}{1 + e^{\rho_{it,s}}}\right)
$$

We found that increasing the number of points used for the quadrature did not measurably change any of the results. The integration points and weights were calculated using the `numpy.polynomial.hermite` module in Python [Jones et al., 2001].

### A.7 Genomics experiments details

We demonstrate the Python and R code used to run and analyze the experiments on the genomics data in a sequence of Jupyter notebooks. The output of these notebooks are included below, though they are best viewed in their original notebook form. The notebooks, as well as scripts and instructions for reproducing our analysis in its entirety, can be found in the git repository [rjgiordan/AISTATS2019SwissArmyIJ](https://github.com/rjgiordan/AISTATS2019SwissArmyIJ).
fit_model_and_save

February 21, 2019

1 Genomics experiment details.

We demonstrate the infinitesimal jackknife on a publicly available data set of mice gene expression in Shoemaker et al. [2015].

Mice were infected with influenza virus, and gene expression was assessed several times after infection, so the observed data consists of expression levels $y_{gt}$ for genes $g = 1, ..., n_g$ and time points $t = 1, ..., n_t$, where in this case $n_g = 1000$ and $n_t = 42$.

This notebook contains the first of three steps in the analysis. In this notebook, we will first load the data and define a basis with a hyperparameter we wish to select with cross validation. We then describe the two stages of our analysis: a regression stage and a clustering stage. We then save the data for further analysis by the notebooks /x6C/x6F/x61/x64/x5F/x61/x6E/x64/x5F/x72/x65/x66/x69/x74 and /x63/x61/x6C/x63/x75/x6C/x61/x74/x65/x5F/x70/x72/x65/x64/x69/x63/x74/x69/x6F/x6E/x5F/x65/x72/x72/x6F/x72.

This notebook assumes you have already followed the instructions in README.md to install the necessary packages and create the dataset.

2 Step 1: Initial fit.

In [1]: import matplotlib.pyplot as plt
    %matplotlib inline

    import numpy as np
    import inspect
    import os
    import sys
    import time

    np.random.seed(3452453) # nothing special about this seed (we hope)!

In [2]: from aistats2019_ij_paper import regression_mixture_lib as rm_lib
   from aistats2019_ij_paper import regression_lib as reg_lib
   from aistats2019_ij_paper import sensitivity_lib as sens_lib
   from aistats2019_ij_paper import spline_bases_lib
   from aistats2019_ij_paper import transform_regression_lib as trans_reg_lib
   from aistats2019_ij_paper import loading_data_utils
   from aistats2019_ij_paper import saving_gmm_utils
   from aistats2019_ij_paper import mse_utils

    import plot_utils_lib
2.1 The first stage: Regression

2.1.1 Load data

In [3]: # Set bmp_data_repo to be the location of a clone of the repo
    # https://github.com/HelleV/genomic_time_series_bmp
    bmp_data_repo = '././genomic_time_series_bmp'
    y_train, y_test, train_indx, timepoints = loading_data_utils.load_genomics_data(
        bmp_data_repo,
        split_test_train = True,
        train_index_file = '././fits/train_index.npy')

Loading data from: ././genomic_time_series_bmp/data/shoemaker2015reprocessed

In [4]: n_train = np.shape(y_train)[0]
    print('number of genes in training set: \n', n_train)

    n_test = np.shape(y_test)[0]
    print('number of genes in test set: \n', n_test)

    n_genes = n_train + n_test

    test_indx = np.setdiff1d(np.arange(n_genes), train_indx)
    gene_indx = np.concatenate((train_indx, test_indx))

number of genes in training set:
700
number of genes in test set:
300

Each gene $y_g$ has 42 observations. Observations are made at 14 timepoints, with 3 replicates at each timepoint.

In [5]: n_t = len(timepoints)
    n_t_unique = len(np.unique(timepoints))

    print('timepoints: \n', timepoints, '\n')
    print('Distinct timepoints: \n', np.sort(np.unique(timepoints)), '\n')
    print('Number of distinct timepoints: ', n_t_unique)

Distinct timepoints:
[ 0  3  6  9 12 18 24 30 36 48 60 72 120 168]

Number of distinct timepoints: 14
Here is the raw data for a few randomly chosen genes.

```python
In [6]: f, axarr = plt.subplots(2, 3, figsize=(15,8))

gene_idx = np.sort(np.random.choice(n_train, 6))

for i in range(6):
    n = gene_idx[i]
    this_plot = axarr[int(np.floor(i / 3)), i % 3]
    this_plot.plot(timepoints, y_train[n,:].T, 'o', color = 'blue');
    this_plot.set_ylabel('gene expression')
    this_plot.set_xlabel('time')
    this_plot.set_title('gene number {0}'.format(n))

f.tight_layout()
```

2.1.2 Define regressors

We model the time course using cubic B-splines. Let $a$ be the degrees of freedom of the B-splines, and this is the parameter we seek to choose using cross-validation.

For a given degrees of freedom, the B-spline basis is given by an $n_t \times n_x$ matrix $X_{df}$, where the each column of $X_{df}$ is a B-spline basis vector evaluated at the $n_t$ timepoints. Note that $n_x$ increases with increasing degrees of freedom.

Note that we only use B-splines to smooth the first 11 timepoints. For the last three timepoints, $t = 72, 120, 168$, we use indicator functions on each timepoint as three extra basis vectors. In other words, we append to the regressor matrix three columns, where each column is 1 if $t = 72, 120$, or 168, respectively, and 0 otherwise. We do this to avoid numerical issues in the matrix $X^T X$. Because the later timepoints are more spread out, the B-spline basis are close to zero at the later timepoints, leading to matrices close to being singular.
We plot the B-spline matrix for several degrees of freedom below:

In [7]: # Simulate passing arguments in on the command line so that the notebook
   # looks more like those in `cluster_scripts`
   class Args():
       def __init__(self):
           pass

   args = Args()
   args.df = 7
   args.degree = 3
   args.num_components = 10

   In [8]: regressors = spline_bases_lib.get_genomics_spline_basis(
         timepoints, df=args.df, degree=3)

   reg = reg.lib.Regressions(y_train, regressors)

   We plot the B-spline matrix for several degrees of freedom below:

   In [9]: f, axarr = plt.subplots(2, 3, figsize=(15,8))

   i = 0
   for df in [4, 5, 6, 7, 8, 9]:
       _regressors = spline_bases_lib.get_genomics_spline_basis(
         timepoints, exclude_num=3, df=df)

       this_plot = axarr[int(np.floor(i / 3)), i % 3]
       this_plot.plot(timepoints, _regressors);
       this_plot.set_xlabel('time')
       this_plot.set_ylabel('B-spline value')
       this_plot.set_title('B-spline basis when df = {}'.format(df))

       i += 1

   f.tight_layout()
We display the regressor matrix below.

```python
In [10]: plt.matshow(regs.x.T)
plt.ylabel('basis')
plt.xlabel('timepoint and replicate')
plt.title('The (transposed) regressor matrix when df = {};'.format(args.df))
```

With the regressor $X$ defined above, for each gene $g$ we model $P\left(y_g|\beta_g, \sigma_g^2\right) = \mathcal{N}\left(y_g|X\beta_g, \sigma_g^2\right)$. In the second stage, we will want to cluster $\beta_g$ taking into account its uncertainty on each gene. To do this, we wish to estimate the posterior mean $E[\beta_g|y_g]$ and covariance $\text{Cov}(\beta_g|y_g)$ with flat priors for both $\beta_g$ and $\sigma_g^2$.

For each gene, we estimate the posterior with a mean field variational Bayes (MFVB) approximation $q\left(\sigma_g^2, \beta_g, \eta_g\right)$ to the posterior $P\left(\beta_g, \sigma_g^2|y_g\right)$.

In particular, we take $q\left(\sigma_g^2, \beta_g, \eta_g\right) = q^*\left(\sigma_g^2\right) q^*\left(\beta_g\right)$, where $q^*\left(\sigma_g^2\right)$ is a dirac delta function, and we optimize over its a location parameter; $q^*\left(\beta_g\right)$ is a Gaussian density and we optimize over its mean and covariance.
The optimal variational approximation has a closed form that is formally identical to the standard frequentist mean and covariance estimate for linear regression. Explicitly, the optimal variational distribution is,

\[ q^*(\beta_g) = \mathcal{N}(\beta_g \mid (X^T X)^{-1} X^T y_g, \hat{\tau}_g (X^T X)^{-1}) \]

\[ q^*(\sigma^2_g) = \delta(\sigma^2_g = \hat{\tau}_g) \]

where \( \hat{\tau}_g = \frac{1}{n_g - n_{\eta g}} \| y_g - X (X^T X)^{-1} X^T y_g \|_2^2 \).

The advantage of the MVFB construction is that defining \( \hat{\eta}_g \) for \( g = 1, ..., n_g \) satisfies set of \( n_g \) independent M-estimation objectives, allowing us to apply our infinitesimal jackknife results. Specifically, defining \( \theta_{reg} := (\eta_1, ..., \eta_{n_g}) \), we wish to minimize

\[ F_{reg}(\theta_{reg}, \alpha) = \sum_{g=1}^{n_g} KL \left( q \left( \sigma^2_g, \beta_g ; \eta_g \right) \parallel P \left( \beta_g, \sigma^2_g ; y_g \right) \right) \]

\[ = - \sum_{g=1}^{n_g} \mathbb{E}_q \left[ \log P \left( \beta_g, \sigma^2_g ; y_g \right) \right] + \mathbb{E}_q \left[ \log q \left( \beta_g, \sigma^2_g ; \eta_g \right) \right] \]

\[ := \sum_{g=1}^{n_g} F_{reg,g}(\eta_g, \alpha) . \]

Our M-estimator, then, is

\[ \frac{\partial F_{reg}(\theta_{reg}, \alpha)}{\partial \theta_{reg}} = 0. \]

The class \( \text{regr} \) can calculate the optimal variational parameters for each gene. In particular, the variational parameters \( \eta_g \) consist of a variational mean and covariance for \( \beta_g \), as well as a location estimate for \( \sigma^2_g \).

**In [11]:**

```python
reg.time = time.time()
opt_reg_params = regr.get_optimal_regression_params()
reg.time = time.time() - reg_time
print('Regression time: {} seconds'.format(reg.time))
```

Regression time: 0.029132366180419922 seconds

Here are what some of the fits look like. Each regression produces a prediction \( \hat{y}_g := X \mathbb{E}_q [\beta_g] \), plotted with the heavy red line above. The light red are predictions when \( \beta_g \) is drawn from \( q^*(\beta_g) \); the spread of the light red is intended to give a sense of the covariance of \( \beta_g \).

**In [12]:**

```python
f, axarr = plt.subplots(2, 3, figsize=(15,8))
for i in range(6):
    n = gene.index[i]
    this_plot = axarr[int(np.floor(i / 3)), i % 3]
    plot_utils_lib.PlotRegressionLine(
        timepoints, regr, opt_reg_params, n, this_plot=this_plot)
f.tight_layout()
```
We also define and save data for the test regressions, which we will use later to evaluate out-of-sample performance. The training regressions will be saved below with the rest of the fit.

```python
In [13]: reg_test = reg_lib.Regressions(y_test, regressors)
    test_regression_outfile = '../fits/test_regressions.json'
    with open(test_regression_outfile, 'w') as outfile:
        outfile.write(regs_test.to_json())
```

### 2.2 The second stage: fit a mixture model.

#### 2.2.1 Transform the parameters before clustering

We are interested in the pattern of gene expression, not the absolute level, so we wish to cluster $\hat{y}_g - \bar{\hat{y}}_g$, where $\bar{\hat{y}}_g$ is the average over time points. Noting that the $n_t \times n_t$ matrix $\text{Cov}_q(\hat{y}_g - \bar{\hat{y}}_g)$ is rank-deficient because we have subtracted the mean, the final step is to rotate $\hat{y}_g - \bar{\hat{y}}_g$ into a basis where the zero eigenvector is a principle axis and then drop that component.

Call these transformed regression coefficients $\gamma_g$ and observe that $\text{Cov}_q(\gamma_g)$ has a closed form and is full-rank. It is these $\gamma_g$s that we will cluster in the second stage.

We briefly note that the re-centering operation could have been equivalently achieved by making a constant one of the regressors. We chose this implementation because it also allows the user to cluster more complex, non-linear transformations of the regression coefficients, though we leave this extension for future work.

We note that the transformations described in this section are done automatically in the GMM class. We are only calculating these transformations here for exposition.

```python
In [14]: # Get the matrix that does the transformation.
    transform_mat, unrotate_transform_mat = \
            trans_reg_lib.get_reversible_predict_and_demean_matrix(regs.x)
    trans_obs_dim = transform_mat.shape[0]
```
If $T$ is the matrix that effects the transformation, then

$$E_q[\gamma] = TE_q[\beta]\quad\text{Cov}_q(\gamma) = TCov_q(\beta)T^T$$

The transformed parameters are also regression parameters, just in a different space.

We now visualize the transformed coefficients and their uncertainty.

```python
In [18]: # Apply the transformation
transformed_reg_params = \
   trans_reg_lib.multiply_regression_by_matrix(\n   opt_reg_params, transform_mat)

We now visualize the transformed coefficients and their uncertainty.

In [16]: f, axarr = plt.subplots(2, 3, figsize=(15,8))

transformed_beta = transformed_reg_params[‘beta_mean’]
transformed_beta_info = transformed_reg_params[‘beta_info’]

for i in range(8):
    n = gene_index[i]
    this_plot = axarr[int(np.floor(i / 3)), i % 3]
    this_plot.plot(transformed_beta[n, :], color = ‘red’);
    this_plot.set_ylabel(‘transformed coefficient’)\n    this_plot.set_xlabel(‘index’)
    this_plot.set_title(‘gene number {i}’.format(n))

# draw from the variational distribution, to plot uncertainties
for j in range(30):
    transformed_beta_draw = np.random.multivariate_normal(\n        transformed_beta[n, :], \n        np.linalg.inv(transformed_beta_info[n]))

f.tight_layout()
```
The heavy red lines are the means of the transformed regression coefficients; shaded lines are draws from the variational distribution.

It is these transformed coefficients, $\gamma_g$, that we cluster in the second stage.

2.2.2 Estimate an optimal clustering.

We now define a clustering problem for the $\gamma_g$. Let $n_k$ be the number of clusters, and $\mu_1, ..., \mu_{n_k}$ be the cluster centers. Also let $z_{gk}$ be the binary indicator for the $g$th gene belonging to cluster $k$. We then define the following generative model

\[
P(\pi) = \text{Dirichlet}(\omega) \\
P(\mu) = \mathcal{N}(\mu_0|0, \Sigma_0) \quad \text{for} \quad k = 1, ..., n_k \\
P(z_{gk} = 1|\pi_k) = \pi_k \quad \text{for} \quad k = 1, ..., n_k; \ n = 1, ..., n_g \\
P(\gamma_g|z_{gk} = 1, \mu_k, \eta_g) = \mathcal{N}(\gamma_g|\mu_k, \text{Cov}_q(\gamma_g) + \epsilon I_{n-1}) \quad \text{for} \quad k = 1, ..., n_k; \ n = 1, ..., n_g.
\]

where $\epsilon$ is a small regularization parameter, which helped our optimization produce more stable results.

We will estimate the clustering using the maximum a posteriori (MAP) estimator of $\theta_{\text{clust}} := (\mu, \pi)$. This defines an optimization objective that we seek to minimize:

\[
F_{\text{clust}}(\theta_{\text{clust}}, \theta_{\text{reg}}) = -\sum_{g=1}^{n_g} F_{q_g} \left\{ \log P(\gamma_g|\eta_g, \mu, \pi, z_g) - \log P(z_g|\pi) \right\} - \log P(\mu) - \log P(\pi)
\]

which, for every value of $\theta_{\text{reg}}$, we expect to satisfy

\[
\frac{\partial F_{\text{clust}}(\theta_{\text{clust}}, \theta_{\text{reg}})}{\partial \theta_{\text{clust}}} = 0.
\]
Note that \( \theta_{\text{clust}} \) involves only the “global” parameters \( \mu \) and \( \pi \). We did take a variational distribution for the \( zg_{ks} \), represented by independent Bernoulli distribution, but the optimal \( q^*_z \) can be written as a function of \( \mu \) and \( \pi \). Hence, our optimization objective only involves these global parameters.

In [17]: # Define prior parameters.
    num_components = args.num_components
    epsilon = 0.1
    loc_prior_info_scalar = 1e-5
    
    trans_obs_dim = regs.x.shape[1] - 1
    prior_params = \
        rm_lib.get_base_prior_params(trans_obs_dim, num_components)
    prior_params['probs_alpha'][:, :] = 1
    prior_params['centroid_prior_info'] = loc_prior_info_scalar * np.eye(trans

In [18]: gmm = rm_lib.GMM(args.num_components,
    prior_params, regs, opt_reg_params,
    inflate_coef_cov=None,
    cov_regularization=epsilon)

In our experiment, the number of clusters \( n_k \) was chosen to be 10. We set \( \omega \) to be the ones vector of length \( n_k \). The prior info for the cluster centers \( \Sigma_0 \) is \( 1e-05 \times I \). \( \epsilon \) was set to be 0.1.

Let us examine the optimization objective. First, we’ll inspect the likelihood terms. What follows is the likelihood given that gene \( g \) belongs to cluster \( k \).

In [19]: print(inspect.getsource(rm_lib.get_log_lk_nk))
    
def get_log_lk_nk(centroids, probs, x, x_infos):
        log_lk = \
            -0.5 * (-2 * np.einsum('ni,kj,nij->nk', x, centroids, x_infos) +
                  np.einsum('k-kij,nij->nk', centroids, centroids, x_infos))
        log_probs = np.log(probs[0, :])
        log_lk_by_nk = log_lk + log_probs.T
        return log_lk_by_nk

We can then optimize for \( q^*_z \), which can be parametrized by its mean \( E_{q^*_z}[z] \). We note that this update has a closed form given \( \theta_{\text{clust}} \), so there is no need to solve an optimization problem to find \( q^*_z \). We additionally note that we do not use the EM algorithm, which we found to have exhibit extremely poor convergence rates. Rather, we set \( q^*_z \) to its optimal value given \( \theta_{\text{clust}} \) and return the objective as a function of \( \theta_{\text{clust}} \) alone, allowing the use of more general and higher-quality optimization routines.

In [20]: print(inspect.getsource(rm_lib.get_e_z))
With the optimal parameters for $z_{nk}$, we combine the likelihood term with the prior and entropy terms.

```python
def get_e_z(log_lik_by_nk):
    log_const = paragami.simplex_patterns.logsumexp(log_lik_by_nk, axis=1)
    e_z = np.exp(log_lik_by_nk - log_const)
    return e_z
```

This objective function is wrapped in the GMM class method get_params_kl.

```python
def wrap_get_loglik_terms(gmm_params, transformed_reg_params):
    log_lik_by_nk = get_log_lik_nk(  
        centroids=gmm_params['centroids'],
        probs=gmm_params['probs'],
        x=transformed_reg_params['beta_mean'],
        x_infos=transformed_reg_params['beta_info'])
    e_z = get_e_z(log_lik_by_nk)
    return log_lik_by_nk, e_z
```

```python
def wrap_get_kl(gmm_params, transformed_reg_params, prior_params):
    log_lik_by_nk, e_z = \
    wrap_get_loglik_terms(gmm_params, transformed_reg_params)
    log_prior = get_log_prior(  
        gmm_params['centroids'], gmm_params['probs'], prior_params)
    return get_kl(log_lik_by_nk, e_z, log_prior)
```

2.2.3 Optimization

For optimization we make extensive use of the autograd library for automatic differentiation and the paragami library for parameter packing and sparse Hessians. These packages' details are beyond the scope of the current notebook.
First, we do a k-means initialization.

```python
In [23]: print('Running k-means init.')
    init_gmm_params = \
        rm_lib.kmeans_init(gmm.transformed_reg_params, \
                            gmm.num_components, 50)
    print('Done.')
    init_x = gmm.gmm_params_pattern.flatten(init_gmm_params, free=True)
```

Running k-means init.

Done.

We note that the match between “exact” cross-validation (removing time points and re-optimizing) and the IJ was considerably improved by using a high-quality second-order optimization method. In particular, for these experiments, we employed the Newton conjugate-gradient trust region method (Chapter 7.1 of Wright et al [1999]) as implemented by the method trust-reg in scipy.optimize, preconditioned by the Cholesky decomposition of an inverse Hessian calculated at an initial approximate optimum.

We found that first-order or quasi-Newton methods (such as BFGS) often got stuck or terminated at points with fairly large gradients. At such points our method does not apply in theory nor, we found, very well in practice.

The inverse Hessian used for the preconditioner was with respect to the clustering parameters only and so could be calculated quickly, in contrast to the $H_1$ matrix used for the IJ, which includes the regression parameters as well.

First, run with a low tolerance to get a point at which to evaluate an initial preconditioner.

```python
In [24]: gmm.conditioned_obj.reset() # Reset the logging and iteration count.
    gmm.conditioned_obj.set_print_every(1)
    opt_time = time.time()
    gmm_opt, init_x2 = gmm.optimize(init_x, gtol=1e-2)
    opt_time = time.time() - opt_time
```

Iter 0: $f = -159.11834165$
Iter 1: $f = -159.67926278$
Iter 2: $f = -159.97782885$
Iter 3: $f = -160.15878120$
Iter 4: $f = -159.59447036$
Iter 5: $f = -160.19209687$
Iter 6: $f = -160.27259154$
Iter 7: $f = -160.29485553$
Iter 8: $f = -160.33460856$
Iter 9: $f = -160.34154288$
Iter 10: $f = -160.32382096$
Iter 11: $f = -160.34447865$
Iter 12: $f = -160.34634639$
Iter 13: $f = -160.34692896$
Next, set the preconditioner using the square root inverse Hessian at the point init_x2.

```python
In [26]: tic = time.time()
    h_cond = gmm.update_preconditioner(init_x2)
    opt_time = time.time() - tic
```

The method optimize fully repeats this process of optimizing and re-calculating the preconditioner until the optimal point does not change.

```python
In [26]: gmm.conditioned_obj.reset()
    tic = time.time()
    gmm_opt, gmm_opt_x = gmm.optimize_fully(
        init_x2, verbose=True)
    opt_time = time.time() - tic
    print("Optimization time: {} seconds").format(opt_time))
```

Preconditioned iteration 1
Running preconditioned optimization.
Iter 0: f = -160.34692898
Iter 1: f = -160.34694250
Iter 2: f = -160.34694250
Preconditioned iteration 2
Getting Hessian and preconditioner.
Running preconditioned optimization.
Iter 3: f = -160.34694250
Iter 4: f = -160.34694250
Converged.
Optimization time: 8.438910484313965 seconds

ParaGAM patterns allow conversion between unconstrained vectors and dictionaries of parameter values. After "folding" the optimal gmm_opt_x, opt_gmm_params contains a dictionary of optimal cluster centroids and cluster probabilities.

```python
In [27]: opt_gmm_params = gmm_gmm_params_pattern.fold(gmm_opt_x, free=True)
    print(opt_gmm_params.keys())
    print(np.sort(opt_gmm_params['probs']))
```

```
{'centroids', 'probs'}
[[0.0156706 0.04016882 0.06955236 0.07427946 0.09373695 0.0947442
  0.09853288 0.12626624 0.15739176 0.23165127]]
```

Each gene’s regression line has an inferred cluster membership given by \( E_{k} | z \), and an expected posterior centroid given by \( \sum_{k} E_{k} | z_{k} | \mu_{k} \). This expected posterior centroid can be untransformed to give a prediction for the observation.

It is the difference between this prediction line — which is a function of the clustering — and the actual data that we consider to be the “error” of the model.
2.2.4 Calculating $H_1$ for the IJ

We seek to choose the degrees of freedom $\alpha$ for the B-splines using cross-validation. We leave out one or more timepoints, and fit using only the remaining timepoints. We then estimate the test error by predicting the value of the genes at the held out timepoints.

To do this, we define time weights $w_t$ by observing that, for each $g$, the term $E_q[\log P(\beta_g, \sigma^2_g | y_g)]$ decomposes into a sum over time points:

$$ F_{\text{reg}, g}(\eta_g, \alpha, w) := -\sum_{t=1}^{m_g} w_t \left( -\frac{1}{2} \sigma^2_g (y_{g,t} - (X\beta_g)_t)^2 - \frac{1}{2} \log \sigma^2_g \right) + E_q \left[ \log q(\beta_g, \sigma^2_g | \eta_g) \right]. $$

We naturally define $F_{\text{reg}}(\theta_{\text{reg}}, \alpha, w) := \sum_{g=1}^{m_g} F_{\text{reg}, g}(\eta_g, \alpha, w)$. 

In [28]:
```python
gmm_pred = mse_utils.get_predictions(gmm, opt_gmm_params, opt_reg_params)

f, axarr = plt.subplots(2, 3, figsize=(15,8))

for i in range(6):
    n = gene_index[i]
    this_plot = axarr[int(np.floor(i / 3)), i % 3]
    plot_utils_lib.PlotRegressionLine(
        timepoints, regs, opt_reg_params, n, this_plot=this_plot)
    plot_utils_lib.PlotPredictionLine(
        timepoints, regs, gmm_pred, n, this_plot=this_plot)

f.tight_layout()
```
By defining $\theta = (\theta_{\text{clust}}, \theta_{\text{reg}})$, we then have an M-estimator

$$G(\theta, w, \alpha) := \begin{bmatrix} \frac{\partial F_{\text{reg}}(\theta_{\text{reg}}, w, \alpha)}{\partial \theta_{\text{reg}}} \\ \frac{\partial F_{\text{clust}}(\theta_{\text{clust}}, \theta_{\text{reg}})}{\partial \theta_{\text{clust}}} \end{bmatrix} = 0.$$  

We can then apply the IJ to approximate the leaving out of various timepoints.

Note that what we call the “Hessian” for this two-step procedure is not really a Hessian, as it is not symmetric. More precisely, it is the Jacobian of $G$, or what we defined as $H_1$ in the text.

Calculating $H_1$ is the most time-consuming part of the infinitesimal jackknife, since the $H_1$ matrix is quite large (though sparse). However, once $H_1$ is computed, calculating each $\theta_I(w)$ is extremely fast.

$H_1$ can be computed in blocks:

$$H_1 = \begin{pmatrix} \nabla^2_{\theta_{\text{reg}}} F_{\text{reg}} & 0 \\ \nabla_{\theta_{\text{reg}}} \nabla_{\theta_{\text{clust}}} F_{\text{clust}} & \nabla^2_{\theta_{\text{clust}}} F_{\text{clust}} \end{pmatrix}$$

The code refers to $\nabla^2_{\theta_{\text{clust}}} F_{\text{clust}}$ as the “GMM Hessian”. It refers to $\nabla_{\theta_{\text{reg}}} \nabla_{\theta_{\text{clust}}} F_{\text{clust}}$ as the “cross Hessian”. And it refers to $\nabla^2_{\theta_{\text{reg}}} F_{\text{reg}}$ as the “regression Hessian”, which itself is block diagonal, with each block an observation. Due to details of the implementation of block sparse Hessians using forward mode automatic differentiation in the class `vittles.SparseBlockHessian`, the code below confusingly refers to each regression parameter as a “block”.

When the `FitDerivatives` class is initialized, it calculates these blocks separately and stacks them into the attribute `full_hess`, which is a sparse matrix representing $H_1$.

```python
In [29]: # Even though $H_1$ is not a Hessian, by force of habit we call the time to compute it "hess_time".
    hess_time = time.time()
    fit_derivs = sens_lib.FitDerivatives(
        opt_gmm_params, opt_reg_params,
        gmm_gmm_params_pattern, regs.reg_params_pattern,
        gmm=gmm, regs=regs,
        print_every=10)
    hess_time = time.time() - hess_time
    print('Total hessian time: {} seconds'.format(hess_time))
```

Initializing `FitDerivatives`.

Getting $t$ Jacobian.

Getting full Hessian.

Getting GMM Hessian...

GMM Hessian time: 2.1917014122009277

Getting cross Hessian...

Cross Hessian time: 34.25235605239868

Getting regression Hessian...

Block index 0 of 66.

Block index 10 of 66.

Block index 20 of 66.

Block index 30 of 66.
2.2.5 Save results as a compressed file.

The results, including $H_1$, are now saved. To calculate the exact CV, these results (including the preconditioner) will be loaded and the model will be refit with timepoints left out. To calculate the IJ, the same results will be loaded and $H_1$ will be used to calculate the IJ.

In [30]:
extra_metadata = dict()
extra_metadata['opt_time'] = opt_time
extra_metadata['reg_time'] = reg_time
extra_metadata['hess_time'] = hess_time
extra_metadata['df'] = args.df
extra_metadata['degree'] = args.degree

npz_outfile = '../fits/initial_fit.npz'
saving_gmm_utils.save_initial_optimum(
    npz_outfile,
gmm=gmm,
reg=reg,
timepoints=timepoints,
fit_deriv=fit_derivs,
extra_metadata=extra_metadata)

2.2.6 Bibliography


Step 2: Refit.

In this notebook, we calculate the parameters used for exact CV by refitting the model initially fit in step one, the notebook `fit_model_and_save`.

For expository purposes this notebook calculates the refit for only one weight vector. To compute exact CV, one would perform the corresponding computation for all leave-k-out weight vectors.

```python
In [1]: from copy import deepcopy
import inspect
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import sys
import time

np.random.seed(3452453)

import paragami

from aistats2019_ij_paper import regression_mixture_lib as rm_lib
from aistats2019_ij_paper import saving_gmm_utils
from aistats2019_ij_paper import mse_utils

import plot-utils_lib

In [2]: # Load the initial fit.
   # This file was produced by the notebook `fit_model_and_save`
   initial_fit_infile = 'fits/initial_fit.npz'
   full_fit, gmm, rejs, metadata = 
       saving_gmm_utils.load_initial_optimum(initial_fit_infile)
   timepoints = metadata['timepoints']

Initializing FitDerivatives.
Using provided t_jac.
Using provided full_hess.
First, choose some timepoints to leave out.

In [3]: # Simulate passing arguments in on the command line.
class Args():
    def __init__(self):
        pass

    args = Args()
    args.num_times = 1
    args.which_comb = 1
    args.max_num_timepoints = 7

The number of points left out (that is, \( k \)) is given by \texttt{num\_times}, which is 1. The largest time-
point we leave out is given by \texttt{max\_num\_timepoints}, which is 7. Because later timepoints are not
affected by the smoothing, there is no reason to leave them out.

There are a certain number of ways to leave \( k \) out of 7 timepoints, and \texttt{which\_comb} chooses
one of them in the order given by the function \texttt{itertools.combinations}. Of course, when \( k = 1 \),
\texttt{which\_comb} simply chooses which timepoint to leave out. \texttt{mse\_utils.get\_indexed\_combination}
maps \texttt{which\_comb} to particular timepoints in a consistent way.

Full exact CV would run this script for all 7 choose \( k \) values of \texttt{which\_comb}.
Because we have repeated measurements at each timepoint, leaving out a single timepoint will
correspond to leaving out multiple row of the observation matrix. Those rows are determined
by \texttt{mse\_utils.get\_time\_weight}, which also returns a weight vector setting these observations'
weights to zero.

In [4]: lo\_inds = mse\_utils.get\_indexed\_combination(
    num\_times=\texttt{args.num\_times}, which\_comb=\texttt{args.which\_comb},
    max\_num\_timepoints=\texttt{args.max\_num\_timepoints})
    new\_time\_w, full\_lo\_inds = mse\_utils.get\_time\_weight(lo\_inds, timepoints)

    print('Left out timepoint: {}\'.format(lo\_inds))
    print('Left out observations: {}\'.format(full\_lo\_inds))
    print('Leave-k-out weights: {}\'.format(new\_time\_w))

Left out timepoint: [1]
Left out observations: [3 4 5]

We now re-optimize with the new weights.

Note that we could either start the optimization at the initial optimum (a “warm start”) or do
a fresh start from k-means. A fresh start is more time consuming but a more stringent test for the
accuracy of the IJ. We calculate both, but report results from the fresh start in the paper. In the
notebook \texttt{examine\_and\_save\_results}, you can choose to examine either set of results.
Here, for consistency with the paper, we re-initialize with k-means.

In [5]: regs\_time\_w = deepcopy(new\_time\_w)
    reg\_params\_w = regs.get\_optimal\_regression\_params()
```python
# Set regression parameters

gmm.set_regression_params(reg_params_w)

# Initialize parameters

init_gmm_params = 
    rm_lib.kmeans_init(gmm.transformed_reg_params,
        gmm.num_components, 50)

init_x = gmm.gmm_params_pattern.flatten(init_gmm_params, free=True)

opt_time = time.time()

gmm_opt, init_x2 = gmm.optimize(init_x, gtol=1e-2)

print('Updating preconditioner...')

kl_hess = gmm.update_preconditioner(init_x2)

print('Running preconditioned optimization...')

# Conditioned optimization

reopt, gmm_params_free_w = gmm.optimize_fully(init_x2, verbose=True)

print(gmm_opt.message)

opt_time = time.time() - opt_time

print('Refit time: {} seconds'.format(opt_time))

# Iterations and cost function values

Iter 0: f = -153.38003431
Iter 1: f = -152.49436715
Iter 2: f = -153.69147895
Iter 3: f = -153.8377915
Iter 4: f = -154.0239812
Iter 5: f = -153.4133391
Iter 6: f = -154.10396120
Iter 7: f = -154.14366282
Iter 8: f = -154.14261201
Iter 9: f = -154.16417745
Iter 10: f = -154.18307547
Iter 11: f = -154.20711481
Iter 12: f = -154.2218064
Iter 13: f = -154.27402715
Iter 14: f = -154.28739474
Iter 15: f = -154.33849929
Iter 16: f = -154.03580241
Iter 17: f = -154.35421130
Iter 18: f = -154.36910489
Iter 19: f = -154.36872488
Iter 20: f = -154.37238982
Iter 21: f = -154.37722095
Iter 22: f = -154.38186985
Iter 23: f = -154.38410992

Updating preconditioner...

Running preconditioned optimization...

Preconditioned iteration 1
```
APPENDIX A. SUPPLEMENTARY MODEL DETAILS

Running preconditioned optimization.
Iter 0: f = -154.38410992
Iter 1: f = -154.38423176
Iter 2: f = -154.38584092
Iter 3: f = -154.21889674
Iter 4: f = -154.42200228
Iter 5: f = -154.39603234
Iter 6: f = -154.39957947
Iter 7: f = -154.41374585
Iter 8: f = -154.43397491
Iter 9: f = -154.43484046
Iter 10: f = -154.43498416
Iter 11: f = -154.43498416
Preconditioned iteration 2
Getting Hessian and preconditioner.
Running preconditioned optimization.
Iter 12: f = -154.43498416
Iter 13: f = -154.43498416
Converged.
Optimization terminated successfully.
Refit time: 14.3515647315979 seconds

We now save the results.

In [6]: gmm_params_w = \
   full_fit.comb_params_pattern['mix'].fold(\n   gmm_params_free_w, free=True)\n   refit_comb_params = {\n   'mix': gmm_params_w,\n   'reg': reg_params_w }\n   refit_comb_params_free = \n   full_fit.comb_params_pattern.flatten(refit_comb_params, free=True)

In [7]: save_filename = \n   './fits/refit_num_times{}__which_comb{}.npz'.format(\n   args.num_times, args.which_comb)\n   print('Saving to {}.format(save_filename))\n   saving_gmm_utils.save_refit(\n   outfile=save_filename,\n   comb_params_free=refit_comb_params_free,\n   comb_params_pattern=full_fit.comb_params_pattern,\n   initial_fit_infile=initial_fit_infile,\n   time_w=new_time_w,\n   lo_ind=lo_ind,\n   full_lo_ind=full_lo_ind)\n
Saving to ./fits/refit_num_times1__which_comb1.npz
calculate_prediction_errors

February 21, 2019

1 Step 3: Calculate the IJ and prediction errors.

In this notebook, for a single weight vector, we calculate the IJ itself as well as the prediction errors for exact CV and IJ. This notebook uses the output of the notebooks load_and_refit and fit_model_and_save.

In [1]: import numpy as np
import paragami
import vittles
import scipy as sp
from scipy import sparse
import time

import seaborn as sns
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

np.random.seed(3452453)

from aistats2019_iij_paper import regression_lib as reg_lib
from aistats2019_iij_paper import sensitivity_lib as sens_lib
from aistats2019_iij_paper import saving_gmm_utils
from aistats2019_iij_paper import mse_utils

import plot_utils_lib

In [2]: # Simulate passing arguments in on the command line.
class Args():
    def __init__(self):
        pass

args = Args()
args.num_times = 1
args.which_comb = 1
args.max_num_timepoints = 7
APPENDIX A. SUPPLEMENTARY MODEL DETAILS

In [3]: # Load the original fit.

    print('Loading original fit.')
    initial_fit_infile = 'fits/initial_fit.npz'
    full_fit, gmm, regs, initial_metadata = \
        saving_gmm._load_initial_optimum(initial_fit_infile)
    opt_comb_params = full_fit.get_comb_params()

Loading original fit.
Initializing FitDerivatives.
Using provided t_jac.
Using provided full_hess.

In [4]: # Load the test data

    test_regression_infile = 'fits/test_regressions.json'
    with open(test_regression_infile) as infile:
        regs_test = reg_lib.Regressions.from_json(infile.read())

    # Load a refit as specified by `args`.

    refit_filename = \
        'fits/refit_{}.{}which_comb{}.npz'.format(\n            args.num_times, args.which_comb)
    comb_params_free_refit, comb_params_pattern_refit, refit_metadata = \
        saving_gmm._load_refit(refit_filename)
    time_w = refit_metadata['time_w']
    lo_inds = refit_metadata['lo_inds']
    full_lo inds = refit_metadata['full_lo inds']

    assert(comb_params_pattern_refit == full_fit.comb_params_pattern)
    comb_params_refit = comb_params_pattern_refit.fold(\n        comb_params_free_refit, free=True)

    time_w = refit_metadata['time_w']
    lo_inds = refit_metadata['lo inds']
    full_lo inds = refit_metadata['full_lo inds']

The objects named combparams refer to both the regression and clustering parameters. The
name free refers to the unconstrained flat value for the parameters as calculated by paragami.

In [5]: print('Regression pattern: ',
        comb_params_pattern_refit['reg'])
print('Clustering pattern: ',
    comb_params_pattern_refit['mix'])

Regression pattern:  OrderedDict:
[beta_mean] = NumericArrayPattern (700, 10) (lb=-inf, ub=inf)
[beta_info] = PatternArray (700,) of PDMatrix 10x10 (diag_lb = 0.0)
[y_info] = NumericArrayPattern (700,) (lb=0.0, ub=inf)
Clustering pattern:  OrderedDict:
[centroids] = NumericArrayPattern (10, 9) (lb=-inf, ub=inf)
[probs] = SimplexArrayPattern (1,) of 10-d simplices

1.0.1 Calculate the infinitesimal jackknife.

The vittles package makes it easy to calculate linear approximations to the sensitivity of M-estimators to hyperparameters, of which the IJ is a special case. Here, the HyperparameterSensitivityLinearApproximation uses the sparse value of $H_1$ calculated earlier. Note that $H_1$ is factorized during the initialization of weight_sens, and that it takes relatively little time.

In [6]: # Note that if you don't cast the jacobian to a numpy array from
    # a numpy matrix, the output is a 2d-array, causing confusion later.
    weight_sens = vittles.HyperparameterSensitivityLinearApproximation(
        objective_fun=lambda: 0,
        opt_par_value=full_fit.comb_params_free,
        hyper_par_value=regs.time_w,
        hessian_at_opt=np.sparse.csc_matrix(full_fit.full_hess),
        cross_hess_at_opt=np.array(full_fit.t_jac.todense()))

We now use the weight_sens object to approximate the “free” value of the combined parameters at time_w. The IJ operates in unconstrained space, so we use parasigma to fold the unconstrained vector back into a dictionary of parameters.

In [7]: # Get the infinitesimal jackknife for the refit weight vector.
    lr_time = time.time()
    comb_params_free_lin = \
        weight_sens.predict_opt_par_from_hyper_par(time_w)
    lr_time = time.time() - lr_time
    print('Infinitesimal jackknife time: {:.4f}'.format(lr_time))

    comb_params_lin = full_fit.comb_params_pattern.fold

Infinitesimal jackknife time: 0.0011603832244873047

1.0.2 Calculate various prediction errors.

Recall that the prediction error is the difference between the data and the posterior expected cluster centroid for a particular gene. Let us consider the original optimal clustering parameters,
To get the test set error on gene $g$ for these parameters, we need to do the following steps:

1. Run the regression for gene $g$ in the test set.
2. Classify the regression, calculating $E_q[z_g]$. This is a function of the clustering parameters and the regression line for gene $g$.
3. Calculate the expected posterior cluster centroid for gene $g$, which is $\mu^*_g = \sum_k E_q[z_g] \mu_k$.
4. Because the transformation discards the mean information, compare the de-meaned data to the estimated centroid: $\text{error}_{gt} = (y_{gt} - \frac{1}{T} \sum_{t=1}^{T} y_{gt'} - \mu^*_g)$.

Note that step one could re-run the regression either with the original weights or the new weights. We found that this decision does not matter qualitatively. Here and in the paper, we simply classify the original regression, but the notebook `examine_and_save_results` can produce results for other original and re-weighted regressions.

We will examine prediction error on the time points that are left out, that is, for observations in `full_lo_inds`.

In [8]:

```
# Get the training set error on the full data.
train_error = mse_utils.get_lo_err_folded(
    opt_comb_params,
    keep_inds=full_lo_inds,
    mse_reg=reg,
    mse_reg_params=opt_comb_params['reg'],
    gmm=gmm)

# Get the optimal test set regressions.
reg_params_test = reg_test.get_optimal_regression_params()

# Get the test error for the original fit.
orig_test_error = mse_utils.get_lo_err_folded(
    opt_comb_params,
    keep_inds=full_lo_inds,
    mse_reg=reg_test,
    mse_reg_params=reg_params_test,
    gmm=gmm)

orig_pred = mse_utils.get_predictions(
    gmm, opt_comb_params['mix'], reg_params_test)

# Get the test error for the CV refit.
CV_error = mse_utils.get_lo_err_folded(
    comb_params_refit,
    keep_inds=full_lo_inds,
    mse_reg=reg_test,
    mse_reg_params=reg_params_test,
    gmm=gmm)
```
1.0.3 Selected results.

We now make a cursory comparison of the results. For a more detailed analysis, including the results that went into the paper, see the notebook `examine_and_save_results`.

In [9]: cv_excess_error = cv_error - orig_test_error
   ij_excess_error = ij_error - orig_test_error

    def GetColDf(col):
        return pd.DataFrame(
            {'cv_error': cv_error[:, col],
             'cv_excess': cv_excess_error[:, col],
             'ij_error': ij_error[:, col],
             'ij_excess': ij_excess_error[:, col],
             'col': col})

    result = pd.concat([GetColDf(col) for col in range(len(full_lo_indxs))])

If we simply look at the point-by-point error, CV and IJ are highly correlated.

In [10]: sns.jointplot(x='cv_error', y='ij_error', data=result);
However, this is because the error in each point is dominated by the error at the original optimum. To meaningfully compare the IJ to CV, we should compare the difference between the IJ and CV error and the error at the original optimum. The distribution of these “difference-in-difference” errors is shown in the next plot.

Some clear outliers can be seen. However, note that, in this case, overplotting makes IJ looks worse than it is – in the histograms you can see that most differences are very small.

In [11]: sns.jointplot(x='cv_excess', y='ij_excess', data=result);
As you might expect from a linear approximation, the IJ does the worst when the predicted change for CV is large.

In [12]: misfit = np.max(np.abs(cv_excess_error - ij_excess_error), axis=1)
   abs_cv_excess_error = np.max(np.abs(cv_excess_error), axis=1)

   sns.jointplot(abs_cv_excess_error, misfit)

Out[12]: <seaborn.axisgrid.JointGrid at 0x7f3f74fe1908>
Finally, we visualize some of the genes where IJ badly misestimates the CV error. Clearly, in these cases, re-fitting with the left-out points (shown with large dots) produced large changes that the IJ did not capture. In general, it appears that the IJ errs relative to CV by not moving far enough from the original optimum.

Despite the poor fit on these extreme genes, we stress that most genes exhibited small changes in both CV and IJ. For these genes, IJ performs well enough to capture salient aspects of the estimated out-of-sample error. For more detailed analysis of this point, see the notebook `examine_and_save_results`.

```python
In [13]: timepoints = initial_metadata['timepoints']
   timepoints_stretch = np.sqrt(timepoints)

def PlotGenePredictions(gene_ind):
   - figs = plt.subplots(1, 3, figsize=(15, 6))
```
for i in range(3):
    np.random.seed(42)
    plot_utils_lib.PlotRegressionLine(  
        timepoints_stretch, rega_test, rega_params_test, gene_ind,
        figs[i].plot(timepoints_stretch[full_lo_inds],
                     rega_test.y[gene_ind, full_lo_inds], 'o',
    
    plot_utils_lib.PlotPredictionLine(  
        timepoints_stretch, rega_test, orig_pred, gene_ind,
        figs[0].set_title('Gene {} original fit'.format(gene_ind))
    
    plot_utils_lib.PlotPredictionLine(  
        timepoints_stretch, rega_test, ij_pred, gene_ind,
        figs[1].set_title('Gene {} IJ fit'.format(gene_ind))
    
    plot_utils_lib.PlotPredictionLine(  
        timepoints_stretch, rega_test, cv_pred, gene_ind,
        figs[2].set_title('Gene {} CV fit'.format(gene_ind))

In [14]: worst_fits = np.argsort(-i * misfit)

for gene in worst_fits[0:5]:
    PlotGenePredictions(gene)
examine_and_save_results

February 21, 2019

1 Detailed analysis of results.

This notebook loads the output of the scripts in the directory cluster_scripts (particularly, the final script, run_slurm_pred_error.py). It produces the Rdata file that is used for the graphs in the paper as well as a number of supplemental analyses.

```r
In [1]: library(tidyverse)
    library(gridExtra)
    library(repr)  # For setting plot sizes
    source("load_python_data_lib.R")
    py_main <- InitializePython()

Attaching packages  tidyverse 1.2.1
ggplot2 3.1.0  purrr 0.2.5
tibble 1.4.2  dplyr 0.7.8
tidy 0.8.1  stringr 1.3.1
readr 1.1.1  forcats 0.3.0

Conflicts tidyverse_conflicts()
dplyr::filter() masks stats::filter()
dplyr::lag()    masks stats::lag()

Attaching package: gridExtra

The following object is masked from package:dplyr:

    combine

Attaching package: reshape2

The following object is masked from package:tidyr:

    smdh
# Appendix A. Supplementary Model Details

```r
#init_method <- "warm"

# Choose whether or not to re-run the regressions
use_rereg <- FALSE # This is the choice for the paper.
#use_rereg <- TRUE
```

1.0.1 Load the saved data for all dfs and k

```r
In [3]: dfs <- list()
metadata_dfs <- list()

for (lo_num_times in 1:3) {
    cat("lo_num_times ", lo_num_times)
    for (df in 4:8) {
        cat(".
        load_res <- LoadPredictionError(df, lo_num_times, init_method)
        this_refit_err_df <- load_res$refit_err_df
data_metadata_df <- load_res$metadata_df
        this_refit_err_melt <- MeltErrorColumns(this_refit_err_df)
        dfs[[length(dfs) + 1]] <- this_refit_err_melt
        metadata_dfs[[length(metadata_dfs) + 1]] <- this_metadata_df
        }
        cat("\n"")
dfs <- do.call(bind_rows, dfs)
metadata_df <- do.call(bind_rows, metadata_dfs)

lo_num_times 1...
lo_num_times 2...
lo_num_times 3...
Done.
```

1.0.2 Metadata (timing, parameter dimensions)

Make a tidy dataframe with the metadata. The parameter length, Hessian time, and initial optimization time are all reported in the text of the paper. Their values will be derived from this dataframe in knitr.

```r
In [4]: metadata_df <-
   metadata_df %>%
   mutate(lr_hess_time=total_lr_time + initial_hess_time,
            total_lr_time=total_lr_time / num_comb,
            total_refit_time=total_refit_time / num_comb,
```
APPENDIX A. SUPPLEMENTARY MODEL DETAILS

    param_length = gmm_param_length + reg_param_length
print(names(metadata_df))

select(metadata_df, df, param_length) %>%
  group_by(df) %>%
  summarize(param_length = unique(param_length))

select(metadata_df, df, initial_hess_time, initial_opt_time) %>%
  group_by(df) %>%
  summarize(initial_hess_time = median(initial_hess_time),
             initial_opt_time = median(initial_opt_time))

round(median(metadata_df$initial_opt_time), digits = -1)

[1] "num_comb"    "total_lr_time" "total_refit_time"
[4] "initial_opt_time" "initial_reg_time" "initial_hess_time"
[7] "gmm_param_length" "reg_param_length" "df"
[10] "lo_num_times" "init_method" "lr_hess_time"
[13] "avg_lr_time" "avg_refit_time" "param_length"

<table>
<thead>
<tr>
<th>df</th>
<th>param_length</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>23325</td>
</tr>
<tr>
<td>5</td>
<td>31643</td>
</tr>
<tr>
<td>6</td>
<td>38661</td>
</tr>
<tr>
<td>7</td>
<td>46379</td>
</tr>
<tr>
<td>8</td>
<td>54797</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>df</th>
<th>initial_hess_time</th>
<th>initial_opt_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>275.7295</td>
<td>31.4465</td>
</tr>
<tr>
<td>5</td>
<td>295.0325</td>
<td>41.84182</td>
</tr>
<tr>
<td>6</td>
<td>359.6855</td>
<td>35.11145</td>
</tr>
<tr>
<td>7</td>
<td>478.7345</td>
<td>50.88843</td>
</tr>
<tr>
<td>8</td>
<td>584.4987</td>
<td>77.02919</td>
</tr>
</tbody>
</table>

Make a dataframe for the timing plot from the metadata.

In [5]: metadata_graph_df <-

    metadata_df %>%
      select(df, lo_num_times, total_refit_time, lr_hess_time, melt(id.vars = c("lo_num_times", "df")))

head(metadata_graph_df)

<table>
<thead>
<tr>
<th>lo_num_times</th>
<th>df</th>
<th>variable</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>total_refit_time</td>
<td>338.1638</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>total_refit_time</td>
<td>391.6006</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>total_refit_time</td>
<td>423.8322</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>total_refit_time</td>
<td>632.2635</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>total_refit_time</td>
<td>999.0894</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>total_refit_time</td>
<td>1123.7316</td>
</tr>
</tbody>
</table>
1.0.3 Calculate prediction errors

Make summaries of prediction error for various methods and datasets.

In [7]: # In-sample IJ error.
    lr_df <-
    refit_err_melt %>%
    filter(rereg=use_rereg, method="lin", test=FALSE,
APPENDIX A. SUPPLEMENTARY MODEL DETAILS

```r
rename(error=value) %>%
mutate(output="lin_in_sample")
```

# In-sample CV error.
cv_df <-
  refit_err_melt %>%
  filter(rereg=use_rereg, method="ref", test==FALSE,
         rename(error=value) %>%
         mutate(output="cv_in_sample")

# In-sample training error (no points left out).
train_df <-
  refit_err_melt %>%
  filter(rereg=use_rereg, method="ref", test==FALSE,
         rename(error=value) %>%
         mutate(output="train_error")

# Out-of-sample test error.
test_df <-
  refit_err_melt %>%
  filter(rereg=use_rereg, method="ref", test==TRUE,
         rename(error=value) %>%
         mutate(output="test_error")

refit_for_df_choice <- bind_rows(
  lr_df, cv_df, test_df, train_df)

Make a tidy dataframe for choosing df. The graph in the paper will be based on this dataframe.

Note that most of the signal for choosing df is already in the training data error. However,
there is an uptick in error in both CV and IJ for df=8 which is not captured by the training data
error.

In [9]: refit_err_summary <-
  refit_for_df_choice %>%
  group_by(output, df, lo_num_times) %>%
  mutate(esize=abs(error)) %>%
  summarize(med=median(esize),
            mean=mean(esize),
            n_obs=n(),
            se=sd(esize) / sqrt(n_obs),
```
1.0.4 Gene-by-gene accuracy measures.

In [10]: refit_err_plot <-
    refit_err_melt %>%

```r
```
We now look at the correlation between the CV and IJ prediction errors across genes. For each df and k, there are a number of different combinations of left-out points. We report the median, min, and max correlation coefficients across these combinations of left-out points.

First, we show the correlation between the raw prediction errors. Although the correlation is quite high, this is because the training error at the original optimum is the principle source of variation in the errors across genes, and this quantity is common to both CV and IJ.

```r
In [11]: err_corr <- refit_err_plot %>% 
    filter(test==FALSE, rereg==use_rereg) %>% 
    group_by(df, lo_num_times, comb) %>% 
    summarize(r=cor(lin_err, ref_err)) %>% 
    group_by(df, lo_num_times) %>% 
    summarize(med_r=median(r), min_r=min(r), max_r=max(r))

print("Correlation between error: ")
print(err_corr)
```

A more meaningful measure is the correlation in the excess error for IJ and CV over the error at the original fit.

```r
In [12]: diff_corr <- refit_err_plot %>% 
    filter(test==FALSE, rereg==use_rereg) %>% 
    group_by(df, lo_num_times, comb) %>% 
```

APPENDIX A. SUPPLEMENTARY MODEL DETAILS

191
APPENDIX A. SUPPLEMENTARY MODEL DETAILS

```r
summarize(r = cor(lin_e_diff, ref_e_diff)) %>%
group_by(df, lo_num_times) %>%
summarize(med_r = median(r), min_r = min(r), max_r = max(r))
```

```r
print("Correlation between difference from train error: ")
print(diff_corr)
```

[1] "Correlation between difference from train error: 

# A tibble: 15 x 5
# Groups: df [?]

<table>
<thead>
<tr>
<th>df</th>
<th>lo_num_times</th>
<th>med_r</th>
<th>min_r</th>
<th>max_r</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.483</td>
<td>0.0956</td>
<td>0.844</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.577</td>
<td>0.277</td>
<td>0.828</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0.605</td>
<td>0.303</td>
<td>0.833</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0.464</td>
<td>0.143</td>
<td>0.728</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.510</td>
<td>0.330</td>
<td>0.709</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>0.510</td>
<td>0.312</td>
<td>0.671</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>0.655</td>
<td>0.368</td>
<td>0.783</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>0.588</td>
<td>0.218</td>
<td>0.845</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>0.499</td>
<td>0.0701</td>
<td>0.737</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>0.660</td>
<td>0.512</td>
<td>0.760</td>
</tr>
<tr>
<td>11</td>
<td>7</td>
<td>0.564</td>
<td>0.224</td>
<td>0.863</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>0.491</td>
<td>0.0344</td>
<td>0.801</td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td>0.744</td>
<td>0.380</td>
<td>0.900</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>0.646</td>
<td>0.166</td>
<td>0.862</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>0.214</td>
<td>-0.226</td>
<td>0.767</td>
</tr>
</tbody>
</table>

For higher degrees of freedom, increasing the number of left-out points seems to decrease the IJ’s accuracy, as you might expect.

```r
In [13]: ggplot(diff_corr) +
geom_bar(aes(x=paste(df, lo_num_times, sep=""),
         y=med_r, fill=as.character(df), stat="identity")
```
APPENDIX A. SUPPLEMENTARY MODEL DETAILS

Plot the densities of the IJ and CV with points to show outliers. This is a graphical version of the results summarized by the correlation tables above.

In [14]: # There are a few outliers, so limit the extent of the plot so that
   # the bulk of the distribution is visible.
   qlim <- quantile(refit_err_plot$ref_e_diff, c(0.1, 0.9))
   options(repr.plot.width=4, repr.plot.height=20)

   # This plot, or ones like it, is probably the best measure of
   # the accuracy of the IJ.
   ggplot(filter(refit_err_plot, test == FALSE, lo.num_times==1)) +
   geom_point(aes(x=ref_e_diff, y=lin_e_diff), alpha=0.01) +
   geom_density2d(aes(x=ref_e_diff, y=lin_e_diff)) +
geom_abline(aes(slope=1, intercept=0)) +
facet_grid(df ~ rereg) +
xlim(qlim[1], qlim[2]) + ylim(qlim[1], qlim[2])

Warning message:
Removed 10770 rows containing non-finite values (stat_density2d).Warning message:
Removed 10770 rows containing missing values (geom_point).
1.0.5 Save results for plotting in the paper.

```python
In [15]: print(sprintf("Saving to %s", file.path(save_dir, save_filename)))
    save(refit_err_summary,
         metadata_df,
         diff_corr,
         err_corr,
         file=file.path(save_dir, save_filename))

[1] "Saving to ../fits/paper_results_init_kmeans_rereg_FALSE.Rdata"
```
Bibliography


