Abstract

The error or variability of machine learning algorithms is often assessed by repeatedly re-fitting a model with different weighted versions of the observed data. The ubiquitous tools of cross-validation (CV) and the bootstrap are examples of this technique. These methods are powerful in large part due to their model agnosticism but can be slow to run on modern, large data sets due to the need to repeatedly re-fit the model. In this work, we use a linear approximation to the dependence of the fitting procedure on the weights, producing results that can be faster than repeated re-fitting by orders of magnitude. This linear approximation is sometimes known as the “infinitesimal jackknife” in the statistics literature, where it is mostly used to as a theoretical tool to prove asymptotic results. We provide explicit finite-sample error bounds for the infinitesimal jackknife in terms of a small number of simple, verifiable assumptions. Our results apply whether the weights and data are stochastic, deterministic, or even adversarially chosen, and so can be used as a tool for proving the accuracy of the infinitesimal jackknife on a wide variety of problems. As a corollary, we state mild regularity conditions under which our approximation consistently estimates true leave-$k$-out cross-validation for any fixed $k$. These theoretical results, together with modern automatic differentiation software, support the application of the infinitesimal jackknife to a wide variety of practical problems in machine learning, providing a “Swiss Army infinitesimal jackknife.” We demonstrate the accuracy of our methods on a range of simulated and real datasets.

1 Introduction

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 by resampling from the empirical distribution. In particular, we often form a new, proxy dataset by making a set of IID draws from the empirical distribution. 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 exchangeability imply that the dependence of an optimum on the weights is approximately linear for large sample sizes. 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 the classical CV or the bootstrap. The method is an instance of the infinitesimal jackknife, a general methodology that was historically a precursor to cross-validation and the bootstrap [Jaeckel, 1972, Efron, 1982]. Part of our argument is that variants of the infinitesimal jackknife should be reconsidered for modern large-scale applications because, for smooth optimization problems, it can be calculated automatically with modern automatic differentiation tools [Baydin et al., 2017].

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 infinitesimal jackknife 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 found in 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 infinitesimal jackknife 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.

2 Methods and Results

2.1 Problem definition

We consider the problem of estimating an unknown parameter \( \theta \in \Omega_\theta \subseteq \mathbb{R}^D \), with a compact \( \Omega_\theta \) 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, \( \hat{\theta} \in \Omega_\theta \), as the root of a weighted estimating equation. For each \( n = 1, \ldots, N \), let \( g_n(\theta) \) be a function from \( \Omega_\theta \) 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

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

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 \( \hat{\theta} = \arg\min_{\theta \in \Omega_\theta} \frac{1}{N} \sum_{n=1}^{N} f(x_n, \theta) \), then we would 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 6). However, to aid intuition, we will sometimes refer to the \( g_n(\theta) \) as “gradients” and their derivatives as “Hessians.”

When equation (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 (1) for the weight vector of all ones, \( 1_w := (1, \ldots, 1) \), which we denote \( \hat{\theta}_1 := \hat{\theta}(1_w) \).
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. We can show that our approximation is good for all leave-\( k \)-out analyses with probability one if we can show 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, \( w^*_b \) iid \( \sim \) 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 or even adversarial \( W \).

### 2.2 Linear approximation

The main problem we solve is the computational expense involved in evaluating \( \hat{\theta}(w) \) for all the \( w \in W \). 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 \((1)\).

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

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

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

Note that \( G(\hat{\theta}(w), w) = 0 \) because \( \hat{\theta}(w) \) solves equation \((1)\) 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

\[
\frac{d\hat{\theta}(w)}{dw} \bigg|_{w=1} \Delta w = -H_1^{-1} \frac{1}{N} \sum_{n=1}^{N} g_n \left( \hat{\theta}_1 \right) \Delta w
\]

\[
= -H_1^{-1} G \left( \hat{\theta}_1, \Delta w \right).
\]

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

**Definition 2.** Our linear approximation to \( \hat{\theta}(w) \) is given by

\[
\hat{\theta}_{1J} (w) := \hat{\theta}_1 - H_1^{-1} G \left( \hat{\theta}_1, \Delta w \right).
\]

We use the subscript “IJ” for “infinitesimal jackknife,” which is the name for this estimate in the statistics literature [Jaeckel, 1972; Shao, 1993]. Because \( \hat{\theta}_{1J} \) 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 (1). Instead, to calculate \( \hat{\theta}_{1J} \) 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 \( \hat{\theta}_{1J} (w) \) for each new weight costs only \( O(D) \), 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 6 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 \( \hat{\theta}_{1J} \) is in the computation of \( H_1^{-1} \). We leave the important question of approximate calculation of \( H_1^{-1} \) for future work.

### 2.3 Assumptions and results

We now state our key assumptions and results, which are sufficient conditions under which \( \hat{\theta}_{1J}(w) \) will be a good approximation to \( \hat{\theta}(w) \). We defer most proofs to Appendix A. 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 or adversarially chosen \( W \) under appropriate regularity conditions.
Assumption 1 (Smoothness). For all $\theta \in \Omega_\theta$, each $g_n(\theta)$ is continuously differentiable in $\theta$.

The smoothness in Assumption 1 is necessary for a local approximation like Definition 2 to have any hope of being useful.

Assumption 2 (Non-degeneracy). For all $\theta \in \Omega_\theta$, $H(\theta, 1_w)$ is non-singular, with $\sup_{\theta \in \Omega_\theta} \left\| H(\theta, 1_w)^{-1} \right\|_{op} \leq C_{op} < \infty$.

Without Assumption 2, the derivative in Definition 2 would not exist. For an optimization problem, Definition 2 amounts to assuming that the Hessian is positive definite at the optimum. Furthermore, by fixing $C_{op}$, if we want to apply Assumption 2 for $N \to \infty$, we will require that $H_1$ remains strongly positive definite.

Assumption 3 (Bounded averages). There exist finite constants $C_g$ and $C_h$ such that $\sup_{\theta \in \Omega_\theta} \frac{1}{\sqrt{N}} \| g(\theta) \|_2 \leq C_g < \infty$ and $\sup_{\theta \in \Omega_\theta} \frac{1}{\sqrt{N}} \| h(\theta) \|_2 \leq C_h < \infty$.

Assumption 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 $\sup_n \| g_n(\theta) \|_2^2 \to \infty$, as long as $\sup_n \frac{1}{N} \| g_n(\theta) \|_2^2$ remains bounded. This is a key advantage of the present work over many past applications of the infinitesimal jackknife to M-estimation, which require $\sup_n \| g_n(\theta) \|_2^2$ 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 (Local smoothness). There exists a $\Delta_\theta > 0$ and a finite constant $L_h$ such that, $\left\| \theta - \hat{\theta}_1 \right\|_2 \leq \Delta_\theta$ implies that $\left\| \frac{h(\theta) - h(\hat{\theta}_1)}{\sqrt{N}} \right\|_2 \leq L_h \left\| \theta - \hat{\theta}_1 \right\|_2$.

The constants defined in Assumption 4 are needed to calculate our error bounds explicitly.

Assumptions 1-4 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 1 (The assumptions hold under uniform convergence). Let $\Omega_\theta$ 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 \( \theta \in \Omega_\theta \), and that all of \( \mathbb{E}\left[ \sup_{\theta \in \Omega_\theta} \|g_n(\theta)\|^2 \right], \mathbb{E}\left[ \sup_{\theta \in \Omega_\theta} \|h_n(\theta)\|^2 \right] \), and \( \mathbb{E}\left[ \sup_{\theta \in \Omega_\theta} \|r_n(\theta)\|^2 \right] \) are finite.

Then \( \lim_{N \to \infty} P(\text{Assumptions 1–4 hold}) = 1. \)

Lemma 1 follows from the uniform convergence results in [Keener, 2011, Theorems 9.1 and 9.2]. See Appendix A.4 for a detailed proof. A common example to which Lemma 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 1–4 apply to the estimating equation. We also require a boundedness condition for \( W \).

Assumption 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 \( \hat{\theta}_{1\delta}(w) \) will be a good approximation to \( \hat{\theta}(w) \).

Condition 1 (Set complexity). There exists a \( \delta \geq 0 \) and a corresponding set \( W_\delta \subseteq W \) such that

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

Condition 1 is central to establishing when the approximation \( \hat{\theta}_{1\delta}(w) \) is accurate. For a given \( \delta \), \( W_\delta \) will be the class of weight vectors for which \( \hat{\theta}_{1\delta}(w) \) 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 3 we will discuss Condition 1 in more depth, but it will help to first state our main theorem.

Definition 3. The following constants are given by quantities in Assumptions 1–4.

\[
C_{IJ} := 1 + DC_wL_hC_{op}
\]
\[
\Delta_\delta := \min \left\{ \Delta_\delta C_{op}^{-1}, \frac{1}{2} C_{I\delta}^{-1} C_{op}^{-1} \right\}.
\]

Note that, although the parameter dimension \( D \) occurs explicitly only once in Definition 3, all of \( C_w, C_{op}, \) and \( L_h \) in general might also contain dimension dependence. Additionally, the bound \( \delta \) in Condition 1 is 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 1** (Error bound for the approximation). Under Assumptions [1] and Condition [2],
\[
\delta \leq \Delta \delta \Rightarrow \max_{w \in W} \| \hat{\theta}_{IJ}(w) - \hat{\theta}(w) \|_2 \leq 2C_{op}C_{IJ}\delta^2.
\]

We stress that Theorem 1 bounds only the difference between \( \hat{\theta}_{IJ}(w) \) and \( \hat{\theta}(w) \). Theorem 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 1 is useful to prove asymptotic results about the difference \( \| \hat{\theta}_{IJ}(w) - \hat{\theta}(w) \|_2 \). As an illustration, we now show that the uniform consistency of leave-\( k \)-out CV follows from Theorem 1 by a simple application of Hölder’s inequality.

**Corollary 1** (Consistency for leave-\( k \)-out CV). Assume that Assumptions [1] and [3] hold uniformly for all \( N \). Fix an integer \( k \), and let
\[
W_k := \{ w : w_n = 0 \text{ in } k \text{ entries and } 1 \text{ otherwise} \}.
\]
Then, for all \( N \), there exists a constant \( C_K \) such that
\[
\sup_{w \in W_k} \| \hat{\theta}_{IJ}(w) - \hat{\theta}(w) \|_2 \leq C_K \frac{\| g \|_{\infty}^2}{N^2} \leq C_K \frac{\max \{ C_g, C_h \}^2}{N}.
\]

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

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

with a similar bound for \(\|h\|_2\). Consequently, for \(N\) large enough, Condition 1 is satisfied with \(W_\delta = W_k\) and either \(\delta = K \frac{\|g\|_\infty}{N}\) or \(\delta = K \frac{C_{gh}}{\sqrt{N}}\). The result then follows from Theorem 1. \(\square\)

3 Examples

The moral of Theorem 1 is that, under Assumptions 1–5 and Condition 1,

\[ \|\hat{\theta}_{IJ} - \hat{\theta}(w)\| = O(\delta^2) \text{ for } w \in W_\delta. \] That is, if we can make \(\delta\) small enough, \(W_\delta\) big enough, and still satisfy Condition 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 1. We have already seen in Corollary 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, \(\|g_{n^*}(\hat{\theta}_1)\|_1\), is small, Condition 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 \(W_{full}\) is too expressive. In the language of Condition 1, for some small fixed \(\delta\), \(W_\delta\) will be some very restricted subset of \(W_{full}\) in most realistic situations.
Now, suppose that we are using $B$ bootstrap weights, $w^*_b \overset{iid}{\sim} \text{Multinomial}(N, N^{-1})$ for $b = 1, ..., B$, and analyzing an optimization problem as defined in Section 2.1. For a given $w^*_b$, a dataset $x_1^*, ..., x_N^*$ 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 $(x_1, ..., x_N)$. 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 1 is a uniform bound on a centered empirical process of derivatives of the objective function. Note that estimating sample variances by applying the infinitesimal jackknife 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 in 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 take $n^*$ as in the first pathological example, set $w_{n^*} = 0$, and let $w$ be one elsewhere. Then Condition 1 requires $\sup_{\theta \in \Omega} \frac{1}{N} g_{n^*}(\theta) \leq \delta$. 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 $\hat{\theta}_I$ in many cases of interest, as stated in Corollary 4.

### 4 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 infinitesimal jackknife 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. 2017, Maclaurin et al. 2015].

Our work is based on the proof of the Fréchet differentiability of M-estimators in [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, could be adapted to the infinitesimal jackknife. 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 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).

5 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 $Z = \{(x_n, y_n)\}_{n=1}^N$ and have as our parameters $(\theta, b)$, where $\theta \in \mathbb{R}^{100}$ is the vector of regression coefficients and $b \in \mathbb{R}$ is a bias term. In

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 to provide simple illustrative examples.
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. 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. 1 also 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.

6 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. The application is described in detail in Appendix B. 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 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 B).

We implemented the model in scipy [Jones et al., 2001] and computed all derivatives with autograd [Maclaurin et al., 2015]. We note that the match between “exact” cross-validation (removing time points and re-optimizing) and the infinitesimal jackknife 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. 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. 2 shows that the linear approximation to cross validation is a remarkably good approximation to the true out-of-sample error. Moreover, it appears that on this dataset, our approximation is an even better estimate of the test error than cross-validation, possibly due to the difficulty of re-optimizing for the “exact” CV.

For this particular problem with approximately 66199 parameters (the precise number depends on the degrees of freedom), finding the initial optimum takes about 400 seconds. Computing and inverting a dense matrix of this size would be computationally prohibitive, but $H_1$ for the regression objective is extremely sparse and block diagonal, so computing $H_1^{-1}$ also took only around 400 seconds. Once we have $H_1^{-1}$, obtaining the subsequent linear approximations is nearly instantaneous, while repeatedly refitting the model for CV is orders of magnitude more expensive, as seen in Fig. 2.
7 Conclusion

We recommend the consideration of the Swiss Army infinitesimal jackknife for modern machine learning problems. The large size of modern data both increases the need for fast approximations and renders such approximations more accurate. Furthermore, modern automatic differentiation renders many of its practical difficulties obsolete. By stepping back from the strict requirements of classical statistical theory, the infinitesimal jackknife can be seen to be valuable beyond the problems to which it has been traditionally applied while still retaining the benefits of its generality.
References

N. Agarwal, B. Bullins, and E. Hazan. Second-order stochastic optimization in linear time. *Journal of Machine Learning Research*, 2017.

A. Baydin, B. Pearlmutter, A. Radul, and J. Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–153, 2017.

A. Beirami, M. Razaviyayn, S. Shahrampour, and V. Tarokh. On optimal generalizability in parametric learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3458–3468, 2017.

B. Clarke. Uniqueness and Fréchet differentiability of functional solutions to maximum likelihood type equations. *The Annals of Statistics*, 11(4):1196–1205, 1983.

B. Efron. *The Jackknife, the Bootstrap, and Other Resampling Plans*, volume 38. Society for Industrial and Applied Mathematics, 1982.

L. Fernholz. *Von Mises calculus for statistical functionals*, volume 19. Springer Science & Business Media, 1983.

P. Hall. *The bootstrap and Edgeworth expansion*. Springer Science & Business Media, 2013.

L. Jaeckel. The infinitesimal jackknife, memorandum. Technical report, MM 72-1215-11, Bell Lab. Murray Hill, NJ, 1972.

E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001. URL http://www.scipy.org/.

R. W. Keener. *Theoretical statistics: Topics for a core course*. Springer, 2011.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 2017.

Y. Luan and H. Li. Clustering of time-course gene expression data using a mixed-effects model with b-splines. *Bioinformatics*, 19(4):474–482, 2003.

D. Maclaurin, D. Duvenaud, and R. P. Adams. Autograd: Effortless gradients in numpy. In *International Conference on Machine Learning (ICML) AutoML Workshop*, 2015.

R. Mises. On the asymptotic distribution of differentiable statistical functions. *The Annals of Mathematical Statistics*, 18(3):309–348, 1947.

K. R. Rad and A. Maleki. A scalable estimate of the extra-sample prediction error via approximate leave-one-out. *arXiv Preprint*, January 2018.

J. A. Reeds. Jackknifing maximum likelihood estimates. *The Annals of Statistics*, pages 727–739, 1978.
S. Rosset and R. J. Tibshirani. From fixed-x to random-x regression: Bias-variance decompositions, covariance penalties, and prediction error estimation. *Journal of the American Statistical Association*, (just-accepted), 2018.

J. Schott. *Matrix analysis for statistics*. John Wiley & Sons, 2016.

J. Shao. Differentiability of statistical functionals and consistency of the jackknife. *The Annals of Statistics*, pages 61–75, 1993.

J. Shao and D. Tu. *The Jackknife and Bootstrap*. Springer Series in Statistics, 2012.

J. E. Shoemaker, S. Fukuyama, A. J. Eisfeld, D. Zhao, E. Kawakami, S. Sakabe, T. Maemura, T. Gorai, H. Katsura, Y. Muramoto, S. Watanabe, T. Watanabe, K. Fuji, Y. Matsuoka, H. Kitano, and Y. Kawaoka. An Ultrasensitive Mechanism Regulates Influenza Virus-Induced Inflammation. *PLoS Pathogens*, 11(6):1–25, 2015.

S. Wager, T. Hastie, and B. Efron. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *The Journal of Machine Learning Research*, 15(1):1625–1651, 2014.

S. Wright and J. Nocedal. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.
A Detailed assumptions, lemmas, and proofs

A.1 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 1. Tensor array version of Hölder’s inequality. Let $w$ be an array of scalars and let $a = (a_1, ..., a_N)$ be an array of tensors, were each $a_n$ is indexed by $i = 1, ..., D_A$ (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

$$\left\| \frac{1}{N} \sum_{n=1}^{N} w_n a_n \right\|_1 \leq \frac{D_A^{\frac{p}{q}}}{N} \|w\|_p \|a\|_q.$$  

In particular, with $p = q = 2$,

$$\left\| \frac{1}{N} \sum_{n=1}^{N} w_n a_n \right\|_1 \leq \sqrt{D_A} \|w\|_2 \|a\|_2.$$  

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

\[
\left\| \frac{1}{N} \sum_{n=1}^{N} w_n a_n \right\|_1 = \sum_{i=1}^{D_A} \frac{1}{N} \sum_{n=1}^{N} w_n (a_n)_i \\
\leq \frac{1}{N} \sum_{i=1}^{D_A} \left( \sum_{n=1}^{N} |w_n|^p \right)^{\frac{1}{p}} \left( \sum_{n=1}^{N} |(a_n)_i|^q \right)^{\frac{1}{q}} \quad \text{(Hölder)} \\
= \frac{1}{N} \|w\|_p D_A \sum_{i=1}^{D_A} \left( \sum_{n=1}^{N} |(a_n)_i|^q \right)^{\frac{1}{q}} \\
\leq \frac{1}{N} \|w\|_p D_A \left( \frac{1}{D_A} \sum_{i=1}^{D_A} \sum_{n=1}^{N} |(a_n)_i|^p \right)^{\frac{1}{q}} \quad \text{(Jensen applied to $i$)} \\
= \frac{1}{N} \|w\|_p D_A \left( \frac{1}{D_A} \sum_{n=1}^{N} \|a_n\|_q^{p} \right)^{\frac{1}{q}} \\
= \frac{1}{N} \|w\|_p D_A^{1-\frac{1}{q}} \|a\|_q \\
= \frac{D_A^{\frac{p}{q}}}{N} \|w\|_p \|a\|_q. 
\]
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 2.** Let $A$ and $B$ be two matrices. Let $\|A^{-1}\|_{op} \leq C_{op}$ for some finite $C_{op}$, Then

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

**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

$$\left\| A^{-1} - (A - B)^{-1} \right\| \leq \frac{\|A^{-1}\| \|A - B\|}{1 - \|A^{-1}\| \|A - B\|} \|A^{-1}\|.$$  \hspace{1cm} (2)

We will apply equation (2) using the operator norm $\|\cdot\|_{op}$, for which $\|I\|_{op} = 1$. First, note that

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

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

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

so we can apply equation (2). Then

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

$$\leq \frac{\|A^{-1}\|_{op} \|A - B\|_{op}}{1 - \|A^{-1}\|_{op} \|A - B\|_{op}} \|A^{-1}\|_{op} + \|A^{-1}\|_{op} \text{ (Equation 2)}$$

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

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

\[\square\]

### A.2 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 2 below.)

2. When $\|\theta - \hat{\theta}_1\|_2$ is small, then the derivatives $H(\theta, w)$ are close to their optimal value, $H\left(\hat{\theta}_1, 1_w\right)$. (Lemma 3 and Lemma 4 below.)
3. When the derivatives are close to their optimal values, then \( H(\theta, w) \) is uniformly non-singular. (Lemma 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 \( \hat{\theta}_{ij} - \hat{\theta}(w) \) in terms of \( \delta \). (Theorem 2 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 1 small.

**Lemma 2. Small parameter changes.** Under Assumptions 1—3 and Condition 1,

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

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

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

By Assumption 2, \( H\left(\hat{\theta}_1, 1_w\right) \) 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

\[
G\left(\hat{\theta}(w), 1_w\right) - G\left(\hat{\theta}(w), w\right) = H\left(\hat{\theta}_1, 1_w\right)\left(\hat{\theta}(w) - \hat{\theta}_1\right) \Rightarrow 
\]

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

Applying Condition 1 and Assumption 2,

\[
\left\| \hat{\theta}(w) - \hat{\theta}_1 \right\|_2 = \left\| H\left(\hat{\theta}_1, 1_w\right)^{-1}\left(G\left(\hat{\theta}(w), 1_w\right) - G\left(\hat{\theta}(w), w\right)\right) \right\|_2
\]

\[
\leq \left\| H\left(\hat{\theta}_1, 1_w\right)^{-1} \right\|_{op} \left\| G\left(\hat{\theta}(w), 1_w\right) - G\left(\hat{\theta}(w), w\right) \right\|_2
\]

\[
\leq \sup_{\theta \in \Omega_\theta} \left\| H\left(\theta, 1_w\right)^{-1} \right\|_{op} \left\| G\left(\theta, 1_w\right) - G\left(\theta, w\right) \right\|_2
\]

\[
\leq C_{op} \left\| G\left(\hat{\theta}(w), 1_w\right) - G\left(\hat{\theta}(w), w\right) \right\|_2 \quad \text{ (Assumption 2)}
\]

\[
\leq C_{op} \left\| G\left(\hat{\theta}(w), 1_w\right) - G\left(\hat{\theta}(w), w\right) \right\|_1 \quad \text{ (relation between norms)}
\]

\[
\leq C_{op} \sup_{\theta \in \Omega_\theta} \left\| G\left(\theta, 1_w\right) - G\left(\theta, w\right) \right\|_1
\]

\[
\leq C_{op}\delta. \quad \text{(Condition 1)}.
\]
Because we will refer to it repeatedly, we give the set of \( \theta \) defined in Lemma 2 a name.

**Definition 4.** For a given \( \delta \), define the region around \( \hat{\theta}_1 \) given by Lemma 2 as

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

In other words, Lemma 2 states that Condition 1 implies \( \hat{\theta}(w) \in B_{C_{op}\delta} \) when \( w \in W_\delta \).

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

**Lemma 3.** Boundedness and continuity. Under Assumptions 1–5 and Condition 1, for all \( \theta \in B_{\Delta_\theta} \),

\[
\sup_{w \in W} \left\| H(\theta, w) - H(\hat{\theta}_1, w) \right\|_1 \leq DC_w L_h \left\| \theta - \hat{\theta}_1 \right\|_2.
\]

**Proof.** For \( \theta \in B_{\Delta_\theta} \),

\[
\sup_{w \in W} \left\| H(\theta, w) - H(\hat{\theta}_1, w) \right\|_1 = \sup_{w \in W} \left\| \frac{1}{N} \sum_{n=1}^{N} w_n \left( h_n(\theta) - h_n(\hat{\theta}_1) \right) \right\|_1
\]

(by definition)

\[
\leq D \sup_{w \in W} \frac{\|w\|_2}{\sqrt{N}} \frac{\|h(\theta) - h(\hat{\theta}_1)\|_2}{\sqrt{N}} \leq \frac{DC_w}{\sqrt{N}} \|h(\theta) - h(\hat{\theta}_1)\|_2 \leq DC_w L_h \left\| \theta - \hat{\theta}_1 \right\|_2.
\]

We now combine Lemma 2 and Lemma 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 \( \theta \in B_{C_{op}\delta} \).

**Lemma 4.** Bounds for difference in parameters. Under Assumptions 1–5 and Condition 1, if \( \delta \leq \Delta_\theta C_{op}^{-1} \), then

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

**Proof.** By Lemma 2 \( \delta \leq \Delta_\theta C_{op}^{-1} \) implies that \( C_{op}\delta \leq \Delta_\theta \) and so \( B_{C_{op}\delta} \subseteq B_{\Delta_\theta} \). Consequently, we can apply Lemma 3

\[
\sup_{\theta \in B_{C_{op}\delta}} \sup_{w \in W_\delta} \left\| H(\theta, w) - H(\hat{\theta}_1, 1_w) \right\|_1 \leq DC_w L_h \left\| \theta - \hat{\theta}_1 \right\|_2 \leq DC_w L_h C_{op} \delta.
\]
Next, we can use this to write
\[
\sup_{\theta \in B_{\cop, \delta}} \sup_{w \in W_3} \left\| H(\theta, w) - H\left(\hat{\theta}_1, 1_w\right) \right\|_1
= \sup_{\theta \in B_{\cop, \delta}} \sup_{w \in W_3} \left\| H(\theta, w) - H(\theta, 1_w) + H(\theta, 1_w) - H\left(\hat{\theta}_1, 1_w\right) \right\|_1
\leq \sup_{\theta \in B_{\cop, \delta}} \sup_{w \in W_3} \left\| H(\theta, w) - G(\theta, 1_w) \right\|_1 + \sup_{\theta \in B_{\cop, \delta}} \sup_{w \in W_3} \left\| H(\theta, 1_w) - H\left(\hat{\theta}_1, 1_w\right) \right\|_1
\leq \delta + \sup_{\theta \in B_{\cop, \delta}} \sup_{w \in W_3} \left\| H(\theta, 1_w) - H\left(\hat{\theta}_1, 1_w\right) \right\|_1 \tag{Condition 1}
\]
\[
\leq \delta + DC_wL_hC_{\cop, \delta}.
\]

The constant that appears multiplying \(\delta\) at the end of the proof of Lemma 4 will appear often in what follows, so we give it the special name \(C_{1J}\) in Definition 3.

Note that Lemma 4 places a condition on how small \(\delta\) must be in order for our regularity conditions to apply. Lemma 2 will guarantee that \(\hat{\theta}(w) \in B_{\cop, \delta}\), but if we are not able to make \(\delta\) arbitrarily small in Condition 1 then we are not guaranteed to ensure that \(B_{\cop, \delta} \subseteq B_{\Delta}\), will not be able to assume Lipschitz continuity, and none of our results will apply.

Next, using Lemma 4, we can extend the operator bound on \(H^{-1}\) from Assumption 2 to \(H(\theta, w)^{-1}\) for all \(w \in W_3\), not only for \(w = 1_w\).

**Lemma 5.** *Uniform invertibility of the Hessian.* Under Assumptions 1–5 and Condition 1, if \(\delta \leq \min\{\Delta_{\theta}C_{\cop}^{-1}, \frac{1}{2}C_{1J}^{-1}C_{\cop}^{-1}\}\), then
\[
\sup_{\theta \in B_{\cop, \delta}} \sup_{w \in W_3} \left\| H(\theta, w)^{-1} \right\|_{op} \leq 2C_{\cop}.
\]

*Proof.* By Assumption 2, \(\left\| H\left(\hat{\theta}_1, 1_w\right)^{-1} \right\|_{op} \leq C_{\cop}\). So by Proposition 2, it will suffice to select \(\delta\) so that
\[
\sup_{\theta \in B_{\cop, \delta}} \sup_{w \in W_3} \left\| H(\theta, w) - H\left(\hat{\theta}_1, 1_w\right) \right\|_1 \leq \frac{1}{2}C_{\cop}^{-1} \tag{3}
\]
When we can apply Lemma 4, we have
\[
\sup_{\theta \in B_{\cop, \delta}} \sup_{w \in W_3} \left\| H(\theta, w) - H\left(\hat{\theta}_1, 1_w\right) \right\|_1 \leq C_{1J}\delta.
\]
So \(H(\theta, w)\) will satisfy equation (3) if we can apply Lemma 4 and if
\[
\delta \leq \frac{1}{2}C_{\cop}^{-1}C_{1J}^{-1}.
\]
To apply Lemma 4 we additionally require that $\delta \leq \Delta d_{opp}$. By taking $\delta \leq \min \left\{ \Delta d_{opp}, \frac{1}{2} C_{opp}^{-1} C_{opp}^{-1} \right\}$, we satisfy equation (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 3.

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

**Lemma 6. Inverse function theorem.** Under Assumptions 1–5 and Condition 1, and for $\delta \leq \Delta_{\delta}$, there exists a continuous, differentiable function of $w$, $\hat{\theta}(w)$, such that, for all $w \in W$, $G\left(\hat{\theta}(w), w\right) = 0$.

**Proof.** This follows from Lemma 5 and the implicit function theorem.

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

### A.3 Bounding the errors in a Taylor expansion

We are now in a position to use Assumptions 1–5 and Condition 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 3.** For any $w \in W$ for which $H\left(\hat{\theta}(w), w\right)$ is invertible, and for any vector $a \in \mathbb{R}^N$,

$$
\frac{d\hat{\theta}(w)}{dw} |_{w,a} = -H\left(\hat{\theta}(w), w\right)^{-1} G\left(\hat{\theta}(w), a\right).
$$

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

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

Because $H\left(\hat{\theta}(w), w\right)$ is invertible by assumption, the result follows. \hspace{1cm} \square
Definition 5. Define
\[
\hat{\theta}_{1j}(w) := \hat{\theta}_1 + \frac{d\hat{\theta}(w)}{dw} (w - 1_w)
\]
\[
= \hat{\theta}_1 - H_1^{-1} G (\hat{\theta}_1, w) \quad \text{(because } G (\hat{\theta}_1, 1_w) = 0) \]

The first term \( \hat{\theta}_{1j}(w) \) in Definition 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, \( \hat{\theta}_{1j}(w) - \hat{\theta}(w) \).

Theorem 2. Under Assumptions 1–5 and Condition 1, when \( \delta \leq \Delta_\delta \),
\[
\sup_{w \in W_\delta} \| \hat{\theta}_{1j}(w) - \hat{\theta}(w) \|_2 \leq 2 C_{op}^2 C_{1j} \delta^2.
\]

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 \( \tilde{\theta} \) such that \( \| \tilde{\theta} - \hat{\theta}_1 \|_2 \leq \| \hat{\theta}(w) - \hat{\theta}_1 \|_2 \),
\[
0 = G (\hat{\theta}(w), w) = G (\hat{\theta}_1, w) + H (\hat{\theta}, w) (\hat{\theta}(w) - \hat{\theta}_1).
\]
Because \( \delta \in W_\delta \), Lemma 2 implies that \( \hat{\theta}(w) \in B_{C_{op}\delta} \). Because \( \| \hat{\theta} - \hat{\theta}_1 \|_2 \leq \| \hat{\theta}(w) - \hat{\theta}_1 \|_2 \), \( \hat{\theta} \in B_{C_{op}\delta} \) as well. Because \( \hat{\theta} \in B_{C_{op}\delta} \), Lemma 5 implies that \( H (\hat{\theta}, w) \) is invertible, so we can solve for \( \hat{\theta}(w) - \hat{\theta}_1 \).
\[
\hat{\theta}(w) - \hat{\theta}_1 = -H (\hat{\theta}, w)^{-1} G (\hat{\theta}_1, w)
\]
\[
= \left( -H (\hat{\theta}, w)^{-1} + H (\hat{\theta}_1, 1_w)^{-1} - H (\hat{\theta}_1, 1_w)^{-1} \right) G (\hat{\theta}_1, w)
\]
\[
= \left( H (\hat{\theta}_1, 1_w)^{-1} - H (\hat{\theta}, w)^{-1} \right) G (\hat{\theta}_1, w) + \hat{\theta}_{1j}(w) - \hat{\theta}_1.
\]
Eliminating \( \hat{\theta}_1 \) and taking the supremum of both sides we have that

\[
\begin{align*}
\sup_{w \in \mathcal{W}_d} \left\| \hat{\theta}_1 (w) - \hat{\theta} (w) \right\|_2 &= \sup_{w \in \mathcal{W}_d} \left\| \left( H \left( \hat{\theta}_1, 1_w \right)^{-1} - H \left( \hat{\theta}, w \right)^{-1} \right) G \left( \hat{\theta}_1, w \right) \right\|_2 \\
&= \sup_{w \in \mathcal{W}_d} \left\| H \left( \hat{\theta}, w \right)^{-1} \left( H \left( \hat{\theta}, w \right) - H \left( \hat{\theta}_1, 1_w \right) \right) H \left( \hat{\theta}_1, 1_w \right)^{-1} G \left( \hat{\theta}_1, w \right) \right\|_2 \\
&\leq 2C_{op} \sup_{w \in \mathcal{W}_d} \left\| H \left( \hat{\theta}, w \right) - H \left( \hat{\theta}_1, 1_w \right) \right\| \left\| H \left( \hat{\theta}_1, 1_w \right)^{-1} G \left( \hat{\theta}_1, w \right) \right\|_2 \quad \text{(Lemma 5)} \\
&\leq 2C_{op} \sup_{w \in \mathcal{W}_d} \left\| H \left( \hat{\theta}, w \right) - H \left( \hat{\theta}_1, 1_w \right) \right\|_{op} \left\| H \left( \hat{\theta}_1, 1_w \right)^{-1} G \left( \hat{\theta}_1, w \right) \right\|_2 \quad \text{(ordering of matrix norms)} \\
&\leq 2C_{op} C_{IJ} \delta \sup_{w \in \mathcal{W}_d} \left\| H \left( \hat{\theta}_1, 1_w \right)^{-1} G \left( \hat{\theta}_1, w \right) \right\|_2 \quad \text{(Lemma 3)} \\
&\leq 2C_{op} C_{IJ} \delta \left\| G \left( \hat{\theta}_1, w \right) \right\|_2 \quad \text{(Assumption 2)} \\
&= 2C_{op} C_{IJ} \delta \sup_{w \in \mathcal{W}_d} \left\| G \left( \hat{\theta}_1, 1_w \right) - G \left( \hat{\theta}_1, 1_w \right) \right\|_2 \quad \text{(because } G \left( \hat{\theta}_1, 1_w \right) = 0) \\
&\leq 2C_{op} C_{IJ} \delta^2 \quad \text{(Condition 1)}.
\end{align*}
\]

\(
\square
\)

### A.4 Use cases

First, let us state a simple condition under which Assumptions 1-4 hold. It will help to have a lemma for the Lipschitz continuity.

**Lemma 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 \( \frac{\partial a (\theta)}{\partial \theta} = \left( \frac{\partial a_1 (\theta)}{\partial \theta}, ..., \frac{\partial a_N (\theta)}{\partial \theta} \right) \) be an array of tensors of size \( D \times D_A \). Then

\[
\left\| \frac{\partial}{\partial \theta} \left\| a (\theta) \right\|_2 \right\|_2 \leq D_A \left\| \frac{\partial a}{\partial \theta} \right\|_2.
\]
Proof. By direct calculation,

\[
\left\| \frac{\partial}{\partial \theta} \| a(\theta) \|^2 \right\|_2^2 = \sum_{r=1}^D \left( \frac{\partial}{\partial \theta_r} \sum_{n=1}^N \sum_{i=1}^{D_A} a_{n,i}(\theta)^2 \right)^2 \\
= \sum_{r=1}^D \left( \sum_{n=1}^N \sum_{i=1}^{D_A} 2a_{n,i}(\theta) \frac{\partial a_{n,i}(\theta)}{\partial \theta_r} \right)^2 \\
\leq \sum_{r=1}^D \left( \sum_{i=1}^{D_A} \left( \sum_{n=1}^N a_{n,i}(\theta)^2 \right)^{\frac{1}{2}} \left( \sum_{n=1}^N \left( \frac{\partial a_{n,i}(\theta)}{\partial \theta_r} \right)^2 \right)^{\frac{1}{2}} \right)^2 \\
\leq \sum_{r=1}^D \left( 2D_A^2 \left( \frac{1}{D_A} \sum_{i=1}^{D_A} \sum_{n=1}^N a_{n,i}(\theta)^2 \right)^{\frac{1}{2}} \left( \frac{1}{D_A} \sum_{n=1}^N \left( \frac{\partial a_{n,i}(\theta)}{\partial \theta_r} \right)^2 \right)^{\frac{1}{2}} \right)^2 \\
= 4D_A^2 \| a \|^2 \sum_{r=1}^D \left\| \frac{\partial a}{\partial \theta_r} \right\|_2^2 \\
= 4D_A^2 \| a \|^2 \left\| \frac{\partial a}{\partial \theta} \right\|_2^2.
\]

By the chain rule,

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

Lemma 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 \( \theta \in \Omega_\theta \). 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 $i$,

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

Consequently,

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

So for any $\theta_1, \theta_2 \in \Omega$, we have:

$$
|E [\|a(\theta_1)\|_2^2] - E [\|a(\theta_2)\|_2^2]| \leq E [\|a(\theta_1)\|_2^2 - \|a(\theta_2)\|_2^2] \\
\leq E \left[ \left( \sup_{\theta \in \Omega} \left\| \frac{\partial}{\partial \theta} \|a(\theta)\|_2^2 \right\|_2 \right) \|\theta_1 - \theta_2\|_2 \right] \quad (\theta \text{ is not random}) \\
\leq E \left[ \left( \sup_{\theta \in \Omega} \left\| \frac{\partial a(\theta)}{\partial \theta} \right\|_2^2 \right) \|\theta_1 - \theta_2\|_2 \right] \\
\leq \sqrt{E \left[ \left( \sup_{\theta \in \Omega} \left\| \frac{\partial a(\theta)}{\partial \theta} \right\|_2^2 \right) \|\theta_1 - \theta_2\|_2 \right]^2}.
$$

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.** Let $\Omega_\theta$ be a compact set. Let $g_n(\theta)$ be twice continuously differentiable IID random functions. Define

$$h_n(\theta) := \frac{\partial g_n(\theta)}{\partial \theta}$$

$$r_n(\theta) := \frac{\partial^2 g_n(\theta)}{\partial \theta \partial \theta},$$

where $r_n(\theta)$ is a $D \times D \times D$ tensor. Assume that

1a) $E\left[\sup_{\theta \in \Omega_\theta} \|g_n(\theta)\|_2^2\right] < \infty$;

1b) $E\left[\sup_{\theta \in \Omega_\theta} \|h_n(\theta)\|_2^2\right] < \infty$;

1c) $E\left[\sup_{\theta \in \Omega_\theta} \|r_n(\theta)\|_2^2\right] < \infty$;

2) $E[h_n(\theta)]$ is non-singular for all $\theta \in \Omega_\theta$;

3) We can exchange expectation and differentiation.

Then $\lim_{N \to \infty} P(\text{Assumptions 1–4 hold}) = 1$.

**Proof.** The proof is a consequence of Keener [2011, Theorems 9.1 and 9.2]. We will first show that the expected values of the needed functions satisfy Assumptions 1–4, and then that the sample versions converge uniformly.

By Jensen’s inequality,

$$E\left[\sup_{\theta \in \Omega_\theta} \|g_n(\theta)\|_2^2\right] = E\left[\sqrt{\sup_{\theta \in \Omega_\theta} \|g_n(\theta)\|_2^2}\right] \leq \sqrt{E\left[\sup_{\theta \in \Omega_\theta} \|g_n(\theta)\|_2^2\right]}.$$

Also, for the $i^{th}$ component of $g_n(\theta)$

$$E\left[\sup_{\theta \in \Omega_\theta} |g_{n,i}(\theta)|\right] \leq E\left[\sup_{\theta \in \Omega_\theta} \|g_n(\theta)\|_\infty\right] \leq E\left[\sup_{\theta \in \Omega_\theta} \|g_n(\theta)\|_2\right].$$

By Keener [2011, Theorem 9.1], $E\left[\|g_n(\theta)\|_2^2\right]$, $E[\|g_n(\theta)\|_2]$], and $E[g_n(\theta)]$ 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

$$\sup_{\theta \in \Omega_\theta} E\left[\|g_n(\theta)\|_2^2\right] =: Q_g^2 < \infty$$

$$\sup_{\theta \in \Omega_\theta} E\left[\|h_n(\theta)\|_2^2\right] =: Q_h^2 < \infty.$$

Below, these constants will be used to satisfy Assumption 1 and Assumption 3 with high probability.

Because $\Omega_\theta$ is compact, $E[h_n(\theta)]$ is continuous, $E[h_n(\theta)]$ is non-singular, and the operator norm is a continuous function of $E[h_n(\theta)]$, we can also define

$$\sup_{\theta \in \Omega_\theta} \left\|E[h_n(\theta)]^{-1}\right\|_{op} =: Q_{op} < \infty.$$
Below, this constant be used to satisfy Assumption 2 with high probability.

Finally, we turn to the Lipschitz condition. Lemma 8 implies that
\[ |E[\|h_n(\theta_1)\|_2] - E[\|h_n(\theta_2)\|_2]| \leq \sqrt{E[\sup_{\theta \in \Omega} \|r_n(\theta)\|_2^2]} \|\theta_1 - \theta_2\|_2. \]

Define
\[ \Lambda_h = \sqrt{E[\sup_{\theta \in \Omega} \|r_n(\theta)\|_2^2]}, \]
so that we have shown that \( E[\|h_n(\theta)\|_2] \) is Lipschitz in \( \Omega \) 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 1–4 with \( N = 1 \). We now need to show that the sample versions satisfy Assumptions 1–4 with high probability, which will follow from the fact that the sample versions converge uniformly to their expectations by Keener [2011, Theorem 9.2].

First, observe that Assumption 1 holds with probability one by assumption.

For the remaining assumption choose an \( \epsilon > 0 \), and define
\[ C_g := \sqrt{Q_g^2 + \epsilon}, \]
\[ C_h := \sqrt{Q_h^2 + \epsilon}, \]
\[ C_{op} := 2Q_{op}, \]
\[ L_h := \sqrt{D^4\Lambda_h^2 + \epsilon}. \]

By Keener [2011, Theorem 9.2],
\[ \sup_{\theta \in \Omega} \left| \frac{1}{N} \sum_{n=1}^{N} \|g_n(\theta)\|_2^2 - E\left[\|g_n(\theta)\|_2^2\right]\right| \xrightarrow{p_{N \to \infty}} 0. \]

Because
\[ \sup_{\theta \in \Omega} \left| \frac{1}{N} \sum_{n=1}^{N} \|g_n(\theta)\|_2^2 - E\left[\|g_n(\theta)\|_2^2\right]\right| > Q_g^2 + \epsilon \geq \sup_{\theta \in \Omega} E\left[\|g_n(\theta)\|_2^2\right] + \epsilon \Rightarrow \]
\[ \sup_{\theta \in \Omega} \left| \frac{1}{N} \sum_{n=1}^{N} \|g_n(\theta)\|_2^2 - E\left[\|g_n(\theta)\|_2^2\right]\right| > \epsilon, \]
we have
\[ P\left( \sup_{\theta \in \Omega} \left| \frac{1}{N} \sum_{n=1}^{N} \|g_n(\theta)\|_2^2 \right| \geq Q_g^2 + \epsilon \right) \leq \]
\[ P\left( \sup_{\theta \in \Omega} \left| \frac{1}{N} \sum_{n=1}^{N} \|g_n(\theta)\|_2^2 - E\left[\|g_n(\theta)\|_2^2\right]\right| \leq \epsilon \right). \]
so

$$P \left( \sup_{\theta \in \Omega} \left| \frac{1}{N} \sum_{n=1}^{N} \| g_n (\theta) \|_2^2 \right| \geq C_g^2 \right) \xrightarrow{N \to \infty} 0.$$  

An analogous argument holds for $$\frac{1}{N} \| h_n (\theta) \|_2^2$$. Consequently, $$P \ (\text{Assumption 3 holds}) \xrightarrow{N \to \infty} 1$$.

We now consider Assumption 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,

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

By the converse of Proposition 2, because $$\left\| \mathbb{E} [h_n (\theta)]^{-1} \right\|_{op} \leq Q_{op},$$

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

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

Consequently,

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

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

and $$P \ (\text{Assumption 2 holds}) \xrightarrow{N \to \infty} 1$$.

Finally, applying Lemma 8 to $$\frac{1}{\sqrt{N}} \| h (\theta_2) \|_2$$,

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

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

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

29
Consequently,
\[
\left| \frac{1}{\sqrt{N}} \| h(\theta_1) \|_2 - \frac{1}{\sqrt{N}} \| h(\theta_2) \|_2 \right| \geq L_h \| \theta_1 - \theta_2 \|_2 \Rightarrow
\]
\[ D^2 \sqrt{\sup_{\theta \in \Omega} \frac{1}{N} \| r(\theta) \|_2^2} \geq L_h \Rightarrow \]
\[
\sup_{\theta \in \Omega} \frac{1}{N} \| r(\theta) \|_2^2 - \sup_{\theta \in \Omega} \mathbb{E} \left[ \| r_n(\theta) \|_2^2 \right] \geq \frac{L_h^2}{D^4} - \sup_{\theta \in \Omega} \mathbb{E} \left[ \| r_n(\theta) \|_2^2 \right] \Rightarrow \]
\[
\sup_{\theta \in \Omega} \frac{1}{N} \| r(\theta) \|_2^2 - \mathbb{E} \left[ \| r_n(\theta) \|_2^2 \right] \geq \frac{L_h^2}{D^4} - \Lambda_h^2 = \epsilon. \]

However, again by [Keener 2011 Theorem 9.2],
\[
\sup_{\theta \in \Omega} \left| \frac{1}{N} \| r(\theta) \|_2^2 - \mathbb{E} \left[ \| r_n(\theta) \|_2^2 \right] \right| \xrightarrow{P} 0, \quad N \to \infty
\]
so \( P(\text{Assumption 4 holds}) \xrightarrow{N \to \infty} 1. \) \( \square \)
B  Genomics Experiments Details

We demonstrate the python code used to run the experiments on the genomics data in a Jupyter notebook, reproduced below.

This notebook requires Jupyter notebook extensions. See https://github.com/ipython-contrib/jupyter_contrib_nbextensions for installation details.

After installing, run

```
jupyter nbextention enable python-markdown/main
```

before opening the Jupyter notebook.

You may also have to go to File at the top left of the notebook, and click Trust Notebook.

Genomics Experiments 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 $g = 1, \ldots, n_g$ and time points $t = 1, \ldots, n_t$, where in this case $n_g = 1000$ and $n_t = 42$.

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. Finally, we calculate the infinitesimal jackknife and compare it to re-optimizing.

```python
from copy import deepcopy
import inspect
import matplotlib.pyplot as plt
#matplotlib inline
import numpy as np
import os
import scipy as sp
import subprocess
import sys
import time

np.random.seed(3452453)
```

To get the paths right, you will need to run the notebook in a clone of the InfinitesimalJackknifeWorkbench repository. In the same folder where InfinitesimalJackknifeWorkbench was cloned, you will also need to clone the genomic_time_series_bnp and LinearResponseVariationalBayes.py repositories. In the same folder where the InfinitesimalJackknifeWorkbench was cloned, run:

```
git clone https://github.com/NelleV/genomic_time_series_bnp.git
```

We also need the LinearResponseVariationalBayes.py repository:

```
git clone https://github.com/rgiordan/LinearResponseVariationalBayes.py.git
```
Load data and define regressors

To download the data, navigate to the data folder in the genomics repo, e.g.:  

```bash
cd ../../genomic_time_series_bnp/data
```

and run

```bash
make
```

which will download the data from Shoemaker et al. [2015]. We also normalize the data as described in Shoemaker et al. [2015]. The differential analysis tool EDGE (Storey et al. [2005]) is used to extract the 1000 genes that are the most differentially expressed between the infected and the control mice. Our analysis below focuses on only the top 1000 genes.

To extract these 1000 genes that we use, navigate to the /src/exploratory analysis folder in the genomic_time_series_bnp repository, and run make.

For more details, see https://github.com/NelleV/genomic_time_series_bnp. The `load_genomics_data` function below will take care of loading the data into this notebook.

```python
y_train, y_test, timepoints = loading_data_utils.load_genomics_data(
os.path.join(genomic_time_series_dir),
  split_test_train=True,
  train_index_file=mixture_lib_dir + './train_indx.npy')
```

```
loading data from: /home/runjing_liu/Documents/BNP/InfinitesimalJackknifeWorkbench/../genomic_time_series_bnp/data/shoemaker2015reprocessed
```

```
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)
```
number of genes in training set:
700
number of genes in test set:
300

Each gene $y_x$ has 42 observations. Observations are made at 14 timepoints, with 3 replicates at each timepoints.

```python
n_t = len(timepoints)
n_t_unique = len(np.unique(timepoints))
print('timepoints: 
 ' + str(timepoints) + '
')
print('Distinct timepoints: 
' + str(np.sort(np.unique(timepoints))) + '
')
print('Number of distinct timepoints:' + str(n_t_unique))
```

timepoints:
[  0.   0.   0.   3.   3.   3.   6.   6.   6.   9.   9.   9.  12.  12.
  12.  18.  18.  18.  24.  24.  24.  30.  30.  30.  36.  36.  36.  48.
  48.  48.  60.  60.  60.  72.  72.  72.  72.  120. 120. 120. 168. 168.
  168. 168. 168. 168. 168. 168.]

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
f, axarr = plt.subplots(2, 3, figsize=(15,8))
gene_indx = np.sort(np.random.choice(n_train, 6))

for i in range(6):
    n = gene_indx[i]
    axarr[int(np.floor(i / 3)), i % 3].plot(timepoints, y_train[n, :].T, '+', color = 'blue');
    axarr[int(np.floor(i / 3)), i % 3].set_ylabel('gene expression')
    axarr[int(np.floor(i / 3)), i % 3].set_xlabel('time')
    axarr[int(np.floor(i / 3)), i % 3].set_title('gene number {}'.format(n))

f.tight_layout()
```
Define a set of regressions.

We model the time course using cubic B-splines. Let $\alpha$ 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 $n_x \times n_x \times \alpha$ matrix $X_{df}$, where the each column of $X_{df}$ is a B-spline basis vector evaluated at the $n_x$ 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, 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 $t = 72, 120, 168$, respectively, and 0 otherwise. We do this to avoid numerical issues in the matrix $X^TX$.

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:

```python
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)
    axarr[int(np.floor(i / 3)), i % 3].plot(timepoints, _regressors);
    axarr[int(np.floor(i / 3)), i % 3].set_xlabel('time')
    axarr[int(np.floor(i / 3)), i % 3].set_ylabel('B-spline value')
    axarr[int(np.floor(i / 3)), i % 3].set_title('B-spline basis when df = {}'.format(df))
    i += 1
f.tight_layout()
```
We display the regressor matrix below.

```python
df = 7
regressors = spline_bases_lib.get_genomics_spline_basis(timepoints, df=7)
plt.matshow(regressors.T)
plt.ylabel('basis')
plt.xlabel('timepoint and replicate')
plt.title('The (transposed) regressor matrix when df = {}
'.format(df))
```

For the rest of the current notebook, we shall take the degrees of freedom, \( \alpha \), to be 7.

**The first stage: regression**

With the regressor \( X \) defined above, for each gene \( g \) we model \( P(\gamma_g | \beta_g, \sigma_g^2) \sim \mathcal{N}(\gamma_g | X \beta_g, \sigma_g^2) \). In the second stage, we will cluster \( \beta_g \) taking into account its uncertainty on each gene. To do this, in this first stage we 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 \( q \left( \sigma^2, \beta, \eta \right) \) to the posterior \( P \left( \beta, \sigma^2 | y \right) \).

In particular, we take \( q \left( \sigma^2, \beta, \eta \right) = q^* \left( \sigma^2 \right) q^* \left( \beta \right) \), where \( q^* \left( \sigma^2 \right) \) is a dirac delta function, and we optimize over its location parameter, \( q^* \left( \beta \right) \) 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,

\[
q^*(\beta_g) = \mathcal{N} \left( \beta_g \mid \left( X^T X \right)^{-1} X^T y, \hat{\sigma}_g^2 \left( X^T X \right)^{-1} \right)
\]

\[
q^*(\sigma^2_g) = \delta(\sigma^2_g = \hat{\sigma}_g^2)
\]

where \( \hat{\sigma}_g = \frac{1}{n - n_g} \| y - X \left( X^T X \right)^{-1} X^T y \|_2^2 \).

The advantage of the MVFB construction is that \( \hat{\eta}_g \) for \( g = 1, \ldots, n_g \) satisfies set of \( n_g \) independent M-estimation objectives, allowing us to apply our infinitesimal jackknife results. Specifically, defining \( \theta_{reg} := (\eta_1, \ldots, \eta_{n_g}) \), we wish to minimize

\[
F_{reg} \left( \theta_{reg}, \alpha \right) = \sum_{g=1}^{n_g} KL \left( q \left( \sigma^2_g, \beta_g, \eta_g \right) || P \left( \beta_g, \sigma^2_g | y \right) \right)
\]

\[
= -\sum_{g=1}^{n_g} E_q \left[ \log P \left( \beta_g, \sigma^2_g | y \right) \right] + E_q \left[ \log q \left( \beta_g, \sigma^2_g | y \right) \right]
\]

\[
:= \sum_{g=1}^{n_g} F_{reg,g} \left( \eta_g, \alpha \right).
\]

Our M-estimator, then, is

\[
\frac{\partial F_{reg} \left( \theta_{reg}, \alpha \right)}{\partial \theta_{reg}} = 0.
\]

We now instantiate a class that contains these regression results and set the optimal parameters.

```python
# the regression class
def new_params(y_train, regressors):
    reg = reg_lib.Regression(y_train, regressors)

    # set the optimal regression parameters
    reg.set_regression_params()
```

The class `reg` above contains 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 \( \sigma^2_g \), as described above.

Here are what some of the fits look like. Each regression produces a prediction \( \hat{y}_g := X E_q \left[ \beta_g \right] \), plotted with a heavy red line. The light red are predictions when \( \beta_g \) is drawn from \( q^* \left( \beta_g \right) \); the spread of the light red is intended to give a sense of the covariance of \( \beta_g \).
Transforming the regression coefficients

We are interested in the pattern of gene expression, not the absolute level, so we wish to cluster $\hat{y}_k - \bar{y}_k$, where $\bar{y}_k$ is the average over time points. Noting that the $n_t \times n_t$ matrix $\text{Cov}_{\hat{y}_k} (\hat{y}_k - \bar{y}_k)$ is rank-deficient because we have subtracted the mean, the final step is to rotate $\hat{y}_k - \bar{y}_k$ into a basis where the zero eigenvector is a principle axis and then drop that component.
Call these transformed regression coefficients \( \gamma_k \) and observe that \( \text{Cov}_q(\gamma_k) \) has a closed form and is full-rank. It is these \( \gamma_k \)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.

```python
# we get the matrix that does the transformation
transform_mat, unrotate_transform_mat = rm_lib.get_reversible_predict_and_demean_matrix(regs)
trans_obs_dim = transform_mat.shape[0]

# apply the transformation
transformed_reg_params = reg_lib.get_regression_array_params(n_train, transform_mat.shape[0])

beta_mean = regs.reg_params['beta_mean'].get()
beta_cov = regs.reg_params['beta_info'].apply_matrix_function(np.linalg.inv)
y_info = regs.reg_params['y_info'].get()
rm_lib.multiply_by_matrix(beta_mean, beta_cov, y_info, transformed_reg_params, transform_mat)
```

If \( T \) is the matrix that effects the transformation, then

\[
\mathbb{E}_q(\gamma_k) = T \mathbb{E}_q(\beta_k) \\
\text{Cov}_q(\gamma_k) = T \text{Cov}_q(\beta_k) T^T
\]

We now visualize the transformed coefficients and their uncertainty.

```python
f, axarr = plt.subplots(2, 3, figsize=(15, 8))

transformed_beta = transformed_reg_params['beta_mean'].get()
transformed_beta_info = transformed_reg_params['beta_info'].get()

for i in range(6):
    n = gene_indx[i]
    axarr[int(np.floor(i / 3)), i % 3].plot(transformed_beta[n, :], color='red');
    axarr[int(np.floor(i / 3)), i % 3].set_ylabel('transformed coefficient')
    axarr[int(np.floor(i / 3)), i % 3].set_xlabel('index')
    axarr[int(np.floor(i / 3)), i % 3].set_title('gene number {}'.format(n))

    # draw from the variational distribution, to plot uncertainties
    for j in range(30):
        transformed_beta_draw = np.random.multivariate_normal(transformed_beta[n, :], np.linalg.inv(transformed_beta_info[n]))
        axarr[int(np.floor(i / 3)), i % 3].plot(transformed_beta_draw, color='red', alpha=0.08);

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, $\gamma_k$, that we cluster in the second stage.

The second stage: clustering

We now define a clustering problem for the $\gamma_k$. Let $n_k$ be the number of clusters, and $\mu_k, \ldots, \mu_{n_k}$ be the cluster centers. Also let $z_{nk}$ be the binary indicator for the $g$th gene belonging to cluster $k$. We then define the following generative model

\[
P(\pi) = \text{Dirichlet}(\omega)
\]
\[
P(\mu_k) = N(\mu_k | 0, \Sigma_0) \quad \text{for} \quad k = 1, \ldots, n_k
\]
\[
P(z_{nk} = 1 | \pi_k) = \pi_k \quad \text{for} \quad k = 1, \ldots, n_k; \quad n = 1, \ldots, n_g
\]
\[
P(y_g | z_{nk} = 1, \mu_k, \Sigma_k) = N(\gamma_k \nu_k + \text{Cov}_k(\gamma_k) + \epsilon I_{n-1}) \quad \text{for} \quad k = 1, \ldots, n_k; \quad n = 1, \ldots, n_g
\]

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 $\theta_{clust} := (\mu, \pi)$. This defines an optimization objective that we seek to minimize:

\[
F_{clust}(\theta_{clust}, \theta_{reg}) = -\sum_{g=1}^G E_{q^*} \left\{ \log P(y_g | \nu_g, \mu, \pi, z_g) - \log P(z_g | \pi) \right\} - \log P(\mu) - \log P(\pi)
\]

which, for every value of $\theta_{reg}$, we expect to satisfy

\[
\frac{\partial F_{clust}(\theta_{clust}, \theta_{reg})}{\partial \theta_{clust}} = 0.
\]

Note that $\theta_{clust}$ involves only the “global” parameters $\mu$ and $\pi$. We did take a variational distribution for the $z_{nk}$, represented by independent Bernoulli distribution, but the optimal $q^*_k$ can be written as a function of $\mu$ and $\pi$. Hence, our optimization objective only involves these global parameters.
In our experiment, the number of clusters $n_k$ was chosen to be 18. We set $\omega$ to be the ones vector of length $n_k$. The prior info for the cluster centers $\Sigma_0$ is $10^{-5} \times I$. $\epsilon$ was set to be 0.1. We now define a class to perform the mixture modeling.

```python
epsilon = 0.1

In our experiment, the number of clusters $n_k$ was chosen to be 18. We set $\omega$ to be the ones vector of length $n_k$. The prior info for the cluster centers $\Sigma_0$ is $10^{-5} \times I$. $\epsilon$ was set to be 0.1. We now define a class to perform the mixture modeling.

gmm = rm_lib.GMM(num_components, prior_params, regs,
                 transform_mat,
                 inflate_coef_cov=None,
                 cov_regularization=epsilon)
```

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 $k$.

```python
def get_log_lik_nk(params, x, x_infos):
    pv = params.values
    loc_log_lik = -0.5 * (-2 * np.einsum('ni,kj,nij->nk', x, pv['means'], x_infos) + np.einsum('ki,kj,nij->nk', pv['means'], pv['means'], x_infos))
    log_weights = np.log(pv['weights'][0, :])
    log_lik_by_nk = loc_log_lik + log_weights.T
    return log_lik_by_nk
```

We can then optimize for $q_\ast$, which is parametrized by its mean $E_{q_\ast}[z]$. We note that this update has a closed form given $\theta_{datum}$. So there is no need to solve an optimization problem to find $q_\ast(z)$. We additionally note that we do not use the EM algorithm, which we found to have exhibit extremely poor convergence rates. Rather, we set $q_\ast(z)$ to its optimal value given $\theta_{datum}$ and return the objective as a function of $\theta_{datum}$ alone, allowing the use of more general and higher-quality optimization routines.

```python
def get_e_z(log_lik_by_nk):
    log_const = sp.misc.logsumexp(log_lik_by_nk, axis=1)
    e_z = np.exp(log_lik_by_nk - log_const[:, None])
    return e_z
```

With the optimal parameters for $z_{nk}$, we combine the likelihood term with the prior and entropy terms.
def get_kl(self):
    # This function assumes the regression parameters have already been
    # initialized with self.cache_transform.
    log_lik_by_nk = get_log_lik_nk(
        self.params,
        x=self.transformed_reg_params['beta_mean'].get(),
        x_infos=self.transformed_reg_params['beta_info'].get())
    e_z = get_e_z(log_lik_by_nk)
    
    log_prior = get_log_prior(self.prior_params, self.params)
    if self.use_obs_weights:
        w = self.obs_weights.get()
        return get_kl(log_lik_by_nk, e_z, log_prior, w)
    else:
        return get_kl(log_lik_by_nk, e_z, log_prior, w=None)

print(inspect.getsource(rm_lib.get_kl))

def get_kl(log_lik_by_nk, e_z, log_prior, w=None):
    num_obs = log_lik_by_nk.shape[0]
    if w is None:
        return -1 * (np.sum(e_z * log_lik_by_nk) +
                      np.sum(ef.multinoulli_entropy(e_z)) +
                      log_prior) / num_obs
    else:
        return -1 * (np.sum(np.sum(e_z * log_lik_by_nk, axis=1) * w) +
                      np.sum(ef.multinoulli_entropy(e_z) * w) +
                      log_prior) / num_obs

Optimization

For optimization we make extensive use of the autograd (https://github.com/HIPS/autograd) and
LinearResponseVariationalBayes.py (https://github.com/rgiordan/LinearResponseVariationalBayes.py) libraries. In particular, see
the SparseObjectives submodule
(https://github.com/rgiordan/LinearResponseVariationalBayes.py/blob/master/LinearResponseVariationalBayes/SparseObjectives.py)
of LinearResponseVariationalBayes.py for more details about the objective class.

First, we do a k-means initialization.

gmm.kmeans_init(n_kmeans_init=50)
init_x = gmm.params.get_free()
print('Parameter dimension: ', len(init_x))

Parameter dimension: 179

Starting from the k-means initialization, we run the Newton conjugate gradient trust-region algorithm until reaching a loose
convergence tolerance threshold; at this point, we compute the Hessian for the clustering loss $F_{clust} (\theta_{clust}, \theta_{reg})$ with respect to
the cluster parameters $\theta_{clust}$, and use this Hessian as a preconditioner for subsequent Newton trust region steps. This process
was iterated until convergence at high precision.

We found that the infinitesimal jackknife matched the result of re-optimizing more closely when using a preconditioner.

Optimization without a preconditioner:
```python
def optimize(self, init_x, maxiter=500, gtol=1e-6):
    gmm_opt = osp.optimize.minimize(
        lambda par: self.obj.fun_free(par, verbose=True),
        x0=init_x,
        jac=self.obj.fun_free_grad,
        hessp=self.obj.fun_free_hvp,
        method='trust-ncg',
        options={'maxiter': maxiter, 'gtol': gtol})
    return gmm_opt

Optimization with a preconditioner:
```
```
running Newton steps
Iter 0 value: -41.15268412531631
Iter 1 value: -41.25713962866718
Iter 2 value: -41.28546872292904
Iter 3 value: -41.18829707809087
Iter 4 value: -41.29623969671619
Iter 5 value: -41.29922343080497
getting preconditioner
done. Elapsed time: 11.502126932144165

# Continue optimizing with the preconditioner
opt_time = time.time()
gmm_opt, gmm_opt_x = gmm.optimize_fully(gmm_opt.x, init_preconditioner, verbose=True)
opt_time = time.time() - opt_time

# save the optimal parameters
params.set_free(gmm_opt_x)
params_comb_free = gmm.comb_params.get_free()
print(gmm_opt['gmm_opt_cond'].message)
print('done. Elapsed time: ', opt_time)

Using init_preconditioner.
Preconditioned iteration 1
   Running preconditioned optimization.
Iter 6 value: -41.29922343080496
Iter 7 value: -41.29961590863672
Iter 8 value: -41.29974321768379
Iter 9 value: -41.30012373534639
Iter 10 value: -41.28315090402641
Iter 11 value: -41.30001049881842
Iter 12 value: -41.30034994659932
Iter 13 value: -41.30065630416259
Iter 14 value: -41.30121599881144
Iter 15 value: -41.30189228440574
Iter 16 value: -41.29422678443102
Iter 17 value: -41.3028655631816
Iter 18 value: -41.30373934858740
Iter 19 value: -41.30408652654892
Iter 20 value: -41.30410179047606
Iter 21 value: -41.30410358269038
Preconditioned iteration 2
   Getting Hessian and preconditioner.
   Running preconditioned optimization.
Iter 22 value: -41.30410358269038
Iter 23 value: -41.30410358276909
Preconditioned iteration 3
   Getting Hessian and preconditioner.
   Running preconditioned optimization.
Iter 24 value: -41.30410358276909
Iter 25 value: -41.30410358276987
Converged.
A bad approximation caused failure to predict improvement.
done. Elapsed time: 214.79687118530273

# Save the optimal result.
gmm_full = deepcopy(gmm)
```
The Infinitesimal Jackknife

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 \( w \) by observing that, for each \( g \), the term \( \log P \left( \beta_g \mid \sigma_g^2 \right) \) decomposes into a sum over time points:

\[
F_{reg, g} (\eta_g, \alpha, w) := - \sum_{t=1}^n w_t \left( -\frac{1}{2} \sigma_g^{-2} (y_{gt} - (X_{\beta_g})_t)^2 - \frac{1}{2} \log \sigma_g^2 \right) + E_g \left[ \log q (\beta_g, \sigma_g^2 \mid \eta_g) \right].
\]

We naturally define \( F_{reg} (\theta_{reg}, \alpha, w) := \sum_{g=1}^G F_{reg, g} (\eta_g, \alpha, w) \). By defining \( \theta = (\theta_{clust}, \theta_{reg}) \), we then have an M-estimator

\[
G (\theta, w, \alpha) := \left[ \begin{array}{c}
\frac{\partial F_{reg}(\theta_{reg}, \alpha, w)}{\partial \theta_{reg}} \\
\frac{\partial F_{reg}(\theta_{reg}, \alpha, w)}{\partial \theta_{clust}} \\
\end{array} \right] = 0.
\]

And we can apply the IJ to approximate the leaving out of various timepoints.

Computing the linear response objects

We get the quantities necessary for the infinitesimal jackknife. This 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 \( \theta_{reg}(w) \) is extremely fast.

```python
# Get a linear response prediction object
gmm_predictor = rm_lib.GMMPredictor(gmm)
get_predictor_time = time.time() - get_predictor_time
print('Predictors time: ', get_predictor_time)

Getting full Hessian.
GMM hessian time:  7.866747617721558
Cross hessian time:  150.1160533428192
Regression hessian time:  161.10200667381287
Factorizing Hessian.

/home/runjing_liu/anaconda3/lib/python3.6/site-packages/scipy/sparse/linalg/dsolve/linsolve.py:253:
SparseEfficiencyWarning: splu requires CSC matrix format
warn('splu requires CSC matrix format', SparseEfficiencyWarning)

Getting t Jacobian.
Getting obs Jacobian.
Predictors time:  356.4892942905426

print('Full Hessian dimension: ', gmm_predictor.full_hess.shape)

Full Hessian dimension:  (46379, 46379)
```

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. \( H_1 \) can be computed in blocks:

\[
H_1 = \begin{pmatrix}
\nabla^2_{\theta_{reg}} F_{reg} & 0 \\
\nabla^2_{\theta_{reg}} F_{clust} & \nabla^2_{\theta_{clust}} F_{clust}
\end{pmatrix}
\]
Re-optimizing with a timepoint left out

We first calculate a preconditioner based on $\nabla^2_{\theta_{\text{clus}}} F_{\text{clus}}$ to speed up and improve re-optimizing.

```python
# Get a preconditioner from the original optimum.
precond_time = time.time()
preconditioner = rm_lib.get_preconditioner(gmm_predictor.gmm_hess)
precond_time = time.time() - precond_time
print('Preconditioner time: ', precond_time)
```

Preconditioner time: 0.0348508358001709

We choose a time point to leave out and define a weight vector $w_t$.

```python
leave_out_time = 3
w_t = np.ones(gmm.regs.y_obs_dim)
w_t[np.argwhere(timepoints == leave_out_time)] = 0
```

We now re-optimize starting from the original optimum using the preconditioner.
Using init_preconditioner.
Preconditioned iteration 1
  Running preconditioned optimization.
Iter 0 value: -41.097872578016141
Iter 1 value: -41.14529660785101
Iter 2 value: -41.17783816950253
Iter 3 value: -41.20712335138064
Iter 4 value: -41.21117629189541
Iter 5 value: -41.21260774672933
Iter 6 value: -41.218451437548519
Iter 7 value: -41.21388679385335
Iter 8 value: -41.217165795447876
Iter 9 value: -41.21368785364836
Iter 10 value: -41.219297676874186
Iter 11 value: -41.223911451916306
Iter 12 value: -41.22361123439994
Iter 13 value: -41.22365753630111
Iter 14 value: -41.223657828547445
Preconditioned iteration 2
  Getting Hessian and preconditioner.
  Running preconditioned optimization.
Iter 15 value: -41.22365782854743
Iter 16 value: -41.223657828573835
Preconditioned iteration 3
  Getting Hessian and preconditioner.
  Running preconditioned optimization.
Iter 17 value: -41.22365782857385
Iter 18 value: -41.22365782857385
Converged.
A bad approximation caused failure to predict improvement.
time for re-optimization: 184.86519289016724

Computing $\theta_{IJ}$

After the Hessian is computed and factorized, calculating $\theta_{IJ}$ takes a fraction of a second:

```python
# the parameters predicted by the IJ
lr_time = time.time()
w_pred_comb_par = gmm.predictor.predict_for_time_weights(w_t)

print('time to compute IJ (with the Hessian precomputed): ', time.time() - lr_time)
```
This is because each $\theta_{IJ}$ requires only the solution of a sparse, factorized linear system.

```python
print(inspect.getsource(gmm_predictor.predict_for_time_weights))
```

```python
def predict_for_time_weights(self, w_t):
    pred_diff = -1 * self.solve_full_hess(self.t_jac @ (w_t - 1))
    return self.comb_free_param + pred_diff
```

**Comparsion**

```python
def comparison_plot(x, y):
    plt.plot(x, y, 'k.')[
    plt.plot(x, x, 'r')
```

We first examine the difference in parameter values before and after leaving out a timepoint. We compare this difference as computed by re-optimizing against the difference predicted by our linear approximation.

```python
plt.figure()
comparison_plot((w_opt_comb_par - opt_params_comb_free)[gmm_predictor.gmm_inds],
                (w_pred_comb_par - opt_params_comb_free)[gmm_predictor.gmm_inds])
plt.title('Comparison of mixture model parameters ($\theta_{\text{clust}}$)')
plt.xlabel('Diff after re-optimizing')
plt.ylabel('Diff predicted by IJ')

plt.figure()
comparison_plot((w_opt_comb_par - opt_params_comb_free)[gmm_predictor.reg_inds],
                (w_pred_comb_par - opt_params_comb_free)[gmm_predictor.reg_inds])
plt.title('Comparisons regression parameters ($\theta_{\text{reg}}$)')
plt.xlabel('Diff after re-optimizing')
plt.ylabel('Diff predicted by IJ')
```

![Comparison of mixture model parameters ($\theta_{\text{clust}}$)](image.png)
We now calculate the prediction error on the held-out time point. For each gene, we predict its timecourse with 
\( \hat{y}_g = \sum_{k=1}^{n_k} X E_{q^*}[\beta_k] \cdot E_{q^*}[z_k] \). Note that \( q^* \), and therefore our estimate \( \hat{y}_g \), depends on the parameters \( \theta \), which were computed either by re-optimizing, or by using the infinitesimal jackknife. We compute the error \((y_t - \hat{y}_g)^2\) on the held-out timepoints \( t \), and compare the error when \( \hat{y}_g \) is obtained by re-optimizing, or by using the infinitesimal jackknife.

```python
# get MSE using the parameters from the full jackknife
jack_mses = get_mse_utils.get_jack_mses(gmm, np.array([w_opt_comb_par]), np.array([w_t]), unrotate_transform_mat)

# get MSE using the parameters from the IJ
lr_mses = get_mse_utils.get_jack_mses(gmm, np.array([w_pred_comb_par]), np.array([w_t]), unrotate_transform_mat)
```

100%|██████████| 1/1 [00:00<00:00,  7.17it/s]
100%|██████████| 1/1 [00:00<00:00,  5.11it/s]

collection_plot(lr_mses.flatten(), jack_mses.flatten())
plt.xlabel('ij predicted mse')
plt.ylabel('full jackknife mse')

<matplotlib.text.Text at 0x7f360dda36d8>
We do this procedure several times to get a CV estimate for the MSE. I.e., we iterate over leaving out different timepoints, each time getting MSE estimates on the held-out timepoints. We can also leave out multiple timepoints. Finally, we also vary the degree of freedom $\alpha$ in our analysis, each time getting a CV estimate for the test error. We can use this CV error to choose the degrees of freedom $\alpha$. We compare the $\alpha$ chosen from using the true CV error obtained from re-optimizing to the $\alpha$ chosen from using the approximate CV error obtained by our IJ procedure.

Bibliography

J. E. Shoemaker, S. Fukuyama, A. J. Eisfeld, D. Zhao, E. Kawakami, S. Sakabe, T. Maemura, T. Gorai, H. Katsura, Y. Muramoto, S. Watanabe, T. Watanabe, K. Fuji, Y. Matsuoka, H. Kitano, and Y. Kawaoka. An Ultrasensitive Mechanism Regulates Influenza Virus-Induced Inflammation. PLoS Pathogens, 11(6):1–25, 2015

John D Storey, Wenzhong Xiao, Jeffrey T Leek, Ronald G Tompkins, and Ronald W Davis. Significance analysis of time course microarray experiments. Proceedings of the National Academy of Sciences of the United States of America, 102(36):12837–42, 2005. ISSN 0027-8424. doi: 10.1073/pnas.0504609102. URL http://www.pnas.org/content/102/36/12837.abstract.