Estimating the Error of Randomized Newton Methods: A Bootstrap Approach

Jessie X.T. Chen [1] **Miles E. Lopes** [2]

Abstract

Randomized Newton methods have recently become the focus of intense research activity in large-scale and distributed optimization. In general, these methods are based on a "computation-accuracy trade-off", which allows the user to gain scalability in exchange for error in the solution. However, the user does not know how much error is created by the randomized approximation, which can be detrimental in two ways: On one hand, the user may try to assess the unknown error with theoretical worst-case error bounds, but this approach is impractical when the bounds involve unknown constants, and it often leads to excessive computation. On the other hand, the user may select the "sketch size" and stopping criteria in a heuristic manner, but this can lead to unreliable results. Motivated by these difficulties, we show how bootstrapping can be used to *directly estimate the unknown error*, which prevents excessive computation, and offers more confidence about the quality of a randomized solution.

1. Introduction

In recent years, there has been a surge of interest in using randomized approximations to accelerate Newton methods in large-scale and distributed optimization (e.g. Shamir et al., 2014; Erdogdu & Montanari, 2015; Zhang & Lin, 2015; Byrd et al., 2016; Pilanci & Wainwright, 2016; Reddi et al., 2016; Xu et al., 2016; Pilanci & Wainwright, 2017; Wang et al., 2017; 2018; Dünner et al., 2018; Gupta et al., 2019; Li et al., 2020; Roosta-Khorasani & Mahoney, 2019, among many others). At a high level, this rapid development of research has been driven by the fact that computing a Newton step to high precision can be very costly or infeasible in large-scale problems. Instead, randomized approaches make it possible to overcome this challenge by exchanging some degree of accuracy in return for substantial reductions in both processing and communication costs. However, one of the common difficulties faced by users in applying randomized Newton methods is that they do not know how far a randomized Newton step might stray from an exact one.

To deal with the uncertainty in the quality of a randomized solution, users have generally relied on two strategies, which are to (1) assess the unknown error with theoretical worst-case error bounds, or (2) use heuristic rules to select the "sketch size" or number of iterations. But unfortunately, both of these options can be detrimental to the overall performance of randomized algorithms. Indeed, the first option of worst-case analysis is typically inefficient from a computational standpoint, because it can substantially overestimate the actual error of a solution — and hence mislead the user to select an excessive sketch size or number of iterations. Also, this option is limited by the fact that theoretical error bounds often involve unspecified constants or unknown parameters that make it difficult to extract a *numerical* error bound. Meanwhile, the use of heuristics is undesirable from the standpoint of reliability, and it can create difficulty in tuning downstream elements of computational pipeline.

As a way of handling these difficulties, we apply the statistical technique of bootstrapping to estimate the errors of randomized Newton methods. In particular, this approach avoids the conservativeness of worst-case analysis by *directly estimating the actual error* of a given randomized solution. In addition, the bootstrap provides the user with more flexibility in the choice of error metric than is typically available with worst-case error bounds. Next, in comparison to heuristic rules, this approach offers more reliability, by giving the user a systematic procedure that is theoretically justified. Furthermore, the bootstrap is highly scalable in this context (due to its embarrassingly parallel structure), and it is promising from an empirical standpoint as well.

1.1. Background and Setting

Let $a_1, \dots, a_n \in \mathbb{R}^d$ be the rows of a matrix $A \in \mathbb{R}^{n\times d}$ with $n \gg d$, and let $b_1, \dots, b_n$ be the entries of a vector $b \in \mathbb{R}^n$. Consider the problem of minimizing an objective function $f : \mathbb{R}^d \to \mathbb{R}$ of the form

$$f(w) = \tfrac{1}{n}\textstyle\sum_{i=1}^n \varphi(a_i^\top w, b_i) + r(w), \tag{1}$$

[1]Department of Mathematics, University of California, Davis [2]Department of Statistics, University of California, Davis. Correspondence to: <melopes@ucdavis.edu>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

where the functions $\varphi : \mathbb{R}^2 \to \mathbb{R}$ and $r : \mathbb{R}^d \to \mathbb{R}$ are twice differentiable, and φ is convex in its first argument. Objective functions of the form (1) are ubiquitous in machine learning, where the points $\{(a_i, b_i)\}_{i=1}^n$ typically play the role of n observations, and f is viewed as a measure of empirical risk. Some of the most well known examples occur in the fitting of regularized *generalized linear models* and *support vector machines*, where φ corresponds to a loss function, and $r(w)$ is a regularization function (cf. McCullagh & Nelder, 1989; Chapelle, 2007). Another important class of examples arises in solving *linear programs* of the form $\min\{c^\top w \,|\, Aw \leq b\}$ by interior point methods, where $r(w) = c^\top w$ for some cost vector $c \in \mathbb{R}^d$, and φ corresponds to a logarithmic barrier function. (We refer to (Pilanci & Wainwright, 2017) for more detailed examples along these lines.)

Classical Netwon method. When a classical version of Newton's method is applied to minimize (1), the k-th iterate $w_k \in \mathbb{R}^d$ is computed using the gradient, denoted

$$\begin{aligned} g_k &= \nabla f(w_k) \\ &= \tfrac{1}{n}\textstyle\sum_{i=1}^n \partial_1 \varphi(a_i^\top w_k, b_i) a_i + \nabla r(w_k), \end{aligned}$$

as well as the Hessian, denoted

$$\begin{aligned} H_k &= \nabla^2 f(w_k) \\ &= \tfrac{1}{n}\textstyle\sum_{i=1}^n \partial_1^2 \varphi(a_i^\top w_k, b_i) a_i a_i^\top + \nabla^2 r(w_k), \end{aligned}$$

where ∂_1 is the partial derivative with respect to the first argument. More specifically, if H_k is invertible, and if η_k is a step size parameter, then the update rule is

$$w_{k+1} = w_k - \eta_k H_k^{-1} g_k.$$

However, in many cases, it is prohibitive to implement this update rule to high precision, either because n is very large, or because the observations $\{(a_i, b_i)\}_{i=1}^n$ may be stored in a distributed manner, which can lead to high communication costs.

In order to overcome these bottlenecks, randomized Newton methods seek to compute efficient approximations of $H_k^{-1} g_k$. In particular, these approximations are able to leverage the fact that the function (1) has a Hessian that can be theoretically decomposed as

$$H_k = C_k^\top C_k + \nabla^2 r(w_k), \tag{2}$$

where $C_k \in \mathbb{R}^{n\times d}$ is a matrix given by

$$C_k = \tfrac{1}{\sqrt{n}} D_k A \quad \text{with} \quad D_k = \text{diag}\{\sqrt{\partial_1^2 \varphi(a_i^\top w_k, b_i)}\}_{i=1}^n.$$

Below, we briefly review two well-known examples of such randomized algorithms, called NEWTON SKETCH (Pilanci & Wainwright, 2017) and GIANT (Wang et al., 2018), since they will be the focus of our work later on.

The NEWTON SKETCH algorithm. The core idea of the NEWTON SKETCH algorithm is to randomly transform the matrix C_k into a much shorter version $\tilde{C}_k \in \mathbb{R}^{t\times d}$ that can be handled more efficiently, where $t \ll n$. As a matter of terminology, the matrix $\tilde{C}_k$ is referred to as a "sketch" of C_k, and t is known as the "sketch size". In detail, the random transformation is implemented with a random "sketching matrix" $S_k \in \mathbb{R}^{t\times n}$ so that $\tilde{C}_k = S_k C_k$, and in turn, this leads to a sketched Hessian matrix defined as

$$\widetilde{H}_k = \tilde{C}_k^\top \tilde{C}_k + \nabla^2 r(w_k). \tag{3}$$

Accordingly, this algorithm revises the classical Newton method by using the following randomized Newton step,[1]

$$w_{k+1} = w_k - \eta_k (\widetilde{H}_k)^{-1} g_k. \tag{4}$$

In order to ensure that $\widetilde{H}_k$ provides an effective approximation to H_k, the sketching matrix S_k is commonly generated so that it has i.i.d. rows and satisfies the relation $\mathbb{E}[S_k^\top S_k] = I_n$. For example, when S_k is a *uniform sampling matrix*, the rows of S_k are generated as i.i.d. samples from the uniform distribution on the set $\{\sqrt{n/t}\, e_1, \ldots, \sqrt{n/t}\, e_n\} \subset \mathbb{R}^n$, where $e_1, \ldots, e_n$ are the canonical basis vectors.

The GIANT algorithm. When data are stored on a distributed system, controlling the communication cost between different machines (workers) is often of paramount importance. As a way of reducing the communication involved in computing a Newton step, the GIANT algorithm uses an approximation to H_k^{-1} derived from the harmonic mean of local Hessian matrices.

To be more specific, suppose random samples from $\{(a_i, b_i)\}_{i=1}^n$ are evenly distributed across m different workers, and the j-th worker holds samples indexed by $\mathcal{I}_j$. Also, in this context, we will denote the "local sample size" as $t = |\mathcal{I}_j| = n/m$, since it plays a role that is analogous to the sketch size in the NEWTON SKETCH algorithm. Next, if the matrix $\tilde{C}_{k,j} \in \mathbb{R}^{t\times d}$ is defined to have rows given by the set of vectors $\{\frac{1}{\sqrt{t}}[D_k]_{ii} a_i\}_{i\in\mathcal{I}_j}$, then the j-th local approximate Hessian matrix at the k-th iteration is defined by

$$\widetilde{H}_{k,j} = \tilde{C}_{k,j}^\top \tilde{C}_{k,j} + \nabla^2 r(w_k). \tag{5}$$

Once these local Hessians have been computed, they are aggregated in the update rule

$$w_{k+1} = w_k - \tfrac{\eta_k}{m} \textstyle\sum_{j=1}^m (\widetilde{H}_{k,j})^{-1} g_k, \tag{6}$$

[1] We will follow the convention that if M is a singular square matrix, then M^{-1} refers to the pseudoinverse. Nevertheless, the approximate Hessian matrices under consideration will typically be invertible in our settings of interest.

which can be (conceptually) interpreted in terms of the inverse of the harmonic mean $\widetilde{H}_k = \big(\frac{1}{m}\sum_{j=1}^m \widetilde{H}_{k,j}^{-1}\big)^{-1}$ of the local Hessians.

1.2. Problem Formulation

In order to study the algorithmic error of randomized Newton methods, our work will focus on the randomness that comes from within the algorithms, and we will always treat the points $\{(a_i, b_i)\}_{i=1}^n$ and the function f as being deterministic. From this perspective, it is important to clarify that an iterate w_k of such algorithms is a random vector, but the exact optimal solution

$$w_{\text{opt}} = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}}\ f(w)$$

is deterministic. Also, it should be noted that g_k, C_k, and H_k are random in general, as they depend on w_k.

Estimating error with respect to Newton step. Let the exact Newton step and its sketched version be denoted as

$$\Delta_k = H_k^{-1} g_k \qquad \text{and} \qquad \widetilde{\Delta}_k = (\widetilde{H}_k)^{-1} g_k. \tag{7}$$

To measure the quality of an iterate w_{k+1}, we may consider the error of $\tilde{\Delta}_k$, denoted as

$$\epsilon_k = \rho(\widetilde{\Delta}_k, \Delta_k), \tag{8}$$

where $\rho(\cdot,\cdot)$ is a generic non-negative measure of error that is chosen by the user. For example, if $\|\cdot\|_\diamond$ is some norm on $\mathbb{R}^d$, then we can take ρ to be the absolute error $\rho(v', v) = \|v' - v\|_\diamond$, or the relative error $\rho(v', v) = \|v' - v\|_\diamond / \|v\|_\diamond$. The error in the Newton step is of particular interest for functions that are locally quadratic near w_{opt}, because the exact Newton method minimizes quadratic functions in a single step.

Due to the fact that the error ϵ_k in the Newton step is a random variable, it is of interest to study its $(1-\alpha)$-quantile, which is defined as the *tighest possible* upper bound on ϵ_k that holds with probability at least $1-\alpha$,

$$q_{\alpha,k} = \inf\Big\{q \in [0,\infty)\ \Big|\ \mathbb{P}\big(\epsilon_k \le q\big) \ge 1-\alpha\Big\}.$$

Since the quantile $q_{\alpha,k}$ is unknown in practice, we aim to construct an estimate $\widehat{q}_{\alpha,k}$, which is intended to satisfy the bound

$$\epsilon_k \le \widehat{q}_{\alpha,k} \tag{9}$$

with probability nearly equal to, or greater than, $1-\alpha$.

Estimating error with respect to Newton decrement. Another way to measure the quality of an iterate w_k is through its optimality gap $f(w_k) - f(w_{\text{opt}})$. To derive a bound on the optimality gap, it is convenient to consider the squared *Newton decrement*

$$\delta_k^2 = g_k^\top H_k^{-1} g_k. \tag{10}$$

The Newton decrement has special significance in the case when f is a strictly convex self-concordant function — which frequently occurs in the context of interior point methods (cf. Nesterov & Nemirovskii, 1994). In particular, functions of this type that have the form (1) are known to occur in connection with ridge regression, regularized logistic regression, and smoothed hinge loss functions (cf. Zhang & Lin, 2015).

When the function f is strictly convex and self-concordant, it is a classical fact that if w_k is any point in the function's domain, then the optimality gap is bounded according to

$$f(w_k) - f(w_{\text{opt}}) \ \le\ \delta_k^2, \tag{11}$$

provided that $\delta_k \le 0.68$ (Boyd & Vandenberghe, 2004, §9.6.3). However, because the exact quantity δ_k^2 is unknown, it is of interest to measure the error of the approximate (randomized) decrement $\widetilde{\delta}_k^2 = g_k^\top \tilde{H}_k^{-1} g_k$. This error is denoted as

$$\varepsilon_k = \varrho(\widetilde{\delta}_k^2, \delta_k^2), \tag{12}$$

where $\varrho(\cdot,\cdot)$ is another non-negative error measure of the user's choice. By analogy with the earlier definition of $q_{\alpha,k}$, the $(1-\alpha)$-quantile of ε_k is defined as

$$\mathsf{q}_{\alpha,k} = \inf\Big\{q \in [0,\infty)\ \Big|\ \mathbb{P}(\varepsilon_k \le q) \ge 1-\alpha\Big\}.$$

Furthermore, since this parameter is unknown, we seek to construct an estimate $\widehat{\mathsf{q}}_{\alpha,k}$ such that the following bound holds with probability nearly equal to, or greater than, $1-\alpha$,

$$\varepsilon_k \le \widehat{\mathsf{q}}_{\alpha,k}. \tag{13}$$

In turn, this will provide a corresponding bound on the optimality gap. For example, when ϱ is chosen to be the relative error $\varrho(\widetilde{\delta}_k^2, \delta_k^2) = |\tilde{\delta}_k^2 - \delta_k^2|/\delta_k^2$, the estimate $\widehat{\mathsf{q}}_{\alpha,k}$ will satisfy

$$f(w_k) - f(w_{\text{opt}}) \ \le\ \tilde{\delta}_k^2\,(1-\widehat{\mathsf{q}}_{\alpha,k})^{-1}$$

provided that $\widehat{\mathsf{q}}_{\alpha,k} < 1$, and (13) holds.

1.3. Related Work and Contributions

For handling error estimation problems that arise in statistical contexts, bootstrap methods provide a very general framework that is broadly applicable and supported by an extensive literature (e.g. Efron, 1979; 1982; Shao & Tu, 2012; Davison & Hinkley, 1997; Chernick, 2011). (See also the beginning of Section 2.1 for a brief description of the basic principle of bootstrap methods.)

However, in the context of randomized sketching algorithms, only a small subset of the literature has given attention to the problem of estimating error, and likewise, error estimation methods have only been developed for a limited number of these algorithms. Up to now, the existing work has dealt primarily with low-rank matrix approximation or matrix multiplication (e.g. Liberty et al., 2007; Woolfe et al., 2008; Halko et al., 2011; Martinsson & Voronin, 2016; Sorensen & Embree, 2016; Duersch & Gu, 2017; Lopes et al., 2019b; 2018; Yu et al., 2018; Lopes et al., 2019a; Tropp et al., 2019; Lopes et al., 2020). (For further discussion of error estimation in randomized numerical linear algebra, as well as connections to bootstrapping, we refer to the forthcoming survey (Martinsson & Tropp, 2020, §4.5-4.6).) Similarly, in the large literature on stochastic gradient descent, relatively few papers have addressed error estimation (e.g. Fang et al., 2018; Fang, 2019; Su & Zhu, 2018; Li et al., 2018; Anastasiou et al., 2019).

With regard to randomized second-order methods, the paper (Lopes et al., 2018) took an initial step in this direction by studying bootstrap error estimation for randomized least-squares algorithms. However, that work focuses only on ordinary least-squares problems, and does not address non-quadratic objective functions or distributed optimization (as our work does). Also, our work applies bootstrap error estimation to a more general class of twice-differentiable objective functions, and deals with a broader set of error measures (based on the Newton step and Newton decrement). Furthermore, whereas the analysis in (Lopes et al., 2018) is asymptotic, we develop non-asymptotic theory in the cases of the NEWTON SKETCH and GIANT algorithms. Lastly, it should be noted that the distributed setting introduces a significant extra theoretical challenge, which is to show that bootstrap error estimation can account for the accumulation of bias in the approximations of several worker machines.

Notation. The $d \times d$ identity matrix is denoted as I_d, and if A is a matrix of a complicated form, its ij entry is sometimes denoted as $[A]_{ij}$. The norms $\|\cdot\|_2$ and $\|\cdot\|_\infty$ refer to the vector ℓ_2-norm and ℓ_∞-norms on Euclidean space, while $\|\cdot\|_\diamond$ refers to an arbitrary norm. The singular values of a generic real matrix A are denoted $\sigma_j(A) \geq \sigma_{j+1}(A)$, with the largest and smallest respectively denoted as $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$. For a list of real numbers $x_1, \ldots, x_B$, their empirical $(1-\alpha)$-quantile is written as quantile$(x_1, \ldots, x_B; 1-\alpha)$. More precisely, this quantity is defined as $\inf\{q \in \mathbb{R} \,|\, F_B(q) \geq 1-\alpha\}$, where $F_B(q)$ is the empirical distribution function $F_B(q) = \frac{1}{B}\sum_{b=1}^{B} 1\{x_b \leq q\}$. Symbols such as $c, c_0, c_1, \ldots$, are used to denote absolute constants whose value may change from line to line. Lastly, the maximum of two real numbers a and b is denoted $a \vee b$.

2. Methods

In this section, we describe two methods for constructing quantile estimates that are designed to satisfy the conditions (9) and (13). Sections 2.2 and 2.3 present the error estimation methods corresponding to NEWTON SKETCH and GIANT respectively. Later on, we discuss the computational cost of error estimation in Section 2.4.

2.1. Bootstrap Methods in a Nutshell

Bootstrap methods are commonly used for error estimation in the following way. Consider an unknown parameter θ that is to be estimated with a statistic $\widehat{\theta}$ that is computed as a function of i.i.d. data $X_1, \ldots, X_t$, say $\widehat{\theta} = h(X_1, \ldots, X_t)$. In order to assess the accuracy of $\widehat{\theta}$, it is desirable to estimate the quantiles of the random variable $\mathfrak{e} = |\widehat{\theta} - \theta|$. However, since the distribution of $\mathfrak{e}$ is unknown, its quantiles can be numerically estimated by generating a collection of "approximate samples", say $\mathfrak{e}_1^*, \ldots, \mathfrak{e}_B^*$, and then using the empirical quantiles of these values.

In this way, the problem of error estimation is reduced to finding a way to generate the approximate samples $\mathfrak{e}_1^*, \ldots, \mathfrak{e}_B^*$, and many ways of doing this have been proposed in the bootstrap literature. The most basic version of these methods is to sample t values, say $X_1^*, \ldots, X_t^*$, with replacement from $X_1, \ldots, X_t$, and define $\mathfrak{e}_1^* = |\widehat{\theta}^* - \widehat{\theta}|$, where $\widehat{\theta}^* = h(X_1^*, \ldots, X_t^*)$. Likewise, this process is repeated independently to obtain $\mathfrak{e}_2^*, \ldots, \mathfrak{e}_B^*$. Roughly speaking, this procedure can be understood as generating $\widehat{\theta}^*$ so that its fluctuations around $\widehat{\theta}$ are statistically similar to the fluctuations of $\widehat{\theta}$ around θ. In the next two subsections, we show how this general idea can be adapted to the contexts of NEWTON SKETCH and GIANT.

2.2. Bootstrap Error Estimation for NEWTON SKETCH

Based on the preceding discussion, we aim to generate approximate samples of the error variables ϵ_k and ε_k for NEWTON SKETCH. Given that the t rows of the sketching matrix $S_k \in \mathbb{R}^{t\times n}$ are commonly generated to be i.i.d. random vectors, it is helpful to think of them as being i.i.d. data points, analogous to $X_1, \ldots, X_t$ described in Section 2.1. Likewise, the sketched Newton step $\tilde{\Delta}_k$ and decrement $\tilde{\delta}_k^2$ can be loosely interpreted as "statistics" that are functions of these i.i.d. data. Therefore, it is natural to consider generating bootstrapped versions $\tilde{\Delta}_k^*$ and $\tilde{\delta}_k^{*2}$ by uniformly sampling t rows from S_k (i.e. with replacement), and then performing the same computations as for $\tilde{\Delta}_k$ and $\tilde{\delta}_k^2$. Then, bootstrapped error variables can be formed as $\varepsilon_k^* = \rho(\tilde{\Delta}_k^*, \tilde{\Delta}_k)$ and $\epsilon_k^* = \varrho(\tilde{\delta}_k^{*2}, \tilde{\delta}_k^2)$. In essence, Algorithm 1 below is simply a computationally efficient implementation of this idea.

Remark. As a clarification, the intermediate objects $\tilde{H}_k^*$, Δ_k^*, and δ_k^{*2} below are not indexed by b, since they need not be saved. The only essential quantities to save at each

Algorithm 1 Error estimation for NEWTON SKETCH

```
Input: The iterate $w_k$, the sketched Newton step $\tilde{\Delta}_k$, the sketch $\tilde{C}_k$, the sketched decrement $\tilde{\delta}_k^2$, as well as the gradient $g_k$ and the Hessian $\nabla^2 r(w_k)$.
for b = 1 to B do in parallel
    • Construct a matrix $\tilde{C}_k^* \in \mathbb{R}^{t\times d}$ whose rows are sampled uniformly from the rows of $\tilde{C}_k$.
    • Compute the following in succession:
        $\tilde{H}_k^* = (\tilde{C}_k^*)^\top(\tilde{C}_k^*) + \nabla^2 r(w_k)$
        $\tilde{\Delta}_k^* = (\tilde{H}_k^*)^{-1} g_k$
        $\tilde{\delta}_k^{*2} = g_k^\top \tilde{\Delta}_k^*$
        $\epsilon_{k,b}^* = \rho(\tilde{\Delta}_k^*, \tilde{\Delta}_k)$
        $\varepsilon_{k,b}^* = \varrho(\tilde{\delta}_k^{*2}, \tilde{\delta}_k^2)$.
Return: $\widehat{q}_{\alpha,k} = \text{quantile}(\epsilon_{k,1}^*, \dots, \epsilon_{k,B}^*; 1-\alpha)$,
        $\widehat{\mathsf{q}}_{\alpha,k} = \text{quantile}(\varepsilon_{k,1}^*, \dots, \varepsilon_{k,B}^*; 1-\alpha)$.
```

iteration are $\epsilon_{k,b}^*$ and $\varepsilon_{k,b}^*$. Similar considerations will also apply to Algorithm 2.

2.3. Bootstrap Error Estimation for GIANT

Recall that in the setting of the GIANT algorithm, there are m workers indexed by $j = 1, \dots, m$, and the j-th worker holds randomly drawn points from $\{(a_i, b_i)\}_{i=1}^n$ that are indexed by a set $\mathcal{I}_j$ with cardinality $t = |\mathcal{I}_j| = n/m$. Each worker computes a local approximate Newton step

$$\widetilde{\Delta}_{k,j} = \widetilde{H}_{k,j}^{-1} g_k,$$

where the local approximate Hessian is given by

$$\tilde{H}_{k,j} = \tilde{C}_{k,j}^\top \tilde{C}_{k,j} + \nabla^2 r(w_k). \tag{14}$$

The globally improved approximate Newton (GIANT) step $\widetilde{\Delta}_k$ is then computed by averaging the local steps $\widetilde{\Delta}_{k,j}$ over $j = 1, \dots, m$, and the current iterate is updated using $\tilde{\Delta}_k$.

By comparing the above expression (14) for $\widetilde{H}_{k,j}$ with the expression (3) for $\widetilde{H}_k$ in the case of NEWTON SKETCH, there is a natural way to extend the previous error estimation algorithm for NEWTON SKETCH to handle GIANT. Namely, for each $j = 1, \dots, m$, the j-th worker uniformly samples t rows from $\tilde{C}_{k,j}$ (with replacement) to form a matrix $\tilde{C}_{k,j}^* \in \mathbb{R}^{t\times d}$, and the resampled matrix $\tilde{C}_{k,j}^*$ is used to compute a resampled version of the local Newton step (and decrement). Then, the local resampled Newton steps (and decrements) will be communicated to a central processor and averaged over $j = 1, \dots, m$ to form a resampled GIANT step (and decrement). The specific details are listed in Algorithm 2.

Algorithm 2 Error estimation for GIANT

```
Input: The iterate $w_k$, the GIANT step $\tilde{\Delta}_k$, the matrices $\tilde{C}_{k,1}, \dots, \tilde{C}_{k,m}$, the approximate decrement $\tilde{\delta}_k^2$, as well as the gradient $g_k$ and the Hessian $\nabla^2 r(w_k)$.
for j = 1 to m do in parallel
    for b = 1 to B do in parallel
        • Construct a matrix $\tilde{C}_{k,j}^* \in \mathbb{R}^{t\times d}$ whose rows are sampled uniformly from the rows of $\tilde{C}_{k,j}$.
        • Compute
            $\tilde{H}_{k,j}^* = (\tilde{C}_{k,j}^*)^\top(\tilde{C}_{k,j}^*) + \nabla^2 r(w_k)$.
        • Compute $\tilde{\Delta}_{k,j,b}^* = (\tilde{H}_{k,j}^*)^{-1} g_k$.
        • Compute $\tilde{\delta}_{k,j,b}^{*2} = g_k^\top \tilde{\Delta}_{k,j,b}^*$.
    • Send the vectors $\tilde{\Delta}_{k,j,1}^*, \dots, \tilde{\Delta}_{k,j,B}^*$ and scalars $\tilde{\delta}_{k,j,1}^{*2}, \dots, \tilde{\delta}_{k,j,B}^{*2}$ to the central processor.
for b = 1 to B do
    • Aggregate local *-steps $\tilde{\Delta}_{k,b}^* = \frac{1}{m}\sum_{j=1}^m \tilde{\Delta}_{k,j,b}^*$.
    • Aggregate local *-decrements $\tilde{\delta}_{k,b}^{*2} = \frac{1}{m}\sum_{j=1}^m \tilde{\delta}_{k,j,b}^{*2}$.
    • Compute *-step error $\epsilon_{k,b}^* = \rho(\tilde{\Delta}_{k,b}^*, \tilde{\Delta}_k)$.
    • Compute *-decrement error $\varepsilon_{k,b}^* = \varrho(\tilde{\delta}_{k,b}^{*2}, \tilde{\delta}_k^2)$.
Return: $\widehat{q}_{\alpha,k} = \text{quantile}(\epsilon_{k,1}^*, \dots, \epsilon_{k,B}^*; 1-\alpha)$,
        $\widehat{\mathsf{q}}_{\alpha,k} = \text{quantile}(\varepsilon_{k,1}^*, \dots, \varepsilon_{k,B}^*; 1-\alpha)$.
```

Remark. Note that each worker $j = 1, \dots, m$ performs its own for-loop over $b = 1, \dots, B$, and each worker may do this in parallel by calling upon several of its own processors (if available). After the loop over $j = 1, \dots, m$ is completed by the m workers, the second loop over $b = 1, \dots, B$ occurs at the central processor. The computations at the central processor consist mostly of inexpensive vector and scalar addition, whereas more substantial matrix computations are done by the workers.

2.4. Computational Cost

We now discuss the computational costs of Algorithms 1 and 2 for error estimation in the respective contexts of NEWTON SKETCH and GIANT. Most importantly, it should be emphasized that these algorithms *do not require any access to the points* $\{(a_i, b_i)\}_{i=1}^n$, which keeps communication costs low.

Cost of Algorithm 1 for NEWTON SKETCH. Given that the loop over $b = 1, \dots, B$ is embarrassingly parallel, it is natural to evaluate the processing cost of Algorithm 1 on a per-iteration basis. For common choices of ρ and ϱ, it is straightforward to check that the processing cost at each iteration is $\mathcal{O}((t+d)d^2)$. In particular, it is notable that this cost is independent of the large dimension n, which allows Algorithm 1 to be highly scalable. Also, to help put this into perspective, it should be noted that the cost of the NEWTON SKETCH algorithm typically scales linearly in n. With regard to the number of bootstrap samples B, it turns out that Algorithm 1 can perform well with surprisingly small choices of B, as will be illustrated by our experiments in Section 4.

Cost of error estimation for GIANT. The processing cost for each worker $j = 1, \dots, m$ in Algorithm 2 is analogous to the processing cost of Algorithm 1 described previously. Also, Algorithm 2 is well-suited to distributed computation, since its loops can be implemented in parallel. On the other hand, a distinct aspect of Algorithm 2 is that it involves an aggregation step after the workers have finished their computations. Nevertheless, this step involves a modest overall communication cost of order $\mathcal{O}(Bmd)$, which is independent of n.

3. Theory

In this section, we analyze Algorithms 1 and 2 in the context of objective functions having the form (1), with r chosen as $r(w) = \frac{\gamma}{2}\|w\|_2^2$ for some $\gamma > 0$. Also, we will focus on the case of relative error,[2] where the error variable is given by $\varepsilon_k = \varrho(\tilde{\delta}_k^2, \delta_k^2) = |\tilde{\delta}_k^2 - \delta_k^2|/\delta_k^2$. Our goal is to show that the estimate $\widehat{\mathsf{q}}_{\alpha,k}$ produced by either algorithm satisfies $\varepsilon_k \leq \widehat{\mathsf{q}}_{\alpha,k}$ with probability not much less than $1-\alpha$.

Theoretical setup. In order to unify our analysis for both NEWTON SKETCH and GIANT, we will work under the following setup, since it allows the matrix $\tilde{H}_k$ defined in (3) for NEWTON SKETCH, and the matrix $\tilde{H}_{k,j}$ defined in (5) for GIANT, to be regarded as equivalent. Namely, this occurs when the sketching matrix S_k in NEWTON SKETCH is generated by uniform sampling from $\{\sqrt{n/t}\, e_1, \dots, \sqrt{n/t}\, e_n\}$, and when the t points held by each worker in GIANT are drawn by uniform sampling from $\{(a_i, b_i)\}_{i=1}^n$. (This type of sampling was also considered by the authors of GIANT (Wang et al., 2018, §A.3).) In addition, we will assume that the workers in GIANT independently sample a fresh set of points at each iteration, and that a fresh sketching matrix S_k in NEWTON SKETCH is generated independently at each iteration (as proposed by the authors of NEWTON SKETCH (Pilanci & Wainwright, 2017)).

Within this setup, it should be noted that in the case of GIANT, the three numbers (n, m, t) satisfy the relation $t = n/m$, whereas in the case of NEWTON SKETCH, these numbers satisfy $m = 1$ and $t \leq n$. Hence, both algorithms may be analyzed simultaneously under the basic condition that $t \leq n/m$. In addition, we will assume that there is an absolute constant $c_1 > 0$, such that $m \leq c_1 t$, which is a mild assumption from a practical standpoint, and will help to simplify the form of our main result.

Notation and definitions. For the regularization function $r(w) = \frac{\gamma}{2}\|w\|_2^2$, observe that the Hessian H_k of f at the k-th iteration has the form

$$H_k = C_k^\top C_k + \gamma I_d. \tag{15}$$

For each $i = 1, \dots, n$, we define the i-th *ridge leverage score* as

$$\ell_{i,k}^\gamma = \left[C_k (C_k^\top C_k + \gamma I_d)^{-1} C_k^\top\right]_{ii} \tag{16}$$

When $\gamma = 0$, this coincides with the standard leverage score. Next, the *effective dimension* d_k^γ is defined as as

$$d_k^\gamma = \ell_{1,k}^\gamma + \cdots + \ell_{n,k}^\gamma = \textstyle\sum_{j=1}^d \frac{\sigma_j^2(C_k)}{\sigma_j^2(C_k)+\gamma}, \tag{17}$$

which can be much smaller than d when C_k has only a few dominant singular values (cf. Li et al., 2020). We also use $\mu_k^\gamma = \mu_k^\gamma(C_k)$ to refer to the ridge coherence, defined as

$$\mu_k^\gamma = \max_{1 \leq i \leq n} \frac{n \ell_{i,k}^\gamma}{d_k^\gamma}. \tag{18}$$

In the case when γ is set to 0, the quantity μ_k^γ reduces to the ordinary matrix coherence (cf. Candès & Recht, 2009), but it should be noted that γ will be taken as positive in our work. Roughly speaking, the ridge coherence μ_k^γ measures how evenly information is spread among the rows of matrix C_k.

Assumption 1. *Let $x_k \in \mathbb{R}^n$ denote the random vector given by $x_k = \frac{1}{\delta_k} C_k H_k^{-1} g_k$, and let $s \in \mathbb{R}^n$ be a random vector sampled uniformly from the set $\{\sqrt{n} e_1, \dots, \sqrt{n} e_n\}$, independently of x_k. Then there is an absolute constant $c_0 > 0$ such that the following bound holds for any $\gamma > 0$ when $x_k \neq 0$,*

$$\operatorname{var}\left((s^\top x_k)^2 \middle| x_k\right) \geq \tfrac{1}{c_0(1+\gamma)^2} \tag{19}$$

Remarks. To provide some intuition for the condition (19), it arises from the fact that our analysis is based on showing that the distribution of the random variable $(\tilde{\delta}_k^2 - \delta_k^2)/\delta_k^2$ is approximately Gaussian (conditionally on w_k). To establish such a Gaussian approximation using non-asymptotic tools like the Berry-Esseen theorem, it is important to ensure that the conditional variance of this random variable does not become too small. However, there are certain collections of points $\{(a_i, b_i)\}_{i=1}^n$ that can cause this variance to be arbitrarily small, or even zero. As a simple example,

[2]Under the stated choice of r, note that if $\delta_k^2 = 0$ holds, then $\tilde{\delta}_k^2 = 0$ must also for both NEWTON SKETCH and GIANT, and so the relative error will be understood as $\varepsilon_k = 0$ in this case.

consider the case when all the observations are the same, $(a_1, b_1) = \cdots = (a_n, b_n)$, and when the function f corresponds to ridge regression — for which it can be checked that C_k is proportional to A. In this case, when NEWTON SKETCH or GIANT use uniform sampling to construct $\tilde{C}_k$ or $\tilde{C}_{k,j}$, it follows that every realization of these matrices must be the same, which causes the variance of $(\tilde{\delta}_k^2 - \delta_k^2)/\delta_k^2$ (conditionally on w_k) to be zero. Similarly, in this case, it can be checked that the quantity $\operatorname{var}((s^\top x_k)^2|x_k)$ in (19) is equal to zero as well. So, roughly speaking, the condition (19) can be interpreted as a way of ruling out "degenerate" cases.

The following theorem is our main theoretical result.

Theorem 1. *Suppose that Assumption 1 holds within the setup described above, and let $\widehat{\mathsf{q}}_{\alpha,k}$ denote the second output of Algorithm 1 or 2. Then, there is an absolute constant $c > 0$ such that the bound*

$$\mathbb{P}\Big(\tfrac{|\tilde{\delta}_k^2 - \delta_k^2|}{\delta_k^2} \leq \widehat{\mathsf{q}}_{\alpha,k}\Big) \;\geq\; 1 - \alpha - c\,\omega$$

holds for some positive number ω satisfying

$$\omega \leq \tfrac{(1+\gamma)^3 \mathbb{E}[(\mu_k^\gamma d_k^\gamma)^3] \log(n)^c}{\sqrt{t}} + \tfrac{\sqrt{\log(B)}}{\sqrt{B}}.$$

Remarks. To comment on same basic features of this result, observe that it is non-asymptotic, and that it accounts for the sampling in the optimization algorithms (measured by t), as well as the sampling in the bootstrap error estimation (measured by B). With regard to the proof, the essential task is to show that the random variable $(\tilde{\delta}_k^2 - \delta_k^2)/\delta_k^2$ has a distribution that is approximately Gaussian. In developing such an approximation, one of the key challenges is to handle the accumulation of bias across the m workers. More specifically, each approximate inverse Hessian matrix $(\tilde{H}_{k,j})^{-1}$ creates bias in the local approximate Newton steps, and in order to show that the bootstrap works, it is necessary to keep track of the cumulative effect of these local biases.

Validating Assumption 1 for generic matrices. The collection of matrices in $\mathbb{R}^{n\times d}$ with orthornormal columns, denoted as $\mathsf{Stief}(n, d)$, is known to possess a natural uniform probability distribution, called the Haar distribution (Meckes, 2019). From a conceptual point of view, a random matrix $Q \in \mathsf{Stief}(n, d)$ generated from this distribution may be regarded as "generic". Accordingly, we can investigate the condition (19), as well as the size of the quantity μ_k^γ, in a generic sense by considering a situation where the Q factor in the QR-decomposition of C_k is drawn from the Haar distribution. In this way, the following proposition provides a kind of validation for Assumption 1, as well a reference point for the size of μ_k^γ.

To proceed, first note that if $C_k = QR$, then the left side of (19) can be explicitly written as a function ϕ of Q, R, and the unit vector $u = \frac{1}{\delta_k} H_k^{-1/2} g_k$. Namely, we have

$$\operatorname{var}((s^\top x_k)^2 | x_k) = \phi(Q, R, u),$$

where

$$\phi(Q,R,u) = \tfrac{1}{n}\textstyle\sum_{i=1}^n \big(u^\top M(R)^\top Q^\top (n e_i e_i^\top - I_n) Q M(R) u\big)^2,$$

and $M(R) = R(R^\top R + \gamma I_d)^{-1/2}$. Similarly, the ridge coherence μ_k^γ can be expressed as a function ψ of Q and R. That is,

$$\mu_k^\gamma = \psi(Q, R) = \max_{1\leq i\leq n} \frac{n\|M(R)^\top Q^\top e_i\|_2^2}{\|M(R)\|_F^2}.$$

With this notation in place, we may now state the following result.

Proposition 1. *Let $u \in \mathbb{R}^d$ be any fixed unit vector, and let $R \in \mathbb{R}^{d\times d}$ be any fixed upper-triangular matrix satisfying*

$$\tfrac{1}{c} \leq \sigma_{\min}(R) \leq \sigma_{\max}(R) \leq c$$

for some absolute constant $c \geq 1$. Also, let $Q \in \mathbb{R}^{n\times d}$ be a random matrix drawn from the uniform distribution on $\mathsf{Stief}(n, d)$. Under these conditions, there exists an absolute constant $c_0 > 0$, such that the inequality

$$\phi(Q, R, u) \geq \tfrac{1}{c_0(1+\gamma)^2} \tag{20}$$

holds with probability at least $1 - \frac{c_0}{n}(d \vee \log n)^2(1+\gamma)^2$, and the inequality

$$\psi(Q, R) \leq c_0\big(\tfrac{\log(n)}{d} \vee 1\big) \tag{21}$$

holds with probability at least $1 - c_0/n$.

Remarks. Since the columns of a random matrix Q drawn uniformly from $\mathsf{Stief}(n, d)$ are *not independent*, it is necessary in the proof of this result to make use of non-asymptotic tools that can allow for such dependence. Specifically, the proof hinges on the fact that if $\Psi : \mathsf{Stief}(n, d) \to \mathbb{R}$ is a Lipschitz function with respect to the Frobenius norm, then the random variable $\Psi(Q)$ has strong concentration properties (cf. Milman & Schechtman, 2009, p.29).

4. Experiments

In this section, we present a collection of experiments that study how well Algorithms 1 and 2 can estimate the errors of NEWTON SKETCH and GIANT in the context of ℓ_2-regularized logistic regression. Accordingly, the objective function has the form

$$f(w) = \tfrac{1}{n}\textstyle\sum_{i=1}^n \log(1 + \exp(-(a_i^\top w) b_i)) + \tfrac{\gamma}{2}\|w\|_2^2,$$

where the observations satisfy $(a_i, b_i) \in \mathbb{R}^d \times \{\pm 1\}$ for all $i \in \{1, \ldots, n\}$.

Experimental setup. We used the SUSY regression dataset of size ($n = 5{,}000{,}000, d = 18$), which can be obtained from LIBSVM (Chang & Lin, 2011). For all the

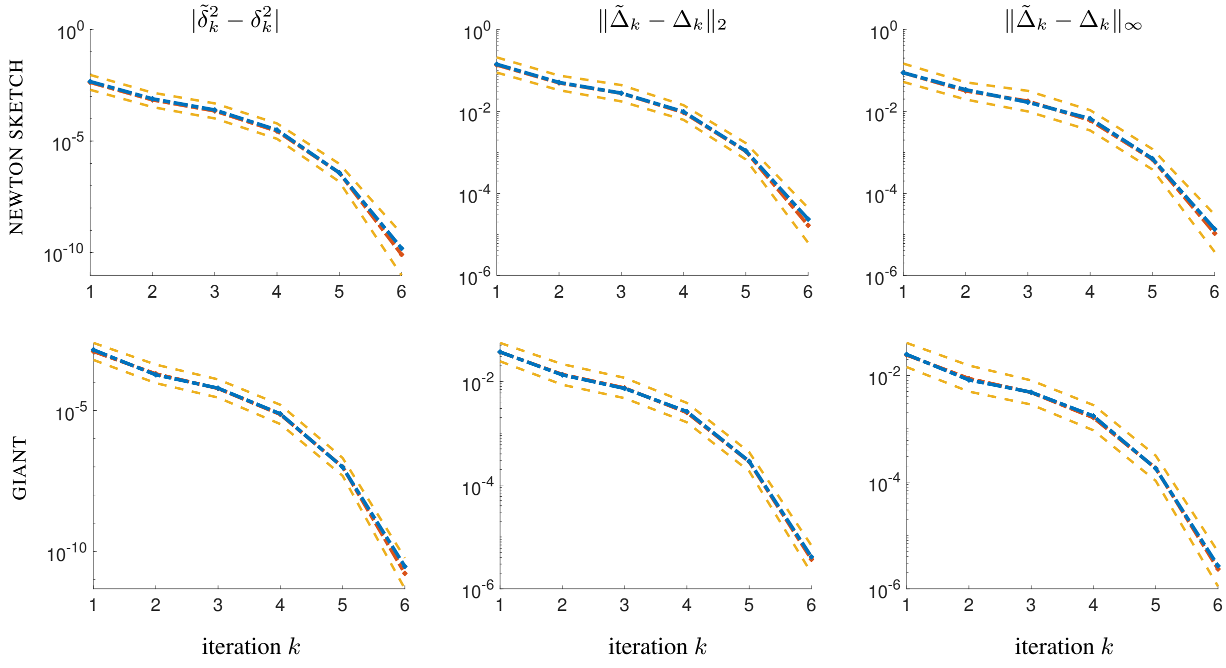

Figure 1. Numerical results on dataset SUSY ($n = 5{,}000{,}000, d = 18$). The plots illustrate the performance of Algorithms 1 and 2 in the task of estimating the quantiles of the three errors $|\tilde{\delta}_k^2 - \delta_k^2|$, $\|\tilde{\Delta}_k - \Delta_k\|_2$, and $\|\tilde{\Delta}_k - \Delta_k\|_\infty$. The blue curves represent the ground truth for the 0.95-quantiles of the errors, as described in the main text. The red curves (which are mostly covered by the blue curves) represent the average of the bootstrap estimates, with the yellow curves being three standard deviations away.

experiments, the regularization parameter was chosen as $\gamma = 10^{-3}$, and the number of bootstrap samples was chosen as $B = 12$. The step size η_k at each iteration of NEWTON SKETCH and GIANT was determined by the Armijo line search so that

$$f(w_k + \eta_k \widetilde{\Delta}_k) \leq f(w_k) + \eta_k \beta \langle \widetilde{\Delta}_k, g_k \rangle.$$

Specifically, the control parameter β was set to $\beta = 0.1$, and the search for the step size was restricted to a grid of values $\eta_k \in \{2^0, 2^{-1}, \ldots, 2^{-10}\}$.

We studied the quantiles of six different kinds of error variables: the absolute error with respect to (1) the Newton decrement, (2) the ℓ_2-norm for the Newton step, and (3) the ℓ_∞-norm for the Newton step, as well as the relative error versions of these three. The results are shown in Figures 1 and 2.

Below, we detail the aspects of the experiments pertaining specifically to Algorithm 1 (for NEWTON SKETCH) and Algorithm 2 (for GIANT).

Experiments for NEWTON SKETCH. The sketched Newton update defined in (4) was independently run 300 times, with 6 iterations $k = 1, \ldots, 6$ in each run. At each iteration, a fresh sketching matrix S_k was generated via uniform sampling, with a sketch size of $t = n/32$. For each realization of S_k, we computed the (true) values of the six error variables mentioned above. This gave 300 total realizations of each type of error variable at each $k = 1, \ldots, 6$. In turn, we used these 300 realizations to compute the empirical 0.95 quantile for each type of error variable, and these quantiles were treated as ground truth for $q_{.05,k}$ and $\mathsf{q}_{.05,k}$. These ground truth values are plotted in blue in Figures 1 and 2. Next, we ran Algorithm 1 on the output associated with each S_k, giving 300 realizations of the bootstrap estimates at each $k = 1, \ldots, 6$ (for each of the six types of 0.95-quantiles). The averages of the bootstrap estimates are plotted in red, with the yellow curves being three standard deviations away. (Note that the red curves are mostly covered by the blue curves in Figure 1.)

Experiments for GIANT. The experiments for GIANT were conducted in a similar way to those for NEWTON SKETCH. We ran the GIANT algorithm 300 times, each time with 6 iterations using the update rule (6). We randomly sampled t data points for each of the m workers before each run, and the data points stayed unchanged on each worker throughout the iterations. We chose the number of machines to be $m = 32$ for all datasets, in correspondence with the sketch size used in NEWTON SKETCH. The true error variables were computed at each iteration (giving 300 realizations of each), and the empirical 0.95-quantiles of these realizations were treated as ground truth for $q_{.05,k}$ and $\mathsf{q}_{.05,k}$. We also ran Algorithm 2 for each run of GIANT, yielding 300 bootstrap estimates for each type of error variable at each iteration. The results are plotted in Figures 1 and 2 using the same

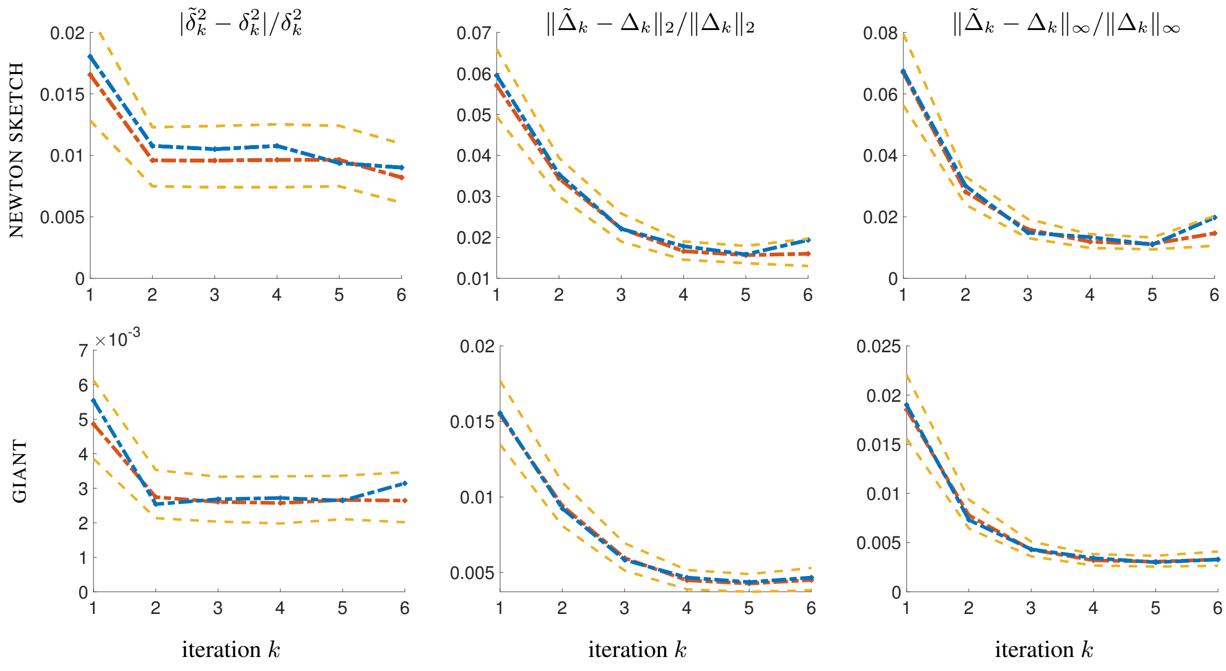

Figure 2. Numerical results on dataset SUSY ($n = 5{,}000{,}000, d = 18$). The plots illustrate the performance Algorithms 1 and 2 in the task of estimating the quantiles of three relative errors $|\tilde{\delta}_k^2 - \delta_k^2|/\delta_k^2$, $\|\tilde{\Delta}_k - \Delta_k\|_2/\|\Delta_k\|_2$, and $\|\tilde{\Delta}_k - \Delta_k\|_\infty/\|\Delta_k\|_\infty$. The labelling scheme for the curves is the same as that used in Figure 1.

scheme as that described above in the context of NEWTON SKETCH.

Remarks. Our experiments show that Algorithms 1 and 2 perform reliably for each of the six types of error variables considered. In particular, the red lines are well aligned with the blue lines in all of the plots, indicating that the bootstrap estimates are nearly unbiased. The small gap between the yellow curves also shows that the bootstrap estimates have fairly low variance — which is encouraging in light of the fact that the estimates were computed using only $B = 12$ bootstrap samples. Furthermore, this performance with a small choice of B also demonstrates that error estimation need not add much cost to the underlying optimization algorithm.

Acknowledgements

MEL gratefully acknowledges partial support from NSF grants DMS-1613218 and DMS-1915786. XTC gratefully acknowledges partial support from NSF HDR TRIPODS grant CCF-1934568.

References

Anastasiou, A., Balasubramanian, K., and Erdogdu, M. A. Normal approximation for stochastic gradient descent via non-asymptotic rates of martingale CLT. In *Proceedings of the Thirty-Second Conference on Learning Theory*, pp. 115–137, 2019.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2004.

Byrd, R. H., Hansen, S. L., Nocedal, J., and Singer, Y. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

Candès, E. J. and Recht, B. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717, 2009.

Chang, C.-C. and Lin, C.-J. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011. URL http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Chapelle, O. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

Chernick, M. R. *Bootstrap Methods: A Guide for Practitioners and Researchers*. John Wiley & Sons, 2011.

Davison, A. C. and Hinkley, D. V. *Bootstrap Methods and their Application*. Cambridge University Press, 1997.

Duersch, J. A. and Gu, M. Randomized QR with column pivoting. *SIAM Journal on Scientific Computing*, 39(4): C263–C291, 2017.

Dünner, C., Lucchi, A., Gargiani, M., Bian, A., Hofmann, T., and Jaggi, M. A distributed second-order algorithm you can trust. *arXiv:1806.07569*, 2018.

Efron, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

Efron, B. *The Jackknife, the Bootstrap and Other Resampling Plans*. SIAM, 1982.

Erdogdu, M. A. and Montanari, A. Convergence rates of sub-sampled Newton methods. *arXiv:1508.02810*, 2015.

Fang, Y. Scalable statistical inference for averaged implicit stochastic gradient descent. *Scandinavian Journal of Statistics*, 46(4):987–1002, 2019.

Fang, Y., Xu, J., and Yang, L. Online bootstrap confidence intervals for the stochastic gradient descent estimator. *The Journal of Machine Learning Research*, 19(1):3053–3073, 2018.

Gupta, V., Kadhe, S., Courtade, T., Mahoney, M. W., and Ramchandran, K. Oversketched Newton: Fast convex optimization for serverless systems. *arXiv:1903.08857*, 2019.

Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Li, T., Liu, L., Kyrillidis, A., and Caramanis, C. Statistical inference using SGD. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Li, X., Wang, S., and Zhang, Z. Do subsampled Newton methods work for high-dimensional data? In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

Liberty, E., Woolfe, F., Martinsson, P.-G., Rokhlin, V., and Tygert, M. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172, 2007.

Lopes, M., Wang, S., and Mahoney, M. W. Error estimation for randomized least-squares algorithms via the bootstrap. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 3217–3226, 2018.

Lopes, M. E., Erichson, N. B., and Mahoney, M. W. Bootstrapping the operator norm in high dimensions: Error estimation for covariance matrices and sketching. *arXiv:1909.06120*, 2019a.

Lopes, M. E., Wang, S., and Mahoney, M. W. A bootstrap method for error estimation in randomized matrix multiplication. *Journal of Machine Learning Research*, 20 (39):1–40, 2019b.

Lopes, M. E., Erichson, N. B., and Mahoney, M. W. Error estimation for sketched SVD via the bootstrap. In *Proceedings of the 37th International Conference on Machine Learning (to appear)*, 2020.

Martinsson, P.-G. and Tropp, J. Randomized numerical linear algebra: Foundations & algorithms. *Acta Numerica (to appear), arXiv:2002.01387*, 2020.

Martinsson, P.-G. and Voronin, S. A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices. *SIAM Journal on Scientific Computing*, 38(5):S485–S507, 2016.

McCullagh, P. and Nelder, J. A. *Generalized Linear Models*. CRC press, 1989.

Meckes, E. S. *The Random Matrix Theory of the Classical Compact Groups*. Cambridge University Press, 2019.

Milman, V. D. and Schechtman, G. *Asymptotic Theory of Finite Dimensional Normed Spaces: Isoperimetric Inequalities in Riemannian Manifolds*, volume 1200. Springer, 2009.

Nesterov, Y. and Nemirovskii, A. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 1994.

Pilanci, M. and Wainwright, M. J. Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares. *The Journal of Machine Learning Research*, 17(1):1842–1879, 2016.

Pilanci, M. and Wainwright, M. J. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

Reddi, S. J., Konečnỳ, J., Richtárik, P., Póczós, B., and Smola, A. AIDE: Fast and communication efficient distributed optimization. *arXiv:1608.06879*, 2016.

Roosta-Khorasani, F. and Mahoney, M. W. Sub-sampled Newton methods. *Mathematical Programming*, 174(1-2): 293–326, 2019.

Shamir, O., Srebro, N., and Zhang, T. Communication-efficient distributed optimization using an approximate Newton-type method. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 1000–1008, 2014.

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

Sorensen, D. C. and Embree, M. A DEIM induced CUR factorization. *SIAM Journal on Scientific Computing*, 38 (3):A1454–A1482, 2016.

Su, W. and Zhu, Y. Uncertainty quantification for online learning and stochastic approximation via hierarchical incremental gradient descent. *arXiv:1802.04876*, 2018.

Tropp, J. A., Yurtsever, A., Udell, M., and Cevher, V. Streaming low-rank matrix approximation with an application to scientific simulation. *SIAM Journal on Scientific Computing*, 41(4):A2430–A2463, 2019.

Wang, J., Lee, J. D., Mahdavi, M., Kolar, M., Srebro, N., et al. Sketching meets random projection in the dual: A provable recovery algorithm for big and high-dimensional data. *Electronic Journal of Statistics*, 11(2):4896–4944, 2017.

Wang, S., Roosta-Khorasani, F., Xu, P., and Mahoney, M. W. Giant: Globally improved approximate Newton method for distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 2332–2342, 2018.

Woolfe, F., Liberty, E., Rokhlin, V., and Tygert, M. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3): 335–366, 2008.

Xu, P., Yang, J., Roosta-Khorasani, F., Ré, C., and Mahoney, M. W. Sub-sampled Newton methods with non-uniform sampling. In *Advances in Neural Information Processing Systems*, pp. 3000–3008, 2016.

Yu, W., Gu, Y., and Li, Y. Efficient randomized algorithms for the fixed-precision low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 39 (3):1339–1359, 2018.

Zhang, Y. and Lin, X. DiSCO: Distributed optimization for self-concordant empirical loss. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 362–370, 2015.