Error Estimation for Randomized Least-Squares Algorithms via the Bootstrap

Miles E. Lopes [1] **Shusen Wang** [2] **Michael W. Mahoney** [2]

Abstract

Over the course of the past decade, a variety of randomized algorithms have been proposed for computing approximate least-squares (LS) solutions in large-scale settings. A longstanding practical issue is that, for any given input, the user rarely knows the *actual error* of an approximate solution (relative to the exact solution). Likewise, it is difficult for the user to know precisely how much computation is needed to achieve the desired error tolerance. Consequently, the user often appeals to worst-case error bounds that tend to offer only qualitative guidance. As a more practical alternative, we propose a bootstrap method to compute *a posteriori error estimates* for randomized LS algorithms. These estimates permit the user to numerically assess the error of a given solution, and to predict how much work is needed to improve a "preliminary" solution. In addition, we provide theoretical consistency results for the method, which are the first such results in this context (to the best of our knowledge). From a practical standpoint, the method also has considerable flexibility, insofar as it can be applied to several popular sketching algorithms, as well as a variety of error metrics. Moreover, the extra step of error estimation does not add much cost to an underlying sketching algorithm. Finally, we demonstrate the effectiveness of the method with empirical results.

1. Introduction

Randomized sketching algorithms have been intensively studied in recent years as a general approach to computing fast approximate solutions to large-scale least-squares (LS) problems (Drineas et al., 2006; Rokhlin & Tygert, 2008; Avron et al., 2010; Drineas et al., 2011; Mahoney, 2011; Drineas et al., 2012; Clarkson & Woodruff, 2013; Woodruff, 2014; Ma et al., 2014; Meng et al., 2014; Pilanci & Wainwright, 2015; 2016). During this time, much progress has been made in analyzing the performance of these algorithms, and existing theory provides a good qualitative description of approximation error (relative to the exact solution) in terms of various problem parameters. However, in practice, the user rarely knows the *actual error* of a randomized solution, or how much extra computation may be needed to achieve a desired level of accuracy.

A basic source of this problem is that it is difficult to translate theoretical error bounds into numerical error bounds that are tight enough to be quantitatively meaningful. For instance, theoretical bounds are often formulated to hold for the worst-case input among a large class of possible inputs. Consequently, they are often pessimistic for "generic" problems, and they may not account for the structure that is unique to the input at hand. Another practical issue is that these bounds typically involve constants that are either conservative, unspecified, or dependent on unknown parameters.

In contrast with worst-case error bounds, we are interested in "a posteriori" error estimates. By this, we mean error bounds that can be estimated numerically in terms of the computed solution or other observable information. Although methods for obtaining a posteriori error estimates are well-developed in some areas of computer science and applied mathematics, there has been very little development for randomized sketching algorithms (cf. Section 1.4). (For brevity, we will usually omit the qualifier 'a posteriori' from now on when referring to error estimation.)

The main purpose of this paper is to show that it is possible to directly estimate the error of randomized LS solutions in a way that is both practical and theoretically-justified. Accordingly, we propose a flexible estimation method that can enhance existing sketching algorithms in a variety of ways. In particular, we will explain how error estimation can help the user to (1) select the "sketch size" parameter, (2) assess the convergence of iterative sketching algorithms, and (3) measure error in a wider range of metrics than can be handled by existing theory.

[1]Department of Statistics, UC Davis [2]ICSI and Department of Statistics, UC Berkeley. Correspondence to: <melopes@ucdavis.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

1.1. Setup and Background

Consider a large, overdetermined LS problem, involving a rank d matrix $A \in \mathbb{R}^{n\times d}$, and a vector $b \in \mathbb{R}^n$, where $n \gg d$. These inputs are viewed as being deterministic, and the exact solution is denoted

$$x_{\text{opt}} := \operatorname*{argmin}_{x\in\mathbb{R}^d} \|Ax - b\|_2. \tag{1}$$

The large number of rows n is often a major computational bottleneck, and sketching algorithms overcome this obstacle by effectively solving a smaller problem involving m rows, where $d \ll m \ll n$. In general, this reduction is carried out with a random sketching matrix $S \in \mathbb{R}^{m\times n}$ that maps the full matrix A into a smaller sketched matrix $\tilde{A} := SA$ of size $m \times d$. However, various sketching algorithms differ in the way that the matrix S is generated, or the way that $\tilde{A}$ is used. Below, we quickly summarize three of the most well-known types of sketching algorithms for LS.

Classic Sketch (CS). For a given sketching matrix S, this type of algorithm produces a solution

$$\tilde{x} := \operatorname*{argmin}_{x\in\mathbb{R}^d} \|S(Ax - b)\|_2, \tag{2}$$

and chronologically, this was the first type of sketching algorithm for LS (Drineas et al., 2006).

Hessian Sketch (HS). The HS algorithm modifies the objective function in the problem (1) so that its Hessian is easier to compute (Pilanci & Wainwright, 2016; Becker et al., 2017), leading to a solution

$$\breve{x} := \operatorname*{argmin}_{x\in\mathbb{R}^d} \Big\{ \tfrac{1}{2}\|SAx\|_2^2 - \langle A^\top b, x\rangle \Big\}. \tag{3}$$

This algorithm is also called "partial sketching".

Iterative Hessian Sketch (IHS). One way to extend HS is to refine the solution iteratively. For a given iterate $\hat{x}_i \in \mathbb{R}^d$, the following update rule is used

$$\hat{x}_{i+1} := \operatorname*{argmin}_{x\in\mathbb{R}^d} \Big\{ \tfrac{1}{2}\|S_{i+1}A(x - \hat{x}_i)\|_2^2 + \langle A^\top(A\hat{x}_i - b)\,,\, x\rangle \Big\},$$

where $S_{i+1} \in \mathbb{R}^{m\times n}$ is a random sketching matrix that is generated independently of $S_1, \ldots, S_i$, as proposed in (Pilanci & Wainwright, 2016). If we let $t \geq 1$ denote the total number of IHS iterations, then we will generally write $\hat{x}_t$ to refer to the final output of IHS.

Remark. If the initial point for IHS is chosen as $\hat{x}_0 = 0$, then the first iterate $\hat{x}_1$ is equivalent to the HS solution $\breve{x}$ in equation (3). Consequently, HS may be viewed as a special case of IHS, and so we will restrict our discussion to CS and IHS for simplicity.

With regard to the choice of the sketching matrix, many options have been considered in the literature, and we refer to the surveys (Mahoney, 2011) and (Woodruff, 2014). Typically, the matrix S is generated so that the relation $\mathbb{E}[S^\top S] = I_n$ holds, and that the rows of S are i.i.d. random vectors (or nearly i.i.d.). Conceptually, our proposed method only relies on these basic properties of S, and in practice, it can be implemented with any sketching matrix.

To briefly review the computational benefits of sketching algorithms, first recall that the cost of solving the full least-squares problem (1) by standard methods is $\mathcal{O}(nd^2)$ (Golub & Van Loan, 2012). On the other hand, if the cost of computing the matrix product SA is denoted C_{sketch}, and if a standard method is used to solve the sketched problem (2), then the total cost of CS is $\mathcal{O}(md^2 + C_{\text{sketch}})$. Similarly, the total cost of IHS with t iterations is $\mathcal{O}(t(md^2 + C_{\text{sketch}}))$. Regarding the sketching cost C_{sketch}, it depends substantially on the choice of S, but there are many types that improve upon the naive $\mathcal{O}(mnd)$ cost of unstructured matrix multiplication. For instance, if S is chosen to be a Sub-sampled Randomized Hadamard Transform (SRHT), then $C_{\text{sketch}} = \mathcal{O}(nd\log(m))$ (Ailon & Chazelle, 2006; Sarlós, 2006; Ailon & Liberty, 2009). Based on these considerations, sketching algorithms can be more efficient than traditional LS algorithms when $md^2 + nd\log(m) \ll nd^2$.

1.2. Problem Formulation

For any problem instance, we will estimate the errors of the random vectors $\tilde{x}$ and $\hat{x}_t$ in terms of high-probability bounds. Specifically, if we let $\|\cdot\|_\circ$ denote *any* norm on $\mathbb{R}^d$, and let $\alpha \in (0,1)$ be fixed, then our goal is to construct numerical estimates $\tilde{\varepsilon}(\alpha)$ and $\hat{\varepsilon}_t(\alpha)$, such that the bounds

$$\|\tilde{x} - x_{\text{opt}}\|_\circ \leq \tilde{\varepsilon}(\alpha) \tag{4}$$

$$\|\hat{x}_t - x_{\text{opt}}\|_\circ \leq \hat{\varepsilon}_t(\alpha) \tag{5}$$

each hold with probability at least $1 - \alpha$. (This probability will account for the randomness in both the sketching algorithm, and the bootstrap sampling described below.) Also, the algorithm for computing $\tilde{\varepsilon}(\alpha)$ or $\hat{\varepsilon}_t(\alpha)$ should be efficient enough so that the total cost of computing $(\tilde{x}, \tilde{\varepsilon}(\alpha))$ or $(\hat{x}_t, \hat{\varepsilon}_t(\alpha))$ is still much less than the cost of computing the exact solution x_{opt} — otherwise, the extra step of error estimation would defeat the purpose of sketching. (This cost will be addressed in Section 2.3.) Since x_{opt} is unknown to the user, it might seem surprising that it is possible to construct error estimates that satisfy the conditions above, and indeed, the limited knowledge of x_{opt} is the main source of difficulty in the problem.

1.3. Main Contributions

At a high level, a distinguishing feature of our approach is that it applies inferential ideas from statistics in order to enhance large-scale computations. To be more specific, the novelty of this approach is that it differs from the traditional framework of using bootstrap methods to quantify uncertainty arising from data (Davison & Hinkley, 1997). Instead, we are using these methods to quantify uncertainty in the outputs of randomized algorithms — and there do not seem to be many works that have looked at the bootstrap from this angle. From a more theoretical standpoint, another main contribution is that we offer the first guarantees for a posteriori error estimation in the setting of randomized LS algorithms (to the best of our knowledge). Looking beyond this setting, there may be further opportunities for using bootstrap methods to estimate the errors of other randomized algorithms. In concurrent work, we have taken this approach in the distinct settings of randomized matrix multiplication, and randomized ensemble classifiers (Lopes et al., 2017; Lopes, 2018).

1.4. Related work

The general problem of error estimation for approximation algorithms has been considered in a wide range of situations, and we refer to the following works for surveys and examples: (Pang, 1987; Verfürth, 1994; Jiránek et al., 2010; Ainsworth & Oden, 2011; Colombo & Vlassis, 2016). In the context of sketching algorithms, there is only a handful of papers that address error estimation, and these are geared toward low-rank approximation (Liberty et al., 2007; Woolfe et al., 2008; Halko et al., 2011), or matrix multiplication (Sarlós, 2006; Lopes et al., 2017). In addition to the works just mentioned, the recent preprint (Ahfock et al., 2017) explores statistical properties of the CS and HS algorithms, and it develops analytical formulas for describing how $\tilde{x}$ and $\breve{x}$ fluctuate around x_{opt}. Although these formulas offer insight into error estimation, their application is limited by the fact that they involve unknown parameters. Also, the approach in (Ahfock et al., 2017) does not address IHS. Lastly, error estimation for LS approximations can be studied from a Bayesian perspective, and this has been pursued in the paper (Bartels & Hennig, 2016), but with a focus on algorithms that differ from the ones studied here.

Notation. The following notation is needed for our proposed algorithms. Let $\tilde{b} := Sb \in \mathbb{R}^m$ denote the sketched version of b. If $\mathbf{i} = (i_1, \dots, i_m)$ is a vector containing m numbers from $\{1, \dots, m\}$, then $\tilde{A}(\mathbf{i}, :)$ refers to the $m \times d$ matrix whose jth row is equal to the i_jth row of $\tilde{A}$. Similarly, the jth component of the vector $\tilde{b}(\mathbf{i})$ is the i_jth component of $\tilde{b}$. Next, for any fixed $\alpha \in (0, 1)$, and any finite set of real numbers $C = \{c_1, \dots, c_k\}$, the expression $\text{quantile}(c_1, \dots, c_k; 1 - \alpha)$ is defined as the smallest element $c_{i_0} \in C$ for which the sum $\frac{1}{k}\sum_{i=1}^{k} 1\{c_i \le c_{i_0}\}$ is at least $1 - \alpha$. Lastly, the distribution of a random variable U is denoted $\mathcal{L}(U)$, and the conditional distribution of U given a random variable V is denoted $\mathcal{L}(U|V)$.

2. Method

The proposed bootstrap method is outlined in Sections 2.1 and 2.2 for the cases of CS and IHS respectively. After the method is presented in algorithmic form (for each case), we give heuristic interpretations to explain why it works. The formal analysis can be found in the proof of Theorem 1 in the supplement (Lopes et al., 2018). Later on, in Section 2.3, we discuss computational cost and speedups.

2.1. Error Estimation for CS

The main challenge we face is that the distribution of the random variable $\|\tilde{x} - x_{\text{opt}}\|_\circ$ is unknown. If we had access to this distribution, we could find the tightest possible upper bound on $\|\tilde{x} - x_{\text{opt}}\|_\circ$ that holds with probability at least $1 - \alpha$. (This bound is commonly referred to as the $(1 - \alpha)$-quantile of the random variable $\|\tilde{x} - x_{\text{opt}}\|_\circ$.)

From an intuitive standpoint, the idea of the proposed bootstrap method is to artificially generate many samples of a random vector, say $\tilde{x}^*$, whose fluctuations around $\tilde{x}$ are statistically similar to the fluctuations of $\tilde{x}$ around x_{opt}. In turn, we can use the empirical $(1-\alpha)$-quantile of the values $\|\tilde{x}^* - \tilde{x}\|_\circ$ to obtain the desired quantity $\tilde{\varepsilon}(\alpha)$ in (4).

Remark. As a technical clarification, it is important to note that our method relies only on a *single run* of CS, involving just one sketching matrix S. Consequently, the bootstrapped vectors $\tilde{x}^*$ will be generated conditionally on the given S. In this way, the bootstrap aims to generate random vectors $\tilde{x}^*$, such that for a given draw of S, the conditional distribution $\mathcal{L}(\tilde{x}^* - \tilde{x} \mid S)$ is approximately equal to the unknown distribution $\mathcal{L}(\tilde{x} - x_{\text{opt}})$.

Algorithm 1. (Error estimate for CS)

```
Input: A positive integer $B$, and the sketches $\tilde{A}$, $\tilde{b}$, and $\tilde{x}$.
For: $l = 1, \dots, B$ do
    • Draw a random vector $\mathbf{i} := (i_1, \dots, i_m)$ by sampling $m$ numbers with replacement from $\{1, \dots, m\}$.
    • Form the matrix $\tilde{A}^* := \tilde{A}(\mathbf{i}, :)$, and vector $\tilde{b}^* := \tilde{b}(\mathbf{i})$.
    • Compute the vector
          $\tilde{x}^* := \underset{x \in \mathbb{R}^d}{\text{argmin}} \|\tilde{A}^* x - \tilde{b}^*\|_2,$   (6)
      and the scalar $\varepsilon_l^* := \|\tilde{x}^* - \tilde{x}\|_\circ$.
Return: $\tilde{\varepsilon}(\alpha) := \text{quantile}(\varepsilon_1^*, \dots, \varepsilon_B^*; 1 - \alpha)$.
```

Heuristic interpretation of Algorithm 1. To explain why the bootstrap works, let $\mathbb{S}_A$ denote the set of positive semidefinite matrices $M \in \mathbb{R}^{n \times n}$ such that $A^\top M A$ is invertible, and define the map $\psi : \mathbb{S}_A \to \mathbb{R}^d$ according to

$$\psi(M) = (A^\top M A)^{-1} A^\top M b. \tag{7}$$

This map leads to the relation[*]

$$\tilde{x} - x_{\text{opt}} = \psi(S^\top S) - \psi(I_n), \tag{8}$$

where I_n denotes the $n \times n$ identity matrix. By analogy, if we let $S^* \in \mathbb{R}^{m\times n}$ denote a matrix obtained by sampling m rows from S with replacement, then $\tilde{x}^*$ can be written as

$$\tilde{x}^* = \underset{x\in\mathbb{R}^d}{\operatorname{argmin}} \|S^*(Ax - b)\|_2, \tag{9}$$

and the definition of ψ gives

$$\tilde{x}^* - \tilde{x} \;=\; \psi(S^{*\top} S^*) - \psi(S^\top S). \tag{10}$$

Using the corresponding relations (8) and (10), it becomes easier to show why the distributions $\mathcal{L}(\tilde{x} - x_{\text{opt}})$ and $\mathcal{L}(\tilde{x}^* - \tilde{x}|S)$ should be nearly equal.

To proceed, if we let $s_1, \dots, s_m \in \mathbb{R}^n$ denote the rows of $\sqrt{m}S$, it is helpful to note the basic algebraic fact

$$S^\top S - I_n = \tfrac{1}{m} \textstyle\sum_{i=1}^m (s_i s_i^\top - I_n). \tag{11}$$

Given that sketching matrices are commonly constructed so that $s_1, \dots, s_m$ are i.i.d. (or nearly i.i.d.) with $\mathbb{E}[s_1 s_1^\top] = I_n$, the matrix $S^\top S$ becomes an increasingly good approximation to I_n as m becomes large. Hence, it is natural to consider a first-order expansion of the right side of (8),

$$\tilde{x} - x_{\text{opt}} \;\approx\; \psi'_{I_n}(S^\top S - I_n), \tag{12}$$

where ψ'_{I_n} is the differential of the map ψ at I_n. Likewise, if we define a set of vectors $v_1, \dots, v_m \in \mathbb{R}^d$ as $v_i := \psi'_{I_n}(s_i s_i^\top - I_n)$, then the linearity of ψ'_{I_n} gives

$$\tilde{x} - x_{\text{opt}} \;\approx\; \tfrac{1}{m} \textstyle\sum_{i=1}^m v_i, \tag{13}$$

and furthermore, the vectors $v_1, \dots, v_m$ are i.i.d. whenever the vectors $s_1, \dots, s_m$ are. Consequently, as the sketch size m becomes large, the central limit theorem suggests that the difference $\sqrt{m}(\tilde{x} - x_{\text{opt}})$ should be approximately Gaussian,

$$\mathcal{L}\big(\sqrt{m}(\tilde{x} - x_{\text{opt}})\big) \;\approx\; \mathcal{N}\big(0, \Sigma\big), \tag{14}$$

where we put $\Sigma := \mathbb{E}[v_1 v_1^\top]$.

To make the connection with $\tilde{x}^* - \tilde{x}$, each of the preceding steps can be carried out in a corresponding manner. Specifically, if the differential of ψ at $S^\top S$ is sufficiently close to the differential at I_n, then an expansion of equation (10) leads to the bootstrap analogue of (13),

$$\tilde{x}^* - \tilde{x} \;\approx\; \tfrac{1}{m} \textstyle\sum_{i=1}^m v_i^*, \tag{15}$$

[*]For standard types of sketching matrices, the event $S^\top S \in \mathbb{S}_A$ occurs with high probability when $A^\top A$ is invertible and m is sufficiently larger than d (and similarly for $S^{*\top} S^*$).

where $v_i^* := \psi'_{I_n}(s_i^* s_i^{*\top} - S^\top S)$, and the vector s_i^* is the ith row of $\sqrt{m}S^*$. Since the row vectors $s_1^*, \dots, s_m^*$ are obtained by sampling with replacement from $\sqrt{m}S$, it follows that the vectors $v_1^*, \dots, v_m^*$ are conditionally i.i.d. given S, and also, $\mathbb{E}[v_i^*|S] = 0$. Therefore, if we condition on S, the central limit theorem suggests that as m becomes large

$$\mathcal{L}\big(\sqrt{m}(\tilde{x}^* - \tilde{x}) \,|\, S\big) \;\approx\; \mathcal{N}\big(0, \hat{\Sigma}\big), \tag{16}$$

where the conditional covariance matrix is denoted by $\hat{\Sigma} := \mathbb{E}\big[v_i^* v_i^{*\top} \big| S\big]$. Comparing the Gaussian approximations (14) and (16), this heuristic argument indicates that the distributions $\mathcal{L}(\tilde{x} - x_{\text{opt}})$ and $\mathcal{L}(\tilde{x}^* - \tilde{x}|S)$ should be close as long as $\hat{\Sigma}$ is close to Σ, and when m is large, this is enforced by the law of large numbers.

2.2. Error Estimation for IHS

At first sight, it might seem that applying the bootstrap to IHS would be substantially different than in the case of CS — given that IHS is an iterative algorithm, whereas CS is a "one-shot" algorithm. However, the bootstrap only needs to be modified slightly. Furthermore, the bootstrap relies on just the final two iterations of *a single run* of IHS.

To fix some notation, recall that t denotes the total number of IHS iterations, and let $S_t \in \mathbb{R}^{m\times n}$ denote the sketching matrix used in the last iteration. Also define the matrix $\tilde{A}_t := S_t A$, and the gradient vector $g_{t-1} := A^\top(A\hat{x}_{t-1} - b)$ that is computed during the second-to-last iteration of IHS. Lastly, we note that the user is free to select any initial point $\hat{x}_0$ for IHS, and this choice does not affect our method.

Algorithm 2. (Error estimate for IHS)

```
Input: A positive integer $B$, the sketch $\tilde{A}_t$, the gradient $g_{t-1}$, the second-to-last iterate $\hat{x}_{t-1}$, and the last iterate $\hat{x}_t$.
For $l = 1, \dots, B$ do
    • Draw a random vector $\mathbf{i} := (i_1, \dots, i_m)$ by sampling $m$ numbers with replacement from $\{1, \dots, m\}$.
    • Form the matrix $\tilde{A}_t^* := \tilde{A}_t(\mathbf{i}, :)$.
    • Compute the vector
          $\hat{x}_t^* := \underset{x\in\mathbb{R}^d}{\operatorname{argmin}} \Big\{ \tfrac{1}{2}\|\tilde{A}_t^*(x - \hat{x}_{t-1})\|_2^2 + \big\langle g_{t-1}, x \big\rangle \Big\}$   (17)
      and the scalar $\varepsilon_{t,l}^* := \|\hat{x}_t^* - \hat{x}_t\|_\circ$.
Return: $\hat{\varepsilon}_t(\alpha) := \text{quantile}(\varepsilon_{t,1}^*, \dots, \varepsilon_{t,B}^*; 1 - \alpha)$
```

Heuristic interpretation of Algorithm 2. The ideas underlying the IHS version of the bootstrap are broadly similar to those discussed for the CS version. In analogy with Algorithm 1, the main point is to argue that the fluctuations of $\hat{x}_t^*$ around $\hat{x}_t$ can be used as a proxy for the fluctuations of $\hat{x}_t$ around x_{opt}.

For a given pair of vectors $\hat{x}_{t-1}$ and g_{t-1}, define the map $\varphi_t : \mathbb{S}_A \to \mathbb{R}^d$ according to

$$\varphi_t(M) = \hat{x}_{t-1} - (A^\top MA)^{-1} g_{t-1}, \tag{18}$$

where we recall that $\mathbb{S}_A$ denotes the set of positive semidefinite matrices $M \in \mathbb{R}^{n\times n}$ such that $A^\top MA$ is invertible. It is straightforward to verify that this definition allows us to represent the difference $\hat{x}_t - x_{\text{opt}}$ and its bootstrap counterpart $\hat{x}_t^* - \hat{x}_t$ in terms of the matching relations

$$\hat{x}_t - x_{\text{opt}} \;=\; \varphi_t(S_t^\top S_t) - \varphi_t(I_n), \tag{19}$$

$$\hat{x}_t^* - \hat{x}_t \;\;=\; \varphi_t(S_t^{*\top} S_t^*) - \varphi_t(S_t^\top S_t), \tag{20}$$

where S_t^* has m rows sampled with replacement from S_t. As in the discussion of Algorithm 1, the right sides of the relations (19) and (20) can be expanded to first order, which allows for approximations based on the central limit theorem to be used. Indeed, it can be argued that as m becomes large, the random vectors $\sqrt{m}(\hat{x}_t - x_{\text{opt}})$ and $\sqrt{m}(\hat{x}_t^* - \hat{x}_t)$ approach a common Gaussian distribution in a conditional sense. However, the details of this argument are much more complex than in the CS case — because the map φ_t is random and varies with t, whereas the map ψ in the CS case is fixed. A formal analysis may be found in the proof of Theorem 1 in the supplement (Lopes et al., 2018).

2.3. Computational Cost and Speedups

Of course, the quality control that is provided by error estimation does not come for free. Nevertheless, there are several special properties of Algorithms 1 and 2 that keep their computational cost in check, and in particular, the cost of computing $(\tilde{x}, \tilde{\varepsilon}(\alpha))$ or $(\hat{x}_t, \hat{\varepsilon}_t(\alpha))$ is much less than the cost of solving the full LS problem (1). These properties are summarized below.

1. **Cost of error estimation is independent of n.**
   An important observation to make about Algorithms 1 and 2 is that their inputs consist of pre-computed matrices of size $m \times d$ or pre-computed vectors of dimension d. Consequently, both algorithms are highly scalable in the sense that their costs do not depend on the large dimension n. As a point of comparison, it should also be noted that sketching algorithms for LS generally have costs that scale linearly with n.

2. **Implementation is embarrassingly parallel.**
   Due to the fact that each bootstrap sample is computed independently of the others, the for-loops in Algorithms 1 and 2 can be easily distributed. *Furthermore, it turns out that in practice, as few as $B = 20$ bootstrap samples are often sufficient to obtain good error estimates, as illustrated in Section 4.* Consequently, even if the error estimation is done on a single workstation, it is realistic to suppose that the user has access to N processors such that the number of bootstrap samples per processor is $B/N = \mathcal{O}(1)$. If this is the case, and if $\|\cdot\|_\circ$ is any ℓ_p norm on $\mathbb{R}^d$, then it follows that the per-processor cost of both algorithms is only $\mathcal{O}(md^2)$. Lastly, the communication costs in this situation are also very modest. In fact, it is only necessary to send a single $m \times d$ matrix, and at most three d-vectors to each processor. In turn, when the results are aggregated, only B scalars are sent back to the central processor.

3. **Bootstrap computations have free warm starts.**
   The bootstrap samples $\tilde{x}^*$ and $\hat{x}_t^*$ can be viewed as perturbations of the actual sketched solutions $\tilde{x}$ and $\hat{x}_t$. This is because the associated optimization problems only differ with respect to resampled versions of $\tilde{A}$ and $\tilde{b}$. Therefore, if a sub-routine for computing $\tilde{x}^*$ or $\hat{x}_t^*$ relies on an initial point, then $\tilde{x}$ or $\hat{x}_t$ can be used as warm starts at no additional cost. By contrast, note that warm starts are not necessarily available when $\tilde{x}$ or $\hat{x}_t$ are first computed. *In this way, the computation of the bootstrap samples is easier than a naive repetition of the underlying sketching algorithm.*

4. **Error estimates can be extrapolated.**
   The basic idea of extrapolation is to estimate the error of a "rough" initial sketched solution, say $\tilde{x}_{\text{init}}$ or $\hat{x}_{\text{init}}$, and then predict how much additional computation should be done to obtain a better solution $\tilde{x}$ or $\hat{x}_t$. There are two main benefits of doing this. First, the computation is *adaptive*, in the sense that "just enough" work is done to achieve the desired degree of error. Secondly, when error estimation is based on the rough initial solution $\tilde{x}_{\text{init}}$, the bootstrap computations are substantially faster, because $\tilde{x}_{\text{init}}$ is constructed from a smaller sketching matrix than $\tilde{x}$ is. There are also two ways that extrapolation can be done — either with respect to the sketch size m, or the number of iterations t, and these techniques are outlined in the following paragraphs.

2.4. Extrapolating with respect to m for CS

The reasoning given in Section 2.1 based on the central limit theorem indicates that the standard deviation of $\|\tilde{x} - x_{\text{opt}}\|_\circ$ scales like $1/\sqrt{m}$ as a function of m. Therefore, if a rough initial solution $\tilde{x}_{\text{init}}$ is computed with a small sketch size m_0 satisfying $d < m_0 < m$, then the fluctuations of $\|\tilde{x}_{\text{init}} - x_{\text{opt}}\|_\circ$ should be larger than those of $\|\tilde{x} - x_{\text{opt}}\|_\circ$

by a factor of $\sqrt{m/m_0}$. This simple scaling relationship is useful to consider, because if we let $\tilde{\varepsilon}_{\text{init}}(\alpha)$ denote the error estimate obtained by applying Algorithm 1 to $\tilde{x}_{\text{init}}$, then it is natural to expect that the re-scaled quantity

$$\tilde{\varepsilon}_{\text{extrap},m}(\alpha) := \sqrt{\frac{m_0}{m}}\,\tilde{\varepsilon}_{\text{init}}(\alpha) \tag{21}$$

should be approximately equal to the ordinary estimate $\tilde{\varepsilon}(\alpha)$ for $\tilde{x}$. The advantage of $\tilde{\varepsilon}_{\text{extrap},m}(\alpha)$ is that it is cheaper to compute, since the bootstrapping can be done with a $m_0 \times d$ matrix, rather than an $m \times d$ matrix. Furthermore, once $\tilde{\varepsilon}_{\text{init}}(\alpha)$ has been computed, the user can instantly obtain $\tilde{\varepsilon}_{\text{extrap},m}(\alpha)$ as a function of m for all $m > m_0$, using the scaling rule (21). In turn, this allows the user to "look ahead" and see how large m should be chosen to achieve a desired level of accuracy. Simulations demonstrating the effectiveness of this technique are given in Section 4.

2.5. Extrapolating with respect to t for IHS

The IHS algorithm is known to enjoy linear convergence in the ℓ_2-norm under certain conditions (Pilanci & Wainwright, 2016). This means that the ith iterate $\hat{x}_i$ satisfies the following bound with high probability

$$\|\hat{x}_i - x_{\text{opt}}\|_2 \leq c\,\eta^i, \tag{22}$$

where $c > 0$ and $\eta \in (0,1)$ are unknown parameters that do not depend on i.

The simple form of this bound lends itself to extrapolation. Namely, if estimates $\hat{c}$ and $\hat{\eta}$ can be obtained after the first 2 iterations of IHS, then the user can construct the extrapolated error estimate

$$\hat{\varepsilon}_{\text{extrap},i}(\alpha) := \hat{c}\,\hat{\eta}^i, \tag{23}$$

which predicts how the error will decrease at all subsequent iterations $i \geq 3$. As a result, the user can adaptively determine how many extra iterations (if any) are needed for a specified error tolerance. Furthermore, with the help of Algorithm 2, it is straightforward to estimate c and η. Indeed, from looking at the condition (22), we desire estimates $\hat{c}$ and $\hat{\eta}$ that solve the two equations

$$\hat{c}\,\hat{\eta} = \hat{\varepsilon}_1(\alpha) \quad \text{and} \quad \hat{c}\,\hat{\eta}^2 = \hat{\varepsilon}_2(\alpha), \tag{24}$$

and direct inspection shows that the choices

$$\hat{\eta} := \tfrac{\hat{\varepsilon}_2(\alpha)}{\hat{\varepsilon}_1(\alpha)} \quad \text{and} \quad \hat{c} := \tfrac{\hat{\varepsilon}_1(\alpha)}{\hat{\eta}} \tag{25}$$

serve this purpose. In Section 4, our experiments show that this simple extrapolation procedure works remarkably well.

3. Main Result

In this section, we show that the estimates $\tilde{\varepsilon}(\alpha)$ and $\hat{\varepsilon}_t(\alpha)$ are consistent — in the sense that they satisfy the desired conditions (4) and (5) as the problem size becomes large. The setup and assumptions for our main result are given below.

Asymptotics. We consider an asymptotic framework involving a sequence of LS problems indexed by n. This means that we allow each of the objects $A = A(n)$, $S = S(n)$, and $b = b(n)$ to implicitly depend on n. Likewise, the solutions $\tilde{x} = \tilde{x}(n)$ and $\hat{x}_t = \hat{x}_t(n)$ implicitly depend on n.

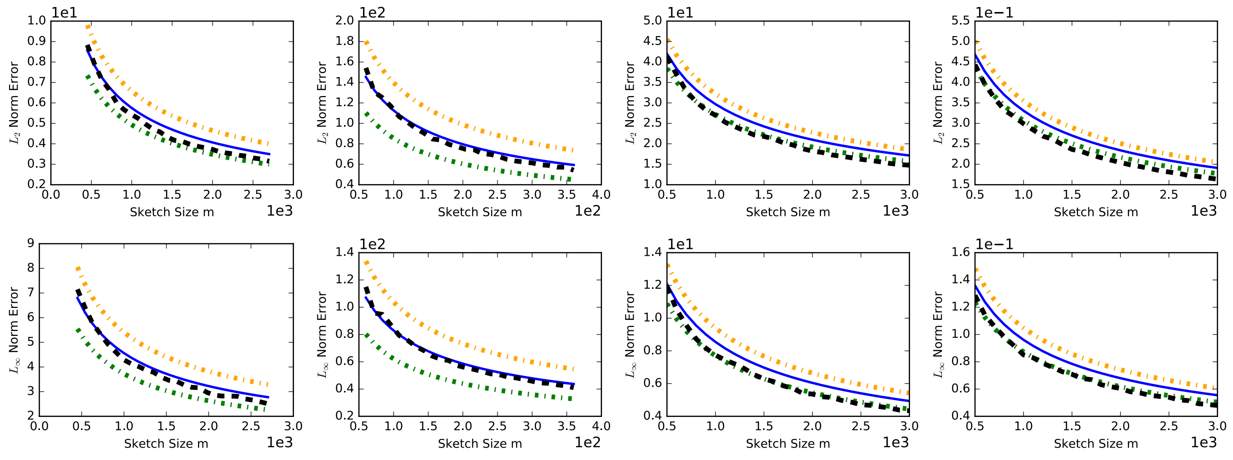
Since sketching algorithms are most commonly used when $d \ll n$, our results will treat d as fixed while $n \to \infty$. Also, the sketch size m is often selected as a large multiple of d, and so we treat $m = m(n)$ as diverging simultaneously with n. However, we make no restriction on the size of the ratio m/n, which may tend to 0 at any rate. In the same way, the number of bootstrap samples $B = B(n)$ is assumed to diverge with n, and the ratio B/n may tend to 0 at any rate. With regard to the number of iterations t, its dependence on n is completely unrestricted, and $t = t(n)$ is allowed to remain fixed or diverge with n. (The fixed case with $t = 1$ is of interest since it describes the HS algorithm.) Apart from these scaling conditions, we use the following two assumptions on A and b, as well as the sketching matrices.

Assumption 1. *The matrix $H_n := \frac{1}{n}A^\top A$ is positive definite for each n, and there is a positive definite matrix $H_\infty \in \mathbb{R}^{d\times d}$ such that $\sqrt{m}(H_n - H_\infty) \to 0$ as $n \to \infty$. Also, if $g_n := \frac{1}{n}A^\top b$, then there is a vector $g_\infty \in \mathbb{R}^d$ such that $\sqrt{m}(g_n - g_\infty) \to 0$. Lastly, let $a_1, \dots, a_n$ denote the rows of A, and let $e_1, \dots, e_n$ denote the standard basis vectors in $\mathbb{R}^n$. Then, for any fixed matrix $C \in \mathbb{R}^{d\times d}$ and fixed vector $c \in \mathbb{R}^d$, the sum $\frac{1}{n^2}\sum_{j=1}^n \big(a_j^\top C a_j + e_j^\top b c^\top a_j\big)^2$ converges to a limit (possibly zero) as $n \to \infty$.*

In essence, this assumption merely ensures that the sequence of LS problems is "asymptotically stable", in the sense that the solution x_{opt} does not change erratically from n to $n+1$.

Assumption 2. *In the case of CS, the rows of S are generated as i.i.d. vectors, where the ith row is of the form $\frac{1}{\sqrt{m}}(s_{i,1}, \dots, s_{i,n})$, and the random variables $s_{i,1}, \dots, s_{i,n}$ are i.i.d. with mean 0, variance 1, $\mathbb{E}[s_{1,1}^4] > 1$, and $\mathbb{E}[s_{1,1}^8] < \infty$. In addition, the distribution of $s_{1,1}$ remains fixed with respect to n, and in the case of IHS, the matrices $S_1, \dots, S_t$ are i.i.d. copies of S.*

Remarks. To clarify the interpretation of our main result, it is important to emphasize that A and b are viewed as deterministic, and the probability statements arise only from the randomness in the sketching algorithm, and the randomness in the bootstrap sampling. From an operational standpoint, the result says that as the problem size becomes large ($n \to \infty$), the outputs $\tilde{\varepsilon}(\alpha)$ and $\hat{\varepsilon}_t(\alpha)$ of our method will bound the errors $\|\tilde{x} - x_{\text{opt}}\|_\circ$ and $\|\hat{x}_t - x_{\text{opt}}\|_\circ$ with a probability that is effectively $1-\alpha$ or larger.



(a) MSD (b) CPU (c) Ill-Conditioned (d) Well-Conditioned

Figure 1. Numerical results for CS with extrapolation. The black dashed curve represents the ideal benchmark $\varepsilon_{\text{CS},m}(.05)$ described in the text. The average extrapolated estimate is shown in blue, with the yellow and green curves being one standard deviation away. Note: The upper row shows results for ℓ_2 error ($\|\cdot\|_\circ = \|\cdot\|_2$), and the lower row shows results for ℓ_∞ error ($\|\cdot\|_\circ = \|\cdot\|_\infty$).

Theorem 1. *Let $\|\cdot\|_\circ$ be any norm on $\mathbb{R}^d$, and suppose that Assumptions 1 and 2 hold. Also, for any number $\alpha \in (0,1)$ chosen by the user, let $\tilde{\varepsilon}(\alpha)$ and $\hat{\varepsilon}_t(\alpha)$ be the outputs of Algorithms 1 and 2 respectively. Then, there is a sequence of numbers $\delta_n > 0$ satisfying $\delta_n \to 0$ as $n \to \infty$, such that the following inequalities hold for all n,*

$$\mathbb{P}\Big(\|\tilde{x} - x_{\text{opt}}\|_\circ \le \tilde{\varepsilon}(\alpha)\Big) \ge 1 - \alpha - \delta_n, \tag{26}$$

and

$$\mathbb{P}\Big(\|\hat{x}_t - x_{\text{opt}}\|_\circ \le \hat{\varepsilon}_t(\alpha)\Big) \ge 1 - \alpha - \delta_n. \tag{27}$$

Remarks. Although this result can be stated in a concise form, the proof is actually quite involved (cf. (Lopes et al., 2018)). Perhaps the most significant technical obstacle is the sequential nature of the IHS algorithm. To handle the dependence of $\hat{x}_t$ on the previous iterates, it is natural to analyze $\hat{x}_t$ conditionally on them. However, because the set of previous iterates can grow with n, it seems necessary to establish distributional limits that hold "uniformly" over those iterates — and this need for uniformity creates difficulties when applying standard arguments.

More generally, to place this result within the context of the sketching literature, it is worth noting that guarantees for sketching algorithms typically show that a sketched solution is close to an exact solution with high probability (up to multiplicative constants). By contrast, Theorem 1 is more fine-grained, since it is concerned with *distributional approximation*, in terms of specific quantiles of the random variables $\|\tilde{x} - x_{\text{opt}}\|_\circ$ or $\|\hat{x}_t - x_{\text{opt}}\|_\circ$. In particular, the lower bounds are asymptotically equal to $1 - \alpha$ and *do not involve any multiplicative constants*. Lastly, it should also be noticed that the norm $\|\cdot\|_\circ$ is arbitrary, whereas it is more often the case that analyses of sketching algorithms are restricted to particular norms.

4. Experiments

In this section, we present experimental results in the contexts of CS and IHS. At a high level, there are two main takeaways: (1) The extrapolation rules accurately predict how estimation error depends on m or t, and this is shown in a range of conditions. (2) In all of the experiments, the algorithms are implemented with only $B = 20$ bootstrap samples. The fact that favorable results can be obtained with so few samples underscores the point that the method incurs only modest cost in exchange for an accuracy guarantee.

Data examples. Our numerical results are based on four linear regression datasets; two natural, and two synthetic. The natural datasets 'YearPredictionMSD', $n = 463{,}715$, $d = 90$, abbrev. MSD), and 'cpusmall' ($n = 8,192$, $d = 12$, abbrev. CPU) are available at the LIBSVM repository (Chang & Lin, 2011). The synthetic datasets are both of size ($n = 50{,}000, d = 100$), but they differ with respect to the condition number of $A^\top A$. The condition numbers in the 'Ill-conditioned' and 'Well-conditioned' cases are respectively 10^{12} and 10^2. (Details for the synthetic data are given in the supplement (Lopes et al., 2018).)

Experiments for CS. For each value of m in the grid $\{5d, \dots, 30d\}$, we generated 1,000 independent SRHT sketching matrices $S \in \mathbb{R}^{m \times n}$, leading to 1,000 realizations of of $(\tilde{A}, \tilde{b}, \tilde{x})$. Then, we computed the .95 sample quantile among the 1,000 values of $\|\tilde{x} - x_{\text{opt}}\|$ at each grid point. We denote this value as $\varepsilon_{\text{CS},m}(.05)$, and we view it as an ideal benchmark that satisfies $\mathbb{P}\big(\|\tilde{x} - x_{\text{opt}}\| \le \varepsilon_{\text{CS},m}(.05)\big) \approx .95$ for each m. Also, the

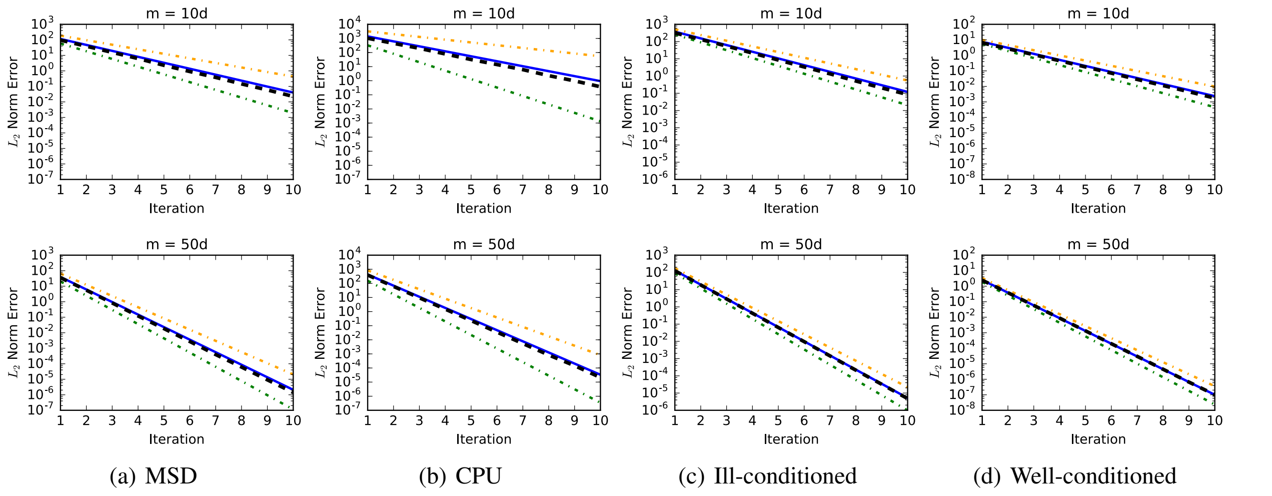


Figure 2. Numerical results for IHS with extrapolation. The black dashed curve represents the ideal benchmark $\varepsilon_{\text{IHS},i}(.05)$ described in the text. The average extrapolated estimate is shown in blue, with the yellow and green curves being one standard deviation away. The upper row shows results for $m = 10d$, and the lower row shows results for $m = 50d$.

value $\varepsilon_{\text{CS},m}(.05)$ is plotted as a function of m with the dashed black line in Figure 1. Next, using an initial sketch size of $m_0 = 5d$, we applied Algorithm 1 to each of the 1,000 realizations of $\tilde{A} \in \mathbb{R}^{m_0 \times d}$ and $\tilde{b} \in \mathbb{R}^{m_0}$ computed previously, leading to 1,000 realizations of the initial error estimate $\tilde{\varepsilon}_{\text{init}}(.05)$. In turn, we applied the extrapolation rule (21) to each realization of $\tilde{\varepsilon}_{\text{init}}(.05)$, providing us with 1,000 extrapolated curves of $\tilde{\varepsilon}_{\text{extrap},m}(.05)$ at all grid points $m \geq m_0$. The average of these curves is plotted in blue in Figure 1, with the yellow and green curves being one standard deviation away.

Comments on results for CS. An important conclusion to draw from Figure 1 is that the extrapolated estimate $\tilde{\varepsilon}_{\text{extrap},m}(.05)$ is a nearly unbiased estimate of $\tilde{\varepsilon}_{\text{CS},m}(.05)$ at values of m that are well beyond m_0. This means that in addition to yielding accurate estimates, the extrapolation rule (21) provides substantial computational savings — because the bootstrap computations can be done at a value m_0 that is much smaller than the value m ultimately selected for a higher quality $\tilde{x}$. Furthermore, these conclusions hold regardless of whether the error is measured with the ℓ_2-norm ($\|\cdot\|_\circ = \|\cdot\|_2$) or the ℓ_∞-norm ($\|\cdot\|_\circ = \|\cdot\|_\infty$), which correspond to the top and bottom rows of Figure 1.

Experiments for IHS. The experiments for IHS were organized similarly to the case of CS, except that the sketch size m was fixed (at either $m = 10d$, or $m = 50d$), and results were considered as a function of the iteration number. To be specific, the IHS algorithm was run 1,000 times, with $t = 10$ total iterations on each run, and with SRHT sketching matrices being used at each iteration. For a given run, the successive error values $\|\hat{x}_i - x_{\text{opt}}\|_2$ at $i = 1, \dots, 10$, were recorded. At each i, we computed the .95 sample quantile among the 1,000 error values, which is denoted as $\varepsilon_{\text{IHS},i}(.05)$, and is viewed as an ideal benchmark that satisfies $\mathbb{P}\big(\|\hat{x}_i - x_{\text{opt}}\|_2 \leq \varepsilon_{\text{IHS},i}(.05)\big) \approx .95$. In the plots, the value $\varepsilon_{\text{IHS},i}(.05)$ is plotted with the dashed black curve as a function of $i = 1, \dots, 10$. In addition, for each of the 1,000 runs, we applied Algorithm 2 at $i = 1$ and $i = 2$, producing 1,000 extrapolated values $\hat{\varepsilon}_{\text{extrap},i}(.05)$ at each $i \geq 3$. The averages of the extrapolated values are plotted in blue, and again, the yellow and green curves are obtained by adding or subtracting one standard deviation.

Comments on results for IHS. At a glance, Figure 2 shows that the extrapolated estimate stays on track with the ideal benchmark, and is a nearly unbiased estimate of $\varepsilon_{\text{IHS},i}(.05)$, for $i = 3, \dots, 10$. An interesting feature of the plots is how much the convergence rate of IHS depends on m. Specifically, we see that after 10 iterations, the choice of $m = 10d$ versus $m = 50d$ can lead to a difference in accuracy that is *4 or 5 orders of magnitude*. This sensitivity to m illustrates why selecting t is a non-trivial issue in practice, and why the extrapolated estimate can provide a valuable source of extra information to assess convergence.

5. Conclusion

We have proposed a systematic approach to answer a very practical question that arises for randomized LS algorithms: "How accurate is a given solution?" A distinctive aspect of the method is that it leverages the bootstrap — a tool ordinarily used for statistical inference — in order to serve a computational purpose. To our knowledge, it is also the first error estimation method for randomized LS that is supported theoretical guarantees. Furthermore, the method does not add much cost to an underlying sketching algorithm, and it has been shown to perform well on several examples.

Acknowledgements

MEL thanks the NSF for partial support of this work under grant DMS 1613218. MWM would like to thank the National Science Foundation, the Army Research Office, and the Defense Advanced Research Projects Agency for providing partial support of this work.

References

Ahfock, D., Astle, W. J., and Richardson, S. Statistical properties of sketching algorithms. *arXiv:1706.03665*, 2017.

Ailon, N. and Chazelle, B. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2006.

Ailon, N. and Liberty, E. Fast dimension reduction using Rademacher series on dual BCH codes. *Discrete & Computational Geometry*, 42(4):615–630, 2009.

Ainsworth, M. and Oden, J. T. *A Posteriori Error Estimation in Finite Element Analysis*, volume 37. John Wiley & Sons, 2011.

Avron, H., Maymounkov, P., and Toledo, S. Blendenpik: Supercharging LAPACK's least-squares solver. *SIAM Journal on Scientific Computing*, 32(3):1217–1236, 2010.

Bartels, S. and Hennig, P. Probabilistic approximate least-squares. In *Artificial Intelligence and Statistics (AISTATS)*, 2016.

Becker, S., Kawas, B., and Petrik, M. Robust partially-compressed least-squares. In *AAAI*, pp. 1742–1748, 2017.

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/.

Clarkson, K. L. and Woodruff, D. P. Low rank approximation and regression in input sparsity time. In *Annual ACM Symposium on theory of computing (STOC)*, 2013.

Colombo, N. and Vlassis, N. A posteriori error bounds for joint matrix decomposition problems. In *Advances in Neural Information Processing Systems (NIPS)*. 2016.

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

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. Sampling algorithms for ℓ_2 regression and applications. In *Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*, 2006.

Drineas, P., Mahoney, M. W., Muthukrishnan, S., and Sarlós, T. Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249, 2011.

Drineas, P., Magdon-Ismail, M., Mahoney, M. W., and Woodruff, D. P. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13:3441–3472, 2012.

Golub, G. H. and Van Loan, C. F. *Matrix Computations*. JHU Press, 2012.

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.

Jiránek, P., Strakoŝ, Z., and Vohralík, M. A posteriori error estimates including algebraic error and stopping criteria for iterative solvers. *SIAM Journal on Scientific Computing*, 32(3):1567–1590, 2010.

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. E. Estimating the algorithmic variance of randomized ensembles via the bootstrap. *The Annals of Statistics (to appear)*, 2018.

Lopes, M. E., Wang, S., and Mahoney, M. W. A bootstrap method for error estimation in randomized matrix multiplication. *arXiv:1708.01945*, 2017.

Lopes, M. E., Wang, S., and Mahoney, M. W. Error estimation for randomized least-squares algorithms via the bootstrap. *arXiv:1803.08021*, 2018.

Ma, P., Mahoney, M., and Yu, B. A statistical perspective on algorithmic leveraging. In *International Conference on Machine Learning (ICML)*, 2014.

Mahoney, M. W. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2): 123–224, 2011.

Meng, X., Saunders, M. A., and Mahoney, M. W. LSRN: A parallel iterative solver for strongly over - or underdetermined systems. *SIAM Journal on Scientific Computing*, 36(2):C95–C118, 2014.

Pang, J.-S. A posteriori error bounds for the linearly-constrained variational inequality problem. *Mathematics of Operations Research*, 12(3):474–484, 1987.

Pilanci, M. and Wainwright, M. J. Randomized sketches of convex programs with sharp guarantees. *IEEE Transactions on Information Theory*, 61(9):5096–5115, 2015.

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.

Rokhlin, V. and Tygert, M. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105(36): 13212–13217, 2008.

Sarlós, T. Improved approximation algorithms for large matrices via random projections. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

Verfürth, R. A posteriori error estimation and adaptive mesh-refinement techniques. *Journal of Computational and Applied Mathematics*, 50(1-3):67–83, 1994.

Woodruff, D. P. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014.

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.