Error Estimation for Sketched SVD via the Bootstrap

Miles E. Lopes [1] **N. Benjamin Erichson** [2] **Michael W. Mahoney** [2]

Abstract

In order to compute fast approximations to the singular value decompositions (SVD) of very large matrices, randomized sketching algorithms have become a leading approach. However, a key practical difficulty of sketching an SVD is that the user does not know how far the sketched singular vectors/values are from the exact ones. Indeed, the user may be forced to rely on analytical worst-case error bounds, which may not account for the unique structure of a given problem. As a result, the lack of tools for error estimation often leads to much more computation than is really necessary. To overcome these challenges, this paper develops a fully data-driven bootstrap method that numerically estimates the *actual* error of sketched singular vectors/values. Furthermore, the method is computationally inexpensive, because it operates only on sketched objects, and hence it requires *no extra passes* over the full matrix being factored.

1. Introduction

During the past fifteen years, randomized sketching algorithms have emerged as a powerful framework for computing approximate solutions to large-scale matrix problems in machine learning, data analysis, and scientific computing. Accordingly, given that the singular value decomposition (SVD) is among the most essential matrix computations in these domains, it has been a major focal point in the literature on randomized numerical linear algebra (RandNLA) (e.g., Frieze et al., 2004; Drineas et al., 2006; Rokhlin et al., 2010; Clarkson & Woodruff, 2009; Halko et al., 2011b;a; Woodruff, 2014; Musco & Musco, 2015; Tropp et al., 2019, among many others). Broadly speaking, this line of work has led to a variety of randomized SVD algorithms that can offer higher speed and scalability than classical deterministic algorithms — provided that the user is willing to tolerate some approximation error. For this reason, the performance of a sketched SVD hinges on an appropriate tradeoff between computational cost and approximation error. However, one of the key unresolved challenges for users is that the actual error is *unknown*, and as a result, it is hard to control the tradeoff efficiently.

In practice, this issue has typically been handled in one of two ways, each with their own limitations. The simplest option is to rely on informal rules of thumb for deciding how much computation to spend on a sketched SVD (e.g., in terms of the "sketch size"), but such rules give no warning when they fail, and this creates significant uncertainty in downstream computations. Alternatively, a more cautious option is to use analytical worst-case error bounds, but these present other challenges: First, these bounds often involve constants that are unspecified or dependent on unknown parameters. Second, even when explicit constants are available, worst-case bounds are necessarily pessimistic, and they may not account for the unique structure of a given problem.

Based on these concerns, the RandNLA literature has shown rising interest in *a posteriori error estimation*, which seeks to improve upon worst-case analysis by numerically quantifying error with data-driven methods (e.g., Liberty et al., 2007; Woolfe et al., 2008; Halko et al., 2011b; Martinsson & Voronin, 2016; Sorensen & Embree, 2016; Duersch & Gu, 2017; Lopes et al., 2018; Yu et al., 2018; Lopes et al., 2019b; Tropp et al., 2019; Chen & Lopes, 2020; Martinsson & Tropp, 2020). (Note that hereafter we will use the simpler phrase "error estimation".) Likewise, error estimation has the potential to make computations *data-adaptive*, so that "just enough" work is done to achieve a specific error tolerance for a specific input. Nevertheless, error estimation methods are still scarce for many sketching algorithms, and in particular, there has not yet been a systematic way to directly estimate the errors of the singular vectors/values in a sketched SVD. Therefore, the primary aim of the current paper is to develop a method for solving this problem. (More specific contributions associated with the method will be outlined in Section 1.3.)

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

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

1.1. Preliminaries on SVD and Sketching

Before we can explain the problem of error estimation in precise terms, it is necessary to briefly review a few aspects of classical SVD algorithms and their sketched versions.

Classical SVD. Let $A \in \mathbb{R}^{n\times d}$ be a very large deterministic input matrix with $n \geq d$. (In the case where $d \geq n$, all of our work can be applied to the transpose of A instead.) The SVD of A is a factorization of the form

$$A = U\Sigma V^\top,$$

where the matrix $U \in \mathbb{R}^{n\times d}$ has orthonormal columns $u_1, \ldots, u_d \in \mathbb{R}^n$ called left singular vectors, the matrix $V \in \mathbb{R}^{d\times d}$ has orthonormal columns $v_1, \ldots, v_d \in \mathbb{R}^d$ called right singular vectors, and the non-negative matrix $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_d) \in \mathbb{R}^{d\times d}$ contains the singular values $\sigma_1 \geq \cdots \geq \sigma_d$. It will also be convenient to refer to the "partial SVD", which returns $(u_1, \sigma_1, v_1), \ldots, (u_k, \sigma_k, v_k)$ for some $k \in \{1, \ldots, d\}$, and includes the ordinary SVD as a special case when $k = d$. In large-scale settings, it is often unaffordable to use classical (deterministic) algorithms to compute the partial SVD to machine precision. With respect to floating point operations, the $\mathcal{O}(ndk)$ cost of this computation can be prohibitive, but an even more severe obstacle arises with respect to communication costs. Namely, in the common situation when A is too large to be stored in fast memory, classical algorithms are often infeasible because they may require many passes over the entire matrix A (cf. Golub & Van Loan, 2012).

Sketched SVD. As a way of improving scalability, sketching algorithms proceed by mapping A to a much shorter matrix $\tilde{A} \in \mathbb{R}^{t\times d}$ with $t \ll n$, referred to as a "sketch" of A. More specifically, the matrix $\tilde{A}$ is constructed as

$$\tilde{A} = SA,$$

where $S \in \mathbb{R}^{t\times n}$ is a random "sketching matrix" that is generated by the user. In essence, the matrix S is generated so that $\tilde{A}$ captures enough information to approximately reconstruct $(u_1, v_1, \sigma_1), \ldots, (u_k, v_k, \sigma_k)$, and a myriad of choices for S have been proposed in the literature (cf. Mahoney, 2011; Woodruff, 2014; Kannan & Vempala, 2017, for overviews). Commonly, these choices ensure that the rows of S are i.i.d. vectors in $\mathbb{R}^n$, and that $\mathbb{E}[S^\top S] = I_n$. For instance, two of the most well-known choices are *Gaussian random projections (RP)*, where the rows of S are drawn from the Gaussian distribution $N(0, \frac{1}{t}I_n)$, and *row-sampling matrices (RS)*, where the rows of S are drawn from the set of re-scaled standard basis vectors $\{\frac{1}{\sqrt{tp_1}}\, e_1, \ldots, \frac{1}{\sqrt{tp_n}}\, e_n\} \subset \mathbb{R}^n$ with sampling probabilities $p_1, \ldots, p_n$.

Once the sketch $\tilde{A}$ has been obtained, the partial SVD of A can then be approximated with a variety of approaches that entail different costs and benefits. (We refer to the previously cited papers for further background.) Among these possibilities, our work will focus on the well-established *sketch-and-solve* approach, which has the merit of being highly "pass efficient" (in the sense of Drineas et al. (2006)), and hence inexpensive with respect to communication. In addition, it will follow as a consequence of our work that this approach has an extra benefit of being very amenable to error estimation. Furthermore, error estimation for sketch-and-solve is relevant to other approaches as well, because if the estimated error is large, then this can guide the user to consider a different approach that may deliver higher accuracy at higher cost.

To summarize how the sketch-and-solve approach works, it applies a classical partial SVD algorithm to the small matrix $\tilde{A}$ in order to compute its leading k singular values $\tilde{\sigma}_1, \ldots, \tilde{\sigma}_k$ and right singular vectors $\tilde{v}_1, \ldots, \tilde{v}_k \in \mathbb{R}^d$. Next, another set of vectors $\breve{u}_j := A\tilde{v}_j$ are computed for $1 \leq j \leq k$ and then normalized to yield $\tilde{u}_j := \breve{u}_j/\|\breve{u}_j\|_2$.[1] Then, the sequence $(\tilde{u}_1, \tilde{\sigma}_1, \tilde{v}_1), \ldots, (\tilde{u}_k, \tilde{\sigma}_k, \tilde{v}_k)$ is returned as an approximation to $(u_1, \sigma_1, v_1), \ldots, (u_k, \sigma_k, v_k)$. Altogether, the number of floating point operations involved is $\mathcal{O}(tdk + C_{\text{sketch}})$, where C_{sketch} is the cost to obtain $\tilde{A}$. In particular, for some popular types of row-sampling sketching matrices, this latter cost is $C_{\text{sketch}} = \mathcal{O}(nd)$, and hence only linear in the size of the input A. Furthermore, in terms of communication, this approach often *only requires 1 or 2 passes over A*.

1.2. The Error Estimation Problem

After the sketched quantities $(\tilde{u}_1, \tilde{\sigma}_1, \tilde{v}_1), \ldots, (\tilde{u}_k, \tilde{\sigma}_k, \tilde{v}_k)$ been computed, the user would (in principle) like to be able to compare them with the exact quantities $(u_1, \sigma_1, v_1), \ldots, (u_k, \sigma_k, v_k)$ in terms of various error measures. To unify our discussion, let $\rho(w, w')$ denote a generic non-negative measure of error for comparing two unit vectors w and w' of the same dimension. Also, since it is of interest to have uniform control of error over a general set of indices $\mathcal{J} \subset \{1, \ldots, k\}$, we will consider the following random error variables

$$\tilde{\epsilon}_{_U}(t) := \max_{j\in\mathcal{J}} \rho(\tilde{u}_j, u_j) \quad \text{and} \quad \tilde{\epsilon}_{_V}(t) := \max_{j\in\mathcal{J}} \rho(\tilde{v}_j, v_j),$$

as well as $\tilde{\epsilon}_{_\Sigma}(t) := \max_{j\in\mathcal{J}} |\tilde{\sigma}_j - \sigma_j|$. Although these random variables have been written as functions of t to indicate that they depend on the choice of t through the sketched quantities, it is important to note that $\tilde{\epsilon}_{_U}(t)$, $\tilde{\epsilon}_{_\Sigma}(t)$, and $\tilde{\epsilon}_{_V}(t)$ are *never observed* by the user.

Problem formulation. Our goal is to estimate the *tightest possible* upper bounds on $\tilde{\epsilon}_{_U}(t)$, $\tilde{\epsilon}_{_\Sigma}(t)$, and $\tilde{\epsilon}_{_V}(t)$ that hold

[1] In the unlikely case $A\tilde{v}_j = 0$, we put $\tilde{u}_j := 0$, and going forward, we will use this same convention when normalizing vectors.

with probability at least $1-\alpha$, for some desired choice of $\alpha \in (0,1)$. In statistical terminology, such bounds are called the $(1-\alpha)$-quantiles of the error variables, and will be denoted as $q_U(t)$, $q_\Sigma(t)$, and $q_V(t)$. More explicitly, $q_U(t)$ is an unknown deterministic parameter defined as

$$q_U(t) := \inf\left\{ q \in [0,\infty) \,\middle|\, \mathbb{P}\big(\tilde{\epsilon}_U(t) \le q\big) \ge 1-\alpha \right\},$$

and similarly for $q_\Sigma(t)$ and $q_V(t)$.

With the above notation in place, we propose to develop a fully data-driven method that will produce numerical quantile estimates $\widehat{q}_U(t)$, $\widehat{q}_\Sigma(t)$, and $\widehat{q}_V(t)$. Specifically, the proposed method is intended to satisfy two main criteria: (1) The estimates should be accurate substitutes for the true quantiles, in the sense that the event

$$\tilde{\epsilon}_U(t) \;\le\; \widehat{q}_U(t) \tag{1}$$

occurs with probability nearly equal to $1-\alpha$, and likewise for $\widehat{q}_\Sigma(t)$ and $\widehat{q}_V(t)$; (2) The method should be computationally affordable — so that the extra step of error estimation does not interfere with the overall benefit of sketching. Accordingly, our work in Sections 3, 4, and 5 will show that these criteria are achieved.

To give a more visual interpretation of the unknown quantile $q_U(t)$, we show in Figure 1 how it is related to the fluctuations of the error variable $\tilde{\epsilon}_U(t)$. If we imagine a hypothetical experiment where an oracle tells the user how $\tilde{\epsilon}_U(t)$ evolves as rows are incrementally added to a random sketching matrix S (up to $t = 3000$ rows), then the red curve displays this evolution. Similarly, the gray curves display the corresponding evolution over many independent repetitions of the same experiment. At any fixed t, the black curve represents the 0.95-quantile $q_U(t)$, which lies above $\tilde{\epsilon}_U(t)$ in 95% of the experiments. Lastly, it should be emphasized that the user is not able to see any of these curves in practice.

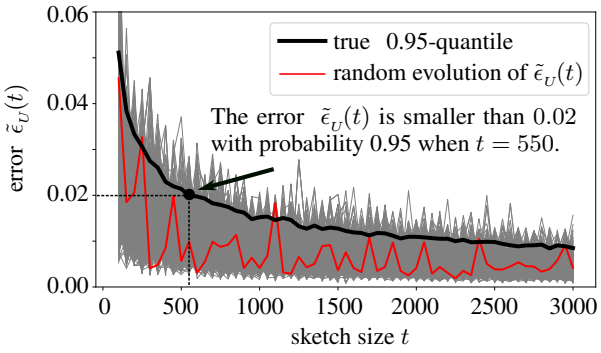

Figure 1. Visual interpretation of the quantile $q_U(t)$.

In this way, if the unknown curve $q_U(\cdot)$ were accessible, it could tell the user if a given initial sketch size t_0 is sufficient, and it could also allow the user to predict what larger sketch size $t_1 > t_0$ might be needed to achieve a smaller error. With this motivation in mind, our proposed method will allow the user to obtain an accurate approximation to the curve $q_U(\cdot)$. Furthermore, the method will produce the approximate curve after only a *single run* of the sketching algorithm at a *single initial sketch size* t_0. Indeed, it is somewhat surprising that this is possible, considering that $q_U(\cdot)$ theoretically describes the error over many runs.

1.3. Main Contributions and Related Work

From a practical standpoint, the most significant contribution of our work is that it provides the first way to directly estimate the errors of singular vectors/values in a sketched SVD. By comparison, the most closely related methods only provide indirect error estimates, since they are designed to compute norm bounds with respect to A and a low-rank approximant[2] (e.g., Liberty et al., 2007; Woolfe et al., 2008; Halko et al., 2011b; Sorensen & Embree, 2016; Yu et al., 2018; Tropp et al., 2019). In particular, these types of approaches have mostly been limited to either the Frobenius or spectral norms, and generally produce upper bounds on the norms that are conservative (cf. Yu et al., 2018, p.1342). On the other hand, our approach is very flexible with respect to the choice of error measure, and it does not suffer from conservativeness because it targets the quantiles of $\tilde{\epsilon}_U(t)$, $\tilde{\epsilon}_\Sigma(t)$, and $\tilde{\epsilon}_V(t)$. Another crucial distinction is that the cited approaches generally require 1 or more extra passes over A, whereas our approach requires *no extra passes*.

Other related work. In recent years, bootstrap methods for error estimation have been considered in a number of related settings. In the statistics literature, the papers (El Karoui & Purdom, 2019; Naumov et al., 2019) have analyzed the bootstrap as a way to estimate the errors of sample eigenvalues and sample eigenvectors in the context of large covariance matrices, which is a topic originating from the classical results in (Beran & Srivastava, 1985). However, due to their focus on covariance matrices, these works are not directly applicable to analyzing both the left and right singular vectors of a sketched SVD. Also, the theoretical setups in these works do not cover the important case of row-sampling sketches that is handled by our analysis. More generally, bootstrapping and other statistical approaches have recently been applied to quantify the errors of randomized least-squares and Newton methods (Ahfock et al., 2017; Lopes et al., 2018; Dobriban & Liu, 2019; Chen & Lopes, 2020), matrix multiplication (Lopes et al., 2019b;a), and gradient descent (Fang et al., 2018; Li et al., 2018; Su & Zhu, 2018; Fang, 2019). See also the forthcoming survey (Martinsson & Tropp, 2020, §4.5-4.6) for additional discussion of bootstrap methods in RandNLA.

[2]Note that the top singular vectors/values of two matrices can be close even when a norm distance between the matrices is large.

2. Method for Error Estimation

Intuition. If it were possible to generate many independent samples of the error variables $\tilde{\epsilon}_U(t)$, $\tilde{\epsilon}_\Sigma(t)$, and $\tilde{\epsilon}_V(t)$, it would be straightforward to construct estimates of $q_U(t)$, $q_\Sigma(t)$ and $q_V(t)$. For instance, if an oracle provided 100 independent samples, say $\tilde{\epsilon}_{V,1}(t), \ldots, \tilde{\epsilon}_{V,100}(t)$, of the error variable $\tilde{\epsilon}_V(t)$, then the 95th percentile of those 100 numbers would generally be a good estimate of $q_V(t)$ when α is chosen as 0.05. However, in practice, generating these samples is infeasible, because it would require the user to re-run the sketching algorithm many times, and then find the (unknown) sketching error for each run. In spite of this difficulty, it turns out that it *is possible* to efficiently generate *approximate* samples of the error variables, and this is the essence of the bootstrap approach.

Generating approximate samples. In order to generate approximate samples of $\tilde{\epsilon}_U(t)$, $\tilde{\epsilon}_\Sigma(t)$, and $\tilde{\epsilon}_V(t)$, the key idea is to create randomly "perturbed versions" of the sketched singular vectors/values. For example, the randomly perturbed version of $\tilde{v}_j$ is denoted as $\tilde{v}_j^*$, and it is intended to satisfy the following property. Namely, for each j, the fluctuations of $\tilde{v}_j^*$ around $\tilde{v}_j$ should be statistically similar to the fluctuations of $\tilde{v}_j$ around v_j. In other words, this idea can be understood in terms of a parallel "bootstrap world", where the vectors $\tilde{v}_j$ and $\tilde{v}_j^*$ respectively play the roles of exact and sketched solutions.

The only remaining ingredient to address is the random mechanism for computing the perturbed versions of $(\tilde{u}_1, \tilde{\sigma}_1, \tilde{v}_1), \ldots, (\tilde{u}_k, \tilde{\sigma}_k, \tilde{v}_k)$. In short, this is done by forming a matrix $\tilde{A}^* \in \mathbb{R}^{t \times d}$ whose rows are sampled with replacement from the rows of $\tilde{A}$, and then doing computations with $\tilde{A}^*$ that are analogous to the ones in the original sketching algorithm. These details are given in Algorithm 1.

As a matter of notation for expressing the outputs of Algorithm 1, it is necessary to define the empirical $(1-\alpha)$-quantile of a list of real numbers $x_1, \ldots, x_B$. This quantity is written as $\text{quantile}[x_1, \ldots, x_B; 1-\alpha]$, and is defined as $\inf\{q \in \mathbb{R} \,|\, F_B(q) \geq 1-\alpha\}$, where we write $F_B(q) := \frac{1}{B}\sum_{b=1}^{B} 1\{x_b \leq q\}$ for the empirical distribution function associated with $x_1, \ldots, x_B$.

Remarks. To clarify a couple of small items, we do not use a subscript b on the right sides of equations (2), (3), and (4) because only the left sides need to be stored. With regard to the number of bootstrap samples B, our experiments in Section 5 will show that the modest choice $B = 30$ works well in our settings of interest.

Algorithm 1 (Bootstrap estimation of sketching error).

Input: The sketch $\tilde{A} \in \mathbb{R}^{t \times d}$, the sketched sequence $(\tilde{\sigma}_1, \tilde{v}_1), \ldots, (\tilde{\sigma}_k, \tilde{v}_k)$, and the number of samples B.

- Compute the vectors $\tilde{A}\tilde{v}_1, \ldots, \tilde{A}\tilde{v}_k$ and let $\breve{u}_1, \ldots, \breve{u}_k$ denote their normalized versions with respect to the ℓ_2-norm.
- **For** $b = 1, \ldots, B$ **do in parallel**

1. Form a matrix $\tilde{A}^* \in \mathbb{R}^{t \times d}$ whose rows are obtained by sampling t rows with replacement from $\tilde{A}$.
2. Compute the top k singular values and right singular vectors of $\tilde{A}^*$, denoted as $\tilde{\sigma}_1^*, \ldots, \tilde{\sigma}_k^*$ and $\tilde{v}_1^*, \ldots, \tilde{v}_k^*$. Then, compute the bootstrap samples

$$\tilde{\epsilon}_{\Sigma,b}^*(t) := \max_{j \in \mathcal{J}} |\tilde{\sigma}_j^* - \tilde{\sigma}_j| \tag{2}$$

$$\tilde{\epsilon}_{V,b}^*(t) := \max_{j \in \mathcal{J}} \rho(\tilde{v}_j^*, \tilde{v}_j). \tag{3}$$

3. Compute the vectors $\tilde{A}\tilde{v}_1^*, \ldots, \tilde{A}\tilde{v}_k^*$ and let $\breve{u}_1^*, \ldots, \breve{u}_k^*$ denote their normalized versions with respect to the ℓ_2-norm. Then, compute the bootstrap sample

$$\tilde{\epsilon}_{U,b}^*(t) := \max_{j \in \mathcal{J}} \rho(\breve{u}_j^*, \breve{u}_j). \tag{4}$$

Return: The estimates $\widehat{q}_U(t)$, $\widehat{q}_\Sigma(t)$, and $\widehat{q}_V(t)$. They are defined as $\widehat{q}_U(t) := \text{quantile}[\tilde{\epsilon}_{U,1}^*(t), \ldots, \tilde{\epsilon}_{U,B}^*(t); 1-\alpha]$, and similarly for $\widehat{q}_\Sigma(t)$, and $\widehat{q}_V(t)$ using the samples generated in (2) and (3).

3. Computational Considerations

Given that sketching algorithms from RandNLA are intended to improve the efficiency of computations, it is important to explain why the extra step of error estimation does not interfere with this goal. Below, we describe some special aspects of Algorithm 1 that make error estimation affordable.

Pass efficiency and scalability. Because the inputs to Algorithm 1 consist entirely of sketched objects, it follows that error estimation *requires no access* to the full matrix A (i.e., zero extra passes). Furthermore, this also means that the processing cost of Algorithm 1 is *independent of the large dimension* n. To put these two features of Algorithm 1 into context, it is worth noting that the sketched SVD typically requires at least 1 or 2 passes over A, and typically has a processing cost that is linear in n. Hence, from this standpoint, if the user can afford to compute a sketched SVD, then the extra step of error estimation should be affordable as well.

Parallelism and cloud/serverless computing. The loop in Algorithm 1 can be executed in an embarrasingly parallel manner, since each iteration $b = 1, \ldots, B$ is independent of the others. In addition, the computations at each iteration only have a small $\mathcal{O}((t \vee k)d)$ memory requirement, which is well-suited to modern distributed computing environments, such as cloud/serverless computing (cf. Kleiner et al., 2014; Jonas et al., 2019). Likewise, it is natural to consider distributing the B iterations across $\mathcal{O}(B)$ machines, and in this case, the processing cost of Algorithm 1 is only

$\mathcal{O}(tdk)$ on a per-machine basis (for common choices of ρ). In fact, our experiments in Section 5 show that when A is on the order of 100GB, it is possible to obtain high quality error estimates *in a matter of seconds* when Algorithm 1 is distributed across B machines.

Extrapolation. One more valuable feature of Algorithm 1 is that it can be substantially accelerated via an *extrapolation rule*. At a high level, this refers to a two-step process of (1) computing a "rough" sketched SVD based on an initial sketch size t_0, and then (2) using Algorithm 1 to forecast what larger sketch size $t_1 > t_0$ is sufficient to achieve a desired error tolerance. At a more technical level, the extrapolation rule may be derived from the fact that the error variables $\tilde{\epsilon}_U(t)$, $\tilde{\epsilon}_\Sigma(t)$, and $\tilde{\epsilon}_V(t)$ tend to have fluctuations on the order of $1/\sqrt{t}$ (due to the central limit theorem).

Based on this anticipated scaling behavior, the error $\tilde{\epsilon}_U(t_0)$ at an initial sketch size t_0 should be larger than the error $\tilde{\epsilon}_U(t_1)$ at a sketch size $t_1 > t_0$ by a factor of about $\sqrt{t_1/t_0}$. Hence, this suggests that if we use Algorithm 1 to obtain an error estimate $\widehat{q}_U(t_0)$ from the initial sketched SVD, then we can re-scale this estimate by a factor of $\sqrt{t_0/t_1}$ to get a "free" estimate of $q_U(t_1)$. In other words, we may define the extrapolated error estimate

$$\widehat{q}_U^{\text{ext}}(t_1) \;:=\; \tfrac{\sqrt{t_0}}{\sqrt{t_1}}\,\widehat{q}_U(t_0) \tag{5}$$

for any choice of t_1 greater than t_0, and likewise for $\widehat{q}_\Sigma^{\text{ext}}(t_1)$ and $\widehat{q}_V^{\text{ext}}(t_1)$.

The crucial point to notice about the extrapolation rule (5) is that running Algorithm 1 based on a sketch of size t_0 is much cheaper than a sketch of size t_1 (by a factor of t_1/t_0 per iteration). Moreover, it turns out that this rule provides accurate estimates even when t_1 is larger than t_0 by *an order of magnitude*, and this will be demonstrated empirically in Section 5. Altogether, this allows the user to allocate computational resources in a way that is *adaptive* to the input at hand.

4. Theory

In this section, we present our main theoretical result (Theorem 1), which shows that all three quantile estimates $\widehat{q}_U(t)$, $\widehat{q}_\Sigma(t)$, and $\widehat{q}_V(t)$ produced by Algorithm 1 are asymptotically valid substitutes for the unknown quantiles $q_U(t)$, $q_\Sigma(t)$, and $q_V(t)$. Furthermore, the result is applicable to either of the cases where $n \gg d$ or $d \gg n$. For brevity, we will deal only with the former case, because the latter case can be handled by considering the transpose of A.

Theoretical setup. Our result is formulated in terms of a sequence of deterministic matrices $A_n \in \mathbb{R}^{n\times d}$ indexed by $n = 1, 2, \ldots$, such that d remains fixed as $n \to \infty$. Likewise, the number $k \in \{1, \ldots, d\}$ and the set of indices $\mathcal{J} \subset \{1, \ldots, k\}$ remain fixed as well. In addition, for each n, there is an associated random sketching matrix $S_n \in \mathbb{R}^{t_n \times n}$ and a number of bootstrap samples B_n such that $t_n \to \infty$ and $B_n \to \infty$ as $n \to \infty$. Here, it is important to note that we make no restriction on the sizes of t_n and B_n relative to n, and hence we allow $t_n/n \to 0$ and $B_n/n \to 0$. Lastly, in order to lighten notation in Theorem 1, we will suppress dependence on n for the outputs of Algorithm 1, as well the exact singular vectors/values (u_j, v_j, σ_j) of A_n and their sketched versions $(\tilde{u}_j, \tilde{v}_j, \tilde{\sigma}_j)$.

With regard to the choice of error measure ρ for the sketched singular vectors, we will focus on the "sine distance" $\rho_{\sin}$, defined for any Euclidean unit vectors w and w' of the same dimension as

$$\rho_{\sin}(w, w') := \sqrt{1 - (w^\top w')^2}. \tag{6}$$

This is a standard error measure in the analysis of SVD, because it is invariant to sign changes of w and w', and hence automatically handles the sign ambiguity of singular vectors (cf. Stewart & Sun, 1990; Anderson et al., 1999). Its name derives from the fact that it can be interpreted as the sine of the acute angle between the one-dimensional subspaces spanned by w and w'.

Next, we state some assumptions for analyzing different types of sketching matrices. When S_n is a Gaussian random projection, we make the following assumption.

Assumption RP. *There is a positive definite matrix G_∞ in $\mathbb{R}^{d\times d}$ such that $\frac{1}{n}A_n^\top A_n \to \mathsf{G}_\infty$ as $n \to \infty$, and the eigenvalues of G_∞ each have multiplicity 1.*

In the case when S_n is a row-sampling matrix, we will use an assumption that augments Assumption RP with a few conditions. To state these conditions, let $(p_1, \ldots, p_n)$ denote the row-sampling probabilities for S_n, and let $a_l \in \mathbb{R}^d$ denote the lth row of A_n. In addition, let $\tilde{r}_n \in \mathbb{R}^d$ denote the first row of the re-scaled sketch $\frac{\sqrt{t}}{\sqrt{n}} S_n A_n$, and let $\mathsf{v}_1, \mathsf{v}_2 \in \mathbb{R}^d$ denote the top two eigenvectors of G_∞.

Assumption RS. *The following conditions hold in addition to Assumption RP. For any fixed matrix $C \in \mathbb{R}^{d\times d}$, the sequence $\mathrm{var}(\tilde{r}_n^\top C \tilde{r}_n)$ converges to a finite limit $\ell(C)$, possibly zero, as $n \to \infty$. Furthermore, if C is chosen as $C = \mathsf{v}_1\mathsf{v}_1^\top$ or $C = \mathsf{v}_1\mathsf{v}_2^\top$, then the limit $\ell(C)$ is positive. Lastly, the condition $\max_{1\le l\le n} \|\frac{1}{\sqrt{np_l}} a_l\|_2 = o(t_n^{1/8})$ holds as $n \to \infty$.*

Remarks. To provide some explanation for Assumptions RP and RS, the first mostly plays the role of a "stability" condition, which ensures that various functions of A_n have well-behaved limits as $n \to \infty$. In particular, the $\frac{1}{n}$ prefactor of the matrix $\frac{1}{n}A_n^\top A_n$ is natural because it allows the matrix to be written as the average $\frac{1}{n}\sum_{l=1}^n a_l a_l^\top$. Also, the requirement that the eigenvalues of G_∞ have multiplicity 1 is used so that tools from matrix calculus can be applied to

the matrix $\frac{1}{n}A_n^\top A_n$ within a neighborhood of G_∞. (Without a requirement of this type, the functions that send a matrix to its eigenvectors/values become non-differentiable (Magnus & Neudecker, 2019, Ch. 9.8).) Next, the conditions in Assumption RS are needed to rule out certain extreme types of matrices A_n that interfere with techniques related to the central limit theorem. (For instance, if all the rows of A_n are identical, then all sketches obtained by row sampling will be identical, and then the error variables will have degenerate distributions.) Lastly, in the supplementary material, we provide detailed examples of matrices A_n that satisfy both assumptions.

In a nutshell, our main result shows that for large problems, Algorithm 1 provides estimates $\widehat{q}_U(t)$, $\widehat{q}_\Sigma(t)$, and $\widehat{q}_V(t)$ that nearly achieve the ideal coverage probability of $1-\alpha$, as in (1). In addition, it is worth noting that the probability $\mathbb{P}$ in Theorem 1 accounts for all sources of randomness (both from the sketching matrix and bootstrap sampling).

Theorem 1. *Suppose that Assumption RP holds when S_n is a Gaussian random projection, or that Assumption RS holds when S_n is a row-sampling matrix. Also, let $\widehat{q}_U(t_n)$, $\widehat{q}_\Sigma(t_n)$, and $\widehat{q}_V(t_n)$ denote the outputs of Algorithm 1. Then, for any fixed set $\mathcal{J} \subset \{1,\ldots,k\}$ containing 1, and any $\alpha \in (0,1)$, the following three limits hold as $n \to \infty$,*

$$\mathbb{P}\Big(\max_{j\in\mathcal{J}} \rho_{\sin}(\tilde{u}_j, u_j) \le \widehat{q}_U(t_n)\Big) \to 1-\alpha, \tag{7}$$

$$\mathbb{P}\Big(\max_{j\in\mathcal{J}} |\tilde{\sigma}_j - \sigma_j| \le \widehat{q}_\Sigma(t_n)\Big) \to 1-\alpha, \tag{8}$$

$$\mathbb{P}\Big(\max_{j\in\mathcal{J}} \rho_{\sin}(\tilde{v}_j, v_j) \le \widehat{q}_V(t_n)\Big) \to 1-\alpha. \tag{9}$$

Remarks. The proof is deferred to the appendices due its length. The main theoretical challenge is to establish central limit theorems for each of the random variables $\max_{j\in\mathcal{J}} \rho_{\sin}(\tilde{u}_j, u_j)$, $\max_{j\in\mathcal{J}} |\tilde{\sigma}_j - \sigma_j|$, and $\max_{j\in\mathcal{J}} \rho_{\sin}(\tilde{v}_j, v_j)$, as well as their bootstrap analogues. In carrying this out, some of the essential technical ingredients are explicit formulas for matrix differentials (Jacobians) associated the functions that send a matrix to its eigenvectors/values (Magnus & Neudecker, 2019, Ch. 9.8). More specifically, these formulas play an important role in determining the asymptotic variance in the central limit theorems just mentioned. Another notable technical point is that the analysis handles the left singular vectors (in $\mathbb{R}^n$) and the right singular vectors (in $\mathbb{R}^d$) in a streamlined way — even though the left singular vectors have a *diverging dimension* as $n \to \infty$.

Lastly, to understand how Theorem 1 fits into the broader context of the literature on sketched SVD, it should be emphasized that our analysis is based on *distributional approximation*, whereas most other theoretical work has been based on tail bounds. The key benefit of distributional approximation is that it allows us to show that the coverage probabilities of $\widehat{q}_U(t_n)$, $\widehat{q}_\Sigma(t_n)$ and $\widehat{q}_V(t_n)$ approach the ideal value of $1-\alpha$. By contrast, tail bounds are often only able to quantify such probabilities up to constants that may be unspecified or conservative.

5. Experiments

In this section, we present a collection of synthetic and natural examples that demonstrate the practical performance of Algorithm 1. In particular, we show that the extrapolation rule (5) accurately predicts error as a function of the sketch size t. For simplicity, all the synthetic examples in Section 5.1 deal with the approximation of the leading triple (u_1, σ_1, v_1), so that the error variables $\tilde{\epsilon}_U(t)$, $\tilde{\epsilon}_\Sigma(t)$, and $\tilde{\epsilon}_V(t)$ correspond to the index set $\mathcal{J} = \{1\}$. Other choices of the index set $\mathcal{J}$ are considered for real data in Section 5.2, as well as for synthetic data in the supplementary material. Also, the sine distance (6) will be used in all examples as the measure of error for the singular vectors, and α will always be set to 0.05.

5.1. Synthetic Examples

First, we consider tall synthetic matrices with $(n,d) = (10^5, 3000)$ that are characterized by low effective rank and varying degrees of singular value decay.

Parameter settings. The matrix A was specified in terms of the three factors U, Σ, and V of its SVD. The factors U and V were generated at random from the uniform (Haar) distributions on the sets of orthonormal matrices of sizes $n \times d$ and $d \times d$ respectively. The singular values of A were chosen as $\Sigma = \mathrm{diag}(1^{-\beta}, 2^{-\beta}, \ldots, d^{-\beta})$ for three choices of the decay parameter $\beta \in \{0.5, 1.0, 2.0\}$.

Design of experiments. For each choice of the sketch size t in a grid ranging from 500 up to 6000, we generated 500 independent sketching matrices $S \in \mathbb{R}^{t\times n}$, which yielded 500 realizations of $\tilde{A} \in \mathbb{R}^{t\times d}$. Here, we used "squared-length sampling" (Frieze et al., 2004) to construct the sketch $\tilde{A}$ in each trial, since it is a popular option for row sampling. (In the supplement, we also provide experiments with sketches obtained from the subsampled randomized Hadamard transform (Ailon & Chazelle, 2006).) Next, each realization of $\tilde{A}$ yielded error variables $\epsilon_U(t)$, $\epsilon_\Sigma(t)$, and $\epsilon_V(t)$. In turn, we treated the empirical 95th percentiles of these 500 realizations as ground truth for the ideal quantiles $q_U(t)$, $q_\Sigma(t)$, and $q_V(t)$, plotted with black dashed lines in Figure 2.

Algorithm 1 was applied to the sketched SVD resulting from each of the 500 matrices $\tilde{A}$ at each sketch size t in the grid, using a choice of $B = 30$ in every instance. In total, this produced 500 realizations of $\widehat{q}_U(t)$, $\widehat{q}_\Sigma(t)$ and $\widehat{q}_V(t)$ at each t. The respective averages of these 500 estimates at each t are plotted with solid blue lines in Figure 2.

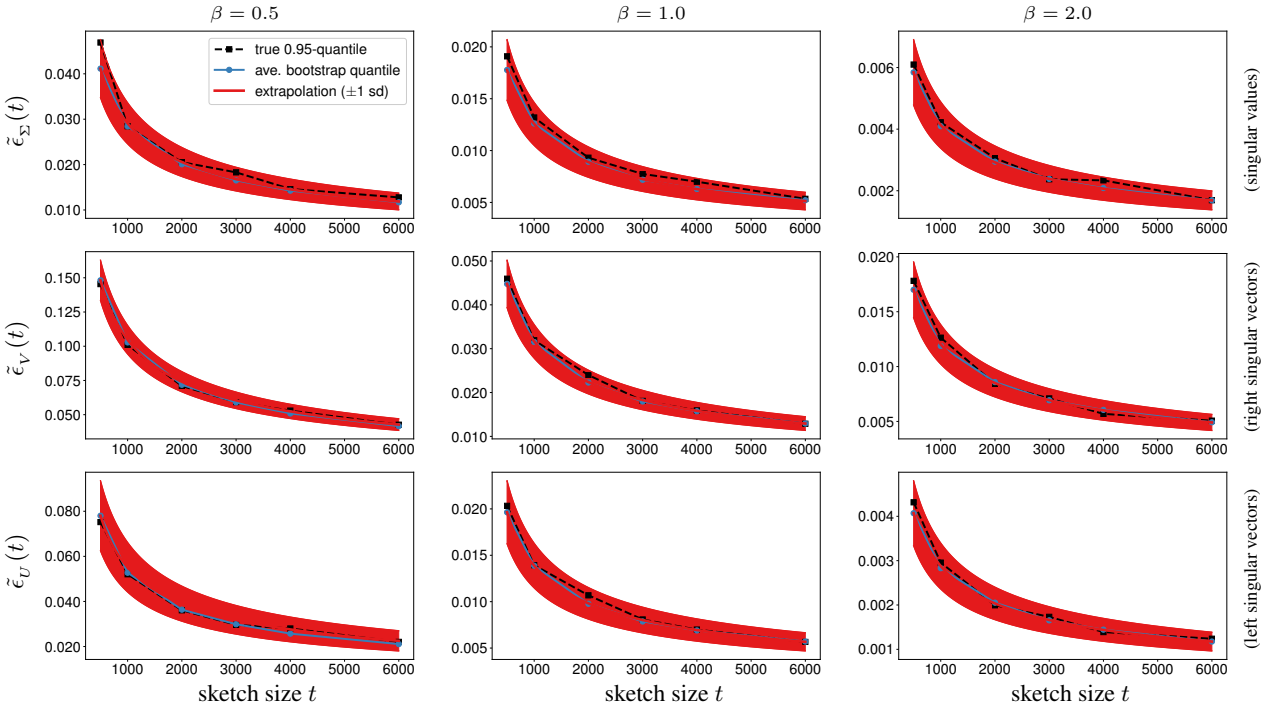

Figure 2. We consider artificial matrices of dimension $(n, d) = (10^5, 3 \times 10^3)$ that have singular value decay profiles of the form $\sigma_j = j^{-\beta}$ for $j \in \{1, \ldots, d\}$ with $\beta \in \{0.5, 1.0, 2.0\}$. The error variables correspond to the index set $\mathcal{J} = \{1\}$, and the simulations involve 500 trials and 30 bootstraps per trial. The three rows of plots correspond to the error quantiles for the singular values (top), right singular vectors (middle), and left singular vectors (bottom).

To study the performance of the extrapolation rule (5), we applied it to each of the 500 quantile estimates produced at $t_0 = 500$, which resulted in 500 realizations of each of the curves $\widehat{q}_U^{\text{ext}}(\cdot)$, $\widehat{q}_\Sigma^{\text{ext}}(\cdot)$, and $\widehat{q}_V^{\text{ext}}(\cdot)$. The respective averages of each type of curve are plotted with solid red lines in Figure 2, and the light red envelopes represent ± 1 standard deviation around the average.

Results for synthetic examples. The results show that the bootstrap quantile estimates, as well as their extrapolated versions, are excellent approximations to the true quantiles over the entire range of t. This behavior is also consistent across the different decay parameters $\beta \in \{0.5, 1.0, 2.0\}$. Moreover, we see that this performance holds when the true quantiles range over several different orders of magnitude. Hence, even in situations where the sketching errors are larger, the bootstrap is helfpul because it can tell the user that a higher precision SVD algorithm may be needed to reach a given error tolerance.

5.2. Examples from Applications

Now we turn to some examples arising from applications in climate science and fluid dynamics.

Sea surface temperature data. In the analysis of sea-surface temperature (SST) data, principal components (henceforth called "modes") play an important role in visualizing the structure of climate patterns. Due to the massive scale of such data, it is impractical to use classical SVD algorithms, but fortunately, it is often possible to gain clear physical insights from approximate computations. Consequently, this application is well-suited to sketching algorithms (Erichson et al., 2020; 2019).

We consider satellite-based recordings of SST data collected during the years 1981 to 2019, comprising $d = 14001$ temporal snapshots (Reynolds et al., 2007). Each snapshot measures the daily temperature means at $n = 691150$ spatial grid points across the globe. In total, the data requires 72GB in storage.

The left panel of Figure 3 shows the performance of the bootstrap estimate $\widehat{q}_U(t)$, where the experiments were organized in the same way as in Section 5.1, and the results are plotted in the same format. In addition to the fact that the extrapolated estimates are fairly accurate, there are two other aspects of this example that are especially encouraging: (1) The intial sketch size $t_0 = 500$ corresponds to an extremely small fraction ($500/691150 \approx 0.0007$) of the data. (2) When Algorithm 1 was distributed across 30 machines, it was possible to generate $B = 30$ bootstrap samples at $t_0 = 500$ in *less than 4 seconds* (including overhead costs). Thus, this is fast enough to provide the user with error estimates on a time scale that is compatible with

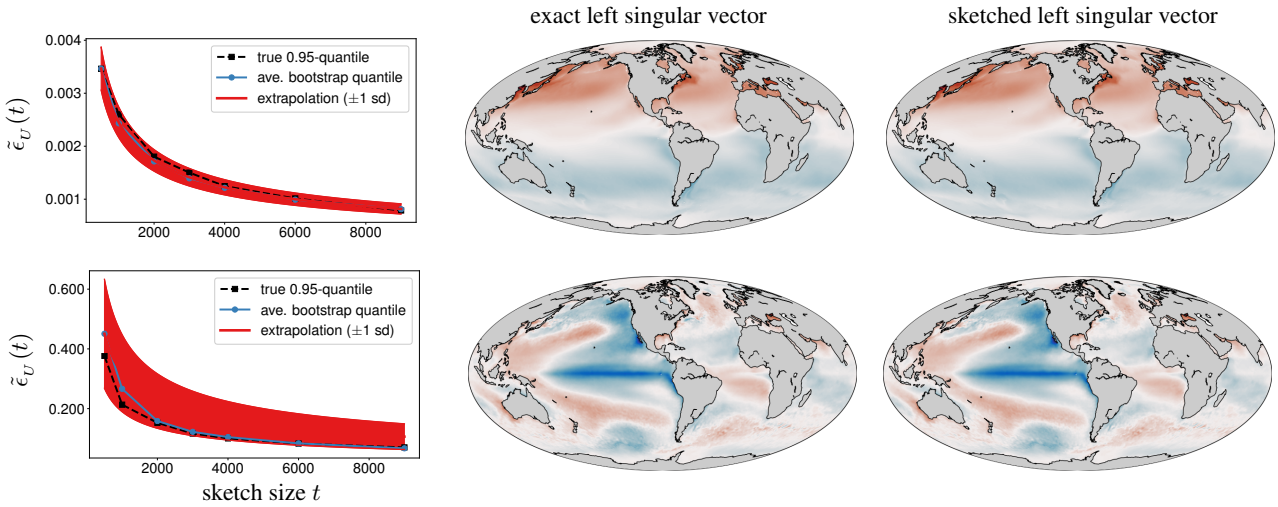

Figure 3. Bootstrap error estimates for the 1st and 4th sketched left singular vectors of the SST dataset (691150×14001), using squared-length sampling. The error variable in the top left plot corresponds to the index set $\mathcal{J} = \{1\}$ and the error variable in the bottom left plot corresponds to $\mathcal{J} = \{4\}$. The simulations involve 500 trials and 30 bootstraps per trial. In addition we show the (exact) deterministic mode and a single instance of a sketched mode using $t = 3000$. At this sketch size, we can see that with 95% probability, the error (sine distance) is less than 0.002 for $\mathcal{J} = \{1\}$, and less than 0.2 for $\mathcal{J} = \{4\}$. In the globe plots, red indicates warm temperatures, while blue indicates cold temperatures.

interactive data analysis.

The right panel of Figure 3 gives a visual comparison between the exact and sketched versions of the 1st and 4th modes. More specifically, the modes are visualized by projecting their 691150 entries onto a set of geospatial coordinates. For the 1st mode, the extrapolated bootstrap method would reliably tell the user that a sketch size of $t = 3000$ corresponds to a sine distance of less than 0.002 with 95% probability, which conforms with the fact that the exact and sketched modes are nearly indistinguishable to the human eye. In the case of the 4th mode, the extrapolated bootstrap method overestimates the error, but only by a small amount.

Large adjacency matrices in fluid dynamics. Computing the eigenvectors of very large adjacency matrices is a frequently encountered problem in many application domains. When these matrices are dense, eigenvector computations are especially costly, which makes sketching algorithms a natural approach. As an illustration of this type of situation, we consider a dense symmetric adjacency matrix that encodes dynamics in a fluid flow system. In this context, the eigenvectors carry information about the strength of vortices in the system. (We refer to (Bai et al., 2019) for further background.) Specifically, the adjacency matrix is of size $n \times n$ with $n = 116964$, which requires 101GB of storage.

The left panel of Figure 4 shows the performance of $\widehat{q}_V(t)$, where we note that v_1 corresponds to the top eigenvector, since A is symmetric. (The experiments here were designed in the same way as in Section 5.1.) From looking at the left panel, we see that the extrapolated bootstrap estimates are accurate over a large range of sketch sizes. Also, by distributing Algorithm 1 across 30 machines, it was possible to generate $B = 30$ bootstrap samples at $t_0 = 500$ in only 11.5 seconds (including overhead costs). Indeed, this is a remarkably short amount of time for error estimation in the context of a 101GB matrix.

The right panel of Figure 4 gives a visual comparison of the exact and sketched eigenvectors, where blue/red correspond to strong/weak vortices. As in the case of the SST data, this comparison shows that Algorithm 1 can provide the user with reliable confirmation that the sketched approximation is of high quality.

6. Conclusion

In this work, we developed a fully data-driven bootstrap method that numerically estimates the *actual* error of sketched singular vectors/values. From a practical standpoint, this allows the user to inspect the quality of a rough initial sketched SVD, and then adaptively predict how much extra work is needed to reach a given error tolerance. Also, our numerical results show that the estimates are accurate for choices of A in a range of conditions, including some large-scale applications related to fluid dynamics and climate science.

Computationally, our method readily scales to very large problems by taking advantage of inherent speedups based on parallelism and extrapolation. In fact, these speedups are so substantial that even when A is on the order of 100GB, it is possible to obtain high quality error estimates *within a*

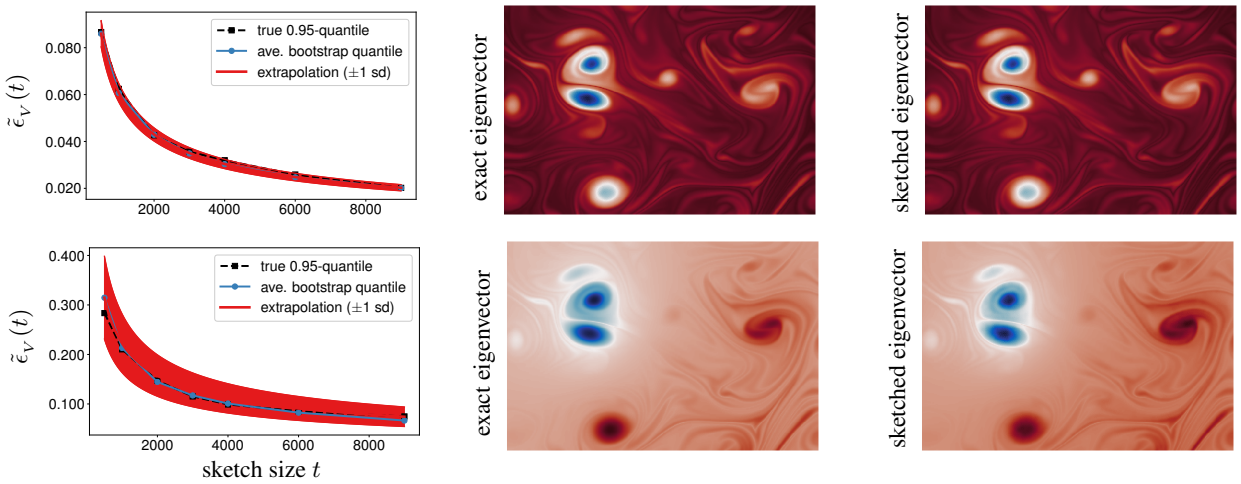

Figure 4. Bootstrap error estimates for the dominant two sketched eigenvectors of a dense adjacency matrix (116964×116964), using squared-length sampling. Here, the error variables correspond to the index set $\mathcal{J} = \{1\}$ (top) and $\mathcal{J} = \{2\}$ (bottom), respectively. The simulations involve 500 trials and 30 bootstraps per trial. In addition we show the (cropped) deterministic and sketched mode using $t = 6000$. At this sketch size, we can see that with 95% probability, the error (sine distance) is less than 0.03 for $\mathcal{J} = \{1\}$, and less than 0.1 for $\mathcal{J} = \{2\}$.

matter of seconds after a sketched SVD has been computed.

Theoretically, we have shown in Theorem 1 that the quantile estimates $\widehat{q}_U(t)$, $\widehat{q}_\Sigma(t)$, and $\widehat{q}_V(t)$ are consistent, in the sense that they achieve the desired coverage probability as the size of the problem becomes large.

Acknowledgements

MEL gratefully acknowledges partial support from NSF grants DMS-1613218 and DMS-1915786. MWM would like to acknowledge DARPA, NSF, ONR, and Intel for providing partial support of this work. NBE gratefully acknowledges Amazon Web Services for supporting this project with EC2 credits. Further, we would like to acknowledge the NOAA for providing the SST data (https://www.esrl.noaa.gov/psd/).

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 *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pp. 557–563, 2006.

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. *LAPACK Users' Guide*. SIAM, third edition, 1999.

Bai, Z., Erichson, N. B., Meena, M. G., Taira, K., and Brunton, S. L. Randomized methods to characterize large-scale vortical flow networks. *PloS one*, 14(11), 2019.

Beran, R. and Srivastava, M. S. Bootstrap tests and confidence regions for functions of a covariance matrix. *The Annals of Statistics*, 13(1):95–115, 1985.

Chen, X. and Lopes, M. E. Estimating the error of randomized Newton methods: A bootstrap approach. In *International Conference on Machine Learning (to appear)*, 2020.

Clarkson, K. L. and Woodruff, D. P. Numerical linear algebra in the streaming model. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pp. 205–214, 2009.

Dobriban, E. and Liu, S. Asymptotics for sketching in least squares regression. In *Advances in Neural Information Processing Systems*, pp. 3670–3680, 2019.

Drineas, P., Kannan, R., and Mahoney, M. W. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183, 2006.

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

El Karoui, N. and Purdom, E. The non-parametric bootstrap and spectral analysis in moderate and high-dimension. In

The 22nd International Conference on Artificial Intelligence and Statistics, pp. 2115–2124, 2019.

Erichson, N. B., Mathelin, L., Kutz, J. N., and Brunton, S. L. Randomized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 18:1867–1891, 2019.

Erichson, N. B., Zheng, P., Manohar, K., Brunton, S. L., Kutz, J. N., and Aravkin, A. Y. Sparse principal component analysis via variable projection. *SIAM Journal on Applied Mathematics*, 80(2):977–1002, 2020.

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.

Frieze, A., Kannan, R., and Vempala, S. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

Golub, G. and Van Loan, C. *Matrix computations*. Matrix Computations. Johns Hopkins University Press, 2012.

Halko, N., Martinsson, P.-G., Shkolnisky, Y., and Tygert, M. An algorithm for the principal component analysis of large data sets. *SIAM Journal on Scientific Computing*, 33(5):2580–2594, 2011a.

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, 2011b.

Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C.-C., Khandelwal, A., Pu, Q., Shankar, V., Carreira, J., Krauth, K., Yadwadkar, N., et al. Cloud programming simplified: A Berkeley view on serverless computing. *arXiv:1902.03383*, 2019.

Kannan, R. and Vempala, S. Randomized algorithms in numerical linear algebra. *Acta Numerica*, 26:95–135, 2017.

Kleiner, A., Talwalkar, A., Sarkar, P., and Jordan, M. I. A scalable bootstrap for massive data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76 (4):795–816, 2014.

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

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., Wang, S., and Mahoney, M. W. Error estimation for randomized least-squares algorithms via the bootstrap. In *International Conference on Machine Learning*, pp. 3223–3232, 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.

Magnus, J. R. and Neudecker, H. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley, 2019.

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

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.

Musco, C. and Musco, C. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems*, pp. 1396–1404, 2015.

Naumov, A., Spokoiny, V., and Ulyanov, V. Bootstrap confidence sets for spectral projectors of sample covariance. *Probability Theory and Related Fields*, 174(3-4):1091–1132, 2019.

Reynolds, R. W., Smith, T. M., Liu, C., Chelton, D. B., Casey, K. S., and Schlax, M. G. Daily high-resolution-blended analyses for sea surface temperature. *Journal of Climate*, 20(22):5473–5496, 2007.

Rokhlin, V., Szlam, A., and Tygert, M. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, 2010.

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

Stewart, G. W. and Sun, J. *Matrix Perturbation Theory*. Academic Press, 1990.

Su, W. J. 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.

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.

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.