Title
Randomized Algorithms for Scalable Machine Learning

Author
Kleiner, Ariel Jacob

Publication Date
2012

Peer reviewed|Thesis/dissertation

Randomized Algorithms for Scalable Machine Learning

by

Ariel Jacob Kleiner

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

and the Designated Emphasis

in

Communication, Computation, and Statistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Michael I. Jordan, Chair
Professor Peter J. Bickel
Professor Martin J. Wainwright

Fall 2012

Randomized Algorithms for Scalable Machine Learning

Abstract

Randomized Algorithms for Scalable Machine Learning

by

Ariel Jacob Kleiner

Doctor of Philosophy in Computer Science

Designated Emphasis in Communication, Computation, and Statistics

University of California, Berkeley

Professor Michael I. Jordan, Chair

Many existing procedures in machine learning and statistics are computationally intractable in the setting of large-scale data. As a result, the advent of rapidly increasing dataset sizes, which should be a boon yielding improved statistical performance, instead severely blunts the usefulness of a variety of existing inferential methods. In this work, we use randomness to ameliorate this lack of scalability by reducing complex, computationally difficult inferential problems to larger sets of significantly smaller and more tractable subproblems. This approach allows us to devise algorithms which are both more efficient and more amenable to use of parallel and distributed computation. We propose novel randomized algorithms for two broad classes of problems that arise in machine learning and statistics: estimator quality assessment and semidefinite programming. For the former, we present the Bag of Little Bootstraps (BLB), a procedure which incorporates features of both the bootstrap and subsampling to obtain substantial computational gains while retaining the bootstrap's accuracy and automation; we also present a novel diagnostic procedure which leverages increasing dataset sizes combined with increasingly powerful computational resources to render existing estimator quality assessment methodology more automatically usable. For semidefinite programming, we present Random Conic Pursuit, a procedure that solves semidefinite programs via repeated optimization over randomly selected two-dimensional subcones of the positive semidefinite cone. As we demonstrate via both theoretical and empirical analyses, these algorithms are scalable, readily benefit from the use of parallel and distributed computing resources, are generically applicable and easily implemented, and have favorable theoretical properties.

Contents

Acknowledgments

I have been tremendously fortunate over the years—both as a Ph.D. student and beforehand—to be surrounded by wonderful family, friends, and colleagues. Their support, encouragement, mentorship, and collaboration have enduringly illuminated my path.

First and foremost, I am deeply grateful to my mother Hanna, my father Myron, and my sister Orli. They have always supported me in every possible way, and any words that I might place upon this page would constitute only a pale facsimile of my true appreciation. I cannot imagine having a more amazing family.

I also owe my profound thanks to my Ph.D. advisor, Michael Jordan. Mike's mentorship and support throughout my years as a Ph.D. student have been truly irreplaceable. I could not have asked for a better academic role model or a better guide in my exploration of the world of machine learning and statistics.

In addition to Mike, I have had the pleasure of collaborating and interacting with a remarkable group of colleagues and friends while at UC Berkeley. I had the good fortune to work with Ameet Talwalkar, Purnamrita Sarkar, and Ali Rahimi (as well as Mike) on different facets of the research presented in this dissertation. My interaction and collaboration with them and with various members of SAIL over the years have been an integral part of my development as a researcher and practitioner of machine learning and statistics. I am also very happy to count many of my colleagues in SAIL and in UC Berkeley Computer Science more broadly as my friends. My experience at Berkeley has been unparalleled, and it is difficult to imagine a group of people having greater collective academic and technical acumen.

Finally, I am lucky to have extraordinary friends from throughout my life, who have been an enduring source of perspective, support, and fun. You all know who you are, and you have my sincere thanks.

Chapter 1

Introduction

Massive datasets are increasingly common in modern data analysis applications, and dataset sizes are growing rapidly. For example, the datasets used to develop models of the actions of internet users (e.g., to predict the probability that a user will click on a given advertisement) routinely contain billions of data points, each having millions or tens of millions of covariates [1, 43]. In natural language processing, whether engaged in language modeling or machine translation, datasets can contain billions of documents and trillions of tokens [16, 75]. Beyond such computational fields, the natural sciences are faced with a similar deluge of data. For instance, experiments and data collection efforts in the physical sciences already generate petabytes of data [5], and modern biology involves the collection and analysis of rapidly growing troves of genomic data [20, 35, 68].

These vast quantities of data which are increasingly available present both substantial opportunities and substantial challenges for machine learning and statistics. Indeed, more data holds the promise of permitting more accurate, as well as more fine-grained, estimation and inference. However, realizing this potential requires the ability to efficiently store, process, and analyze large datasets, which often exceed the storage and processing capabilities of individual processors or compute nodes. With the emergence of multicore and cloud computing and the development of software systems—such as Google's MapReduce [26], Hadoop MapReduce [41], and the Spark cluster computing system [82]—designed to permit robustly leveraging these parallel and distributed architectures, we now have available computing infrastructure which is well suited to storing and processing large datasets. Nonetheless, applying sophisticated data analysis techniques to large datasets often remains challenging, as many existing inferential procedures require computing time (or space) which scales quite adversely as the number of available data points or the data dimensionality increase; furthermore, existing inferential methods are frequently not readily able to utilize parallel and distributed computing resources to achieve scalability.

As a result, procedures in machine learning and statistics have increasingly been developed with an eye toward computational efficiency and scalability. For instance, techniques which repose upon certain types of convex optimization problems (in particular, empirical risk minimizers including logistic regression and Support Vector Machines) have prompted

and benefited from advances in the efficiency and scalability of optimization algorithms. The realization that stochastic gradient descent, despite not readily providing high-precision solutions to optimization problems, yields substantial efficiency gains in statistical settings—without sacrificing statistical performance—was an important step [14]. Though stochastic gradient descent is not straightforwardly parallelizable, initially limiting its efficiency gains to the serial single-processor setting, subsequent efforts have successfully yielded variants well-suited to multicore and distributed computing architectures [62, 83]. Such work on optimization in the context of statistical learning in fact benefits from and is closely related to a larger body of work on scalable, parallel and distributed optimization algorithms [7, 29]. Beyond procedures based on convex optimization, estimation via the EM algorithm has witnessed analogous developments of (serial) online and large-scale distributed variants enabling more efficient and scalable maximum likelihood estimation in probabilistic models with latent variables [53, 79]. Distributed computation in these cases can be achieved via the map-reduce paradigm, which has been recognized as providing easily accessible (though not necessarily very optimized) data parallelism for the EM algorithm and various other basic learning algorithms that can be seen as optimizing sums of functions evaluated at individual data points [21]. Efficient and scalable inference in certain classes of probabilistic models has also recently received a good deal of attention. The latent Dirichlet allocation (LDA) model [13] has been a particular focus of work in this vein, which has yielded online inferential techniques as well as inferential methods that are well-suited to implementation on multicore or large-scale cluster computing architectures [4, 48, 61, 81]. Similar techniques for achieving scalable inference via data parallelism have also begun to emerge for other probabilistic models, such as the Indian Buffet Process [28]. Work having a slightly different focus within the realm of probabilistic modeling has yielded efficiently parallelizable approximate inference methods for graphical models having large numbers of nodes [38, 39, 37]; these algorithms have been accompanied by the development of a computational framework for the effective parallelization of learning procedures defined on large graphs [56, 57].

Despite this burgeoning body of work, the development of methodology for large-scale estimation and inference remains far from complete. Only specific classes of problems or models are addressed by recently developed techniques, such as those discussed above, and even in these cases, new and improved techniques continue to emerge. Importantly, a number of important classes of problems in machine learning and statistics have not been substantially addressed from the standpoint of computational efficiency and scalability. Thus, in this dissertation, we develop new methods which advance the state of the art in computational efficiency and scalability for two important problem classes: estimator quality assessment and semidefinite programming. In both cases, we use randomness to reduce complex, computationally difficult inferential problems to larger sets of significantly smaller and more tractable subproblems. This approach yields both serial efficiency gains and the ability to readily utilize parallel and distributed computing resources to achieve scalability, without sacrificing statistical performance. The resulting algorithms are furthermore generically applicable and easily implemented. In the case of estimator quality assessment, we also develop a novel diagnostic procedure which leverages increasing dataset sizes combined with increasingly

powerful computational resources to render existing estimator quality assessment methodology more automatically usable.

Chapter 2 addresses computational efficiency and scalability for the core inferential problem of estimator quality assessment. Although the bootstrap [30] provides a simple and powerful means of assessing the quality of estimators, in settings involving large datasets—which are increasingly prevalent—the computation of bootstrap-based quantities can be prohibitively demanding computationally. While variants such as subsampling [67] and the m out of n bootstrap [9] can be used in principle to reduce the cost of bootstrap computations, we find that these methods are generally not robust to specification of hyperparameters (such as the number of subsampled data points), and they often require use of more prior information (such as rates of convergence of estimators) than the bootstrap. As an alternative, we introduce the Bag of Little Bootstraps (BLB), a new procedure which incorporates features of both the bootstrap and subsampling to yield a robust, computationally efficient means of assessing the quality of estimators. BLB is well suited to modern parallel and distributed computing architectures and furthermore retains the generic applicability and statistical efficiency of the bootstrap. We demonstrate BLB's favorable statistical performance via a theoretical analysis elucidating the procedure's properties, as well as a simulation study comparing BLB to the bootstrap, the m out of n bootstrap, and subsampling. In addition, we present results from a large-scale distributed implementation of BLB demonstrating its computational superiority on massive data, a method for adaptively selecting BLB's hyperparameters, an empirical study applying BLB to several real datasets, and an extension of BLB to time series data.

Remaining within the context of estimator quality assessment, Chapter 3 introduces a general performance diagnostic for the bootstrap which improves its level of automation by leveraging the availability of increasingly large datasets coupled with increasingly powerful computing resources. Indeed, as datasets become larger, more complex, and more available to diverse groups of analysts, it would be quite useful to be able to automatically and generically assess the quality of estimates, much as we are able to automatically train and evaluate predictive models such as classifiers. However, despite the fundamental importance of estimator quality assessment in data analysis, this task has eluded highly automatic solutions. While the bootstrap provides perhaps the most promising step in this direction, its level of automation is limited by the difficulty of evaluating its finite sample performance and even its asymptotic consistency. Thus, we present a general diagnostic procedure which directly and automatically evaluates the accuracy of the bootstrap's outputs, determining whether or not the bootstrap is performing satisfactorily when applied to a given dataset and estimator. We show via an extensive empirical evaluation on a variety of estimators and simulated and real datasets that our proposed diagnostic is effective.

Chapter 4 shifts to the problem of semidefinite programming, which underlies a variety of procedures in machine learning and statistics; standard generic methods for solving semidefinite programs (SDPs) generally scale quite adversely in the problem dimensionality. We present a novel algorithm, Random Conic Pursuit, that solves SDPs via repeated optimization over randomly selected two-dimensional subcones of the positive semidefinite cone.

This scheme is simple, easily implemented, applicable to very general SDPs, scalable, and theoretically interesting. Its advantages are realized at the expense of an ability to readily compute highly exact solutions, though useful approximate solutions are easily obtained. This property renders Random Conic Pursuit of particular interest for machine learning and statistical applications, in which the relevant SDPs are generally based upon random data and so exact minima are often not a priority. Indeed, we present empirical results to this effect for various SDPs encountered in machine learning and statistics; we also provide an analysis that yields insight into the theoretical properties and convergence of the algorithm.

Finally, Chapter 5 concludes with a discussion of open questions and potential avenues of future work.

Chapter 2

A Scalable Bootstrap for Massive Data

2.1 Introduction

The development of the bootstrap and related resampling-based methods in the 1960s and 1970s heralded an era in statistics in which inference and computation became increasingly intertwined [30, 27]. By exploiting the basic capabilities of the classical von Neumann computer to simulate and iterate, the bootstrap made it possible to use computers not only to compute estimates but also to assess the quality of estimators, yielding results that are quite generally consistent [8, 36, 77] and often more accurate than those based upon asymptotic approximation [44]. Moreover, the bootstrap aligned statistics to computing technology, such that advances in speed and storage capacity of computers could immediately allow statistical methods to scale to larger datasets.

Two recent trends are worthy of attention in this regard. First, the growth in size of datasets is accelerating, with "massive" datasets becoming increasingly prevalent. Second, computational resources are shifting toward parallel and distributed architectures, with multicore and cloud computing platforms providing access to hundreds or thousands of processors. The second trend is seen as a mitigating factor with respect to the first, in that parallel and distributed architectures present new capabilities for storage and manipulation of data. However, from an inferential point of view, it is not yet clear how statistical methodology will transport to a world involving massive data on parallel and distributed computing platforms.

While massive data bring many statistical issues to the fore, including issues in exploratory data analysis and data visualization, there remains the core inferential need to assess the quality of estimators. Indeed, the uncertainty and biases in estimates based on large data can remain quite significant, as large datasets are often high dimensional, are frequently used to fit complex models with large numbers of parameters, and can have many potential sources of bias. Furthermore, even if sufficient data are available to allow highly

accurate estimation, the ability to efficiently assess estimator quality remains essential to allow efficient use of available resources by processing only as much data as is necessary to achieve a desired accuracy or confidence.

The bootstrap brings to bear various desirable features in the massive data setting, notably its relatively automatic nature and its applicability to a wide variety of inferential problems. It can be used to assess bias, to quantify the uncertainty in an estimate (e.g., via a standard error or a confidence interval), or to assess risk. However, these virtues are realized at the expense of a substantial computational burden. Bootstrap-based quantities typically must be computed via a form of Monte Carlo approximation in which the estimator in question is repeatedly applied to resamples of the entire original observed dataset.

Because these resamples have size on the order of that of the original data, with approximately 63% of data points appearing at least once in each resample, the usefulness of the bootstrap is severely blunted by the large datasets increasingly encountered in practice. In the massive data setting, computation of even a single point estimate on the full dataset can be quite computationally demanding, and so repeated computation of an estimator on comparably sized resamples can be prohibitively costly. To mitigate this problem, one might naturally attempt to exploit the modern trend toward parallel and distributed computing. Indeed, at first glance, the bootstrap would seem ideally suited to straightforwardly leveraging parallel and distributed computing architectures: one might imagine using different processors or compute nodes to process different bootstrap resamples independently in parallel. However, the large size of bootstrap resamples in the massive data setting renders this approach problematic, as the cost of transferring data to independent processors or compute nodes can be overly high, as is the cost of operating on even a single resample using an independent set of computing resources.

While the literature does contain some discussion of techniques for improving the computational efficiency of the bootstrap, that work is largely devoted to reducing the number of resamples required [31, 33]. These techniques in general introduce significant additional complexity of implementation and do not eliminate the crippling need for repeated computation of the estimator on resamples having size comparable to that of the original dataset.

Another landmark in the development of simulation-based inference is subsampling [67] and the closely related m out of n bootstrap [9]. These methods (which were introduced to achieve statistical consistency in edge cases in which the bootstrap fails) initially appear to remedy the bootstrap's key computational shortcoming, as they only require repeated computation of the estimator under consideration on resamples (or subsamples) that can be significantly smaller than the original dataset. However, these procedures also have drawbacks. As we show in our simulation study, their success is sensitive to the choice of resample (or subsample) size (i.e., m in the m out of n bootstrap). Additionally, because the variability of an estimator on a subsample differs from its variability on the full dataset, these procedures must perform a rescaling of their output, and this rescaling requires knowledge and explicit use of the convergence rate of the estimator in question; these methods are thus less automatic and easily deployable than the bootstrap. While schemes have been proposed for data-driven selection of an optimal resample size [11], they require significantly

greater computation which would eliminate any computational gains. Also, there has been work on the m out of n bootstrap that has sought to reduce computational costs using two different values of m in conjunction with extrapolation [12, 10]. However, these approaches explicitly utilize series expansions of the estimator's sampling distribution and hence are less automatically usable; they also require execution of the m out of n bootstrap for multiple values of m.

Motivated by the need for an automatic, accurate means of assessing estimator quality that is scalable to large datasets, we introduce a new procedure, the Bag of Little Bootstraps (BLB), which functions by combining the results of bootstrapping multiple small subsets of a larger original dataset. Instead of applying the estimator directly to each small subset, as in the m out of n bootstrap and subsampling, BLB applies the bootstrap to each small subset; in the resampling process of each individual bootstrap run, weighted resamples are formed such that the effect is that of sampling the small subset n times with replacement, but the computational cost is that associated with the size of the small subset. This has the effect that, despite operating only on subsets of the original dataset, BLB does not require analytical rescaling of its output. Overall, BLB has a significantly more favorable computational profile than the bootstrap, as it only requires repeated computation of the estimator under consideration on quantities of data that can be much smaller than the original dataset. As a result, BLB is well suited to implementation on modern distributed and parallel computing architectures which are often used to process large datasets. Also, our procedure maintains the bootstrap's generic applicability, favorable statistical properties (i.e., consistency and higher-order correctness), and simplicity of implementation. Finally, as we show in experiments, BLB is consistently more robust than alternatives such as the m out of n bootstrap and subsampling.

The remainder of our presentation is organized as follows. In Section 2.2, we formalize our statistical setting and notation, present BLB in detail, and discuss the procedure's computational characteristics. Subsequently, in Section 2.3, we elucidate BLB's statistical properties via a theoretical analysis (Section 2.3.1) showing that BLB shares the bootstrap's consistency and higher-order correctness, as well as a simulation study (Section 2.3.2) which compares BLB to the bootstrap, the m out of n bootstrap, and subsampling. Section 2.4 discusses a large-scale implementation of BLB on a distributed computing system and presents results illustrating the procedure's superior computational performance in the massive data setting. We present a method for adaptively selecting BLB's hyperparameters in Section 2.5. Finally, we apply BLB (as well as the bootstrap and the m out of n bootstrap, for comparison) to several real datasets in Section 2.6, and we present an extension of BLB to time series data in Section 2.7.

2.2 Bag of Little Bootstraps (BLB)

2.2.1 Setting and Notation

We assume that we observe a sample $X_1, \ldots, X_n \in \mathcal{X}$ drawn i.i.d. from some (unknown) underlying distribution $P \in \mathcal{P}$; we denote by $\mathbb{P}_n = n^{-1} \sum_{i=1}^{n} \delta_{X_i}$ the corresponding empirical distribution. Based only on this observed data, we compute an estimate $\hat{\theta}_n \in \Theta$ of some (unknown) population value $\theta \in \Theta$ associated with P. For example, $\hat{\theta}_n$ might estimate a measure of correlation, the parameters of a regressor, or the prediction accuracy of a trained classification model. When we wish to explicitly indicate the data used to compute an estimate, we shall write $\hat{\theta}(\mathbb{P}_n)$. Noting that $\hat{\theta}_n$ is a random quantity because it is based on n random observations, we define $Q_n(P) \in \mathcal{Q}$ as the true underlying distribution of $\hat{\theta}_n$, which is determined by both P and the form of the estimator. Our end goal is the computation of an estimator quality assessment $\xi(Q_n(P), P) : \mathcal{Q} \times \mathcal{P} \to \Xi$, for Ξ a vector space; to lighten notation, we shall interchangeably write $\xi(Q_n(P))$ in place of $\xi(Q_n(P), P)$. For instance, ξ might compute a quantile, a confidence region, a standard error, or a bias. In practice, we do not have direct knowledge of P or $Q_n(P)$, and so we must estimate $\xi(Q_n(P))$ itself based only on the observed data and knowledge of the form of the estimator under consideration.

Note that we allow ξ to depend directly on P in addition to $Q_n(P)$ because ξ might operate on the distribution of a centered and normalized version of $\hat{\theta}_n$. For example, if ξ computes a confidence region, it might manipulate the distribution of the statistic $\sqrt{n}(\hat{\theta}_n - \theta)$, which is determined by both $Q_n(P)$ and θ; because θ cannot in general be obtained directly from $Q_n(P)$, a direct dependence on P is required in this case. Nonetheless, given knowledge of $Q_n(P)$, any direct dependence of ξ on P generally has a simple form, often only involving the parameter θ. Additionally, rather than restricting $Q_n(P)$ to be the distribution of $\hat{\theta}_n$, we could instead allow it to be the distribution of a more general statistic, such as $(\hat{\theta}_n, \hat{\sigma}_n)$, where $\hat{\sigma}_n$ is an estimate of the standard deviation of $\hat{\theta}_n$ (e.g., this would apply when constructing confidence intervals based on the distribution of the studentized statistic $(\hat{\theta}_n - \theta)/\hat{\sigma}_n$). Our subsequent development generalizes straightforwardly to this setting, but to simplify the exposition, we will largely assume that $Q_n(P)$ is the distribution of $\hat{\theta}_n$.

Under our notation, the bootstrap simply computes the data-driven plugin approximation $\xi(Q_n(P)) \approx \xi(Q_n(\mathbb{P}_n))$. Although $\xi(Q_n(\mathbb{P}_n))$ cannot be computed exactly in most cases, it is generally amenable to straightforward Monte Carlo approximation via the following algorithm [33]: repeatedly resample n points i.i.d. from $\mathbb{P}_n$, compute the estimate on each resample, form the empirical distribution $\mathbb{Q}_n^*$ of the computed estimates, and approximate $\xi(Q_n(P)) \approx \xi(\mathbb{Q}_n^*)$.

Similarly, using our notation, the m out of n bootstrap (and subsampling) functions as follows, for $m < n$ [9, 67]: repeatedly resample m points i.i.d. from $\mathbb{P}_n$ (subsample m points without replacement from $X_1, \ldots, X_n$), compute the estimate on each resample (subsample), form the empirical distribution $\mathbb{Q}_m^*$ of the computed estimates, approximate $\xi(Q_m(P)) \approx \xi(\mathbb{Q}_m^*)$, and apply an analytical correction to in turn approximate $\xi(Q_n(P))$. This final analytical correction uses prior knowledge of the convergence rate of $\hat{\theta}_n$ as n

increases and is necessary because each value of the estimate is computed based on only m rather than n points.

We use $\mathbf{1}_d$ to denote the d-dimensional vector of ones, and we let I_d denote the $d \times d$ identity matrix.

2.2.2 Bag of Little Bootstraps

The Bag of Little Bootstraps (BLB) functions by averaging the results of bootstrapping multiple small subsets of $X_1, \ldots, X_n$. More formally, given a subset size $b < n$, BLB samples s subsets of size b from the original n data points, uniformly at random (one can also impose the constraint that the subsets be disjoint). Let $\mathcal{I}_1, \ldots, \mathcal{I}_s \subset \{1, \ldots, n\}$ be the corresponding index multisets (note that $|\mathcal{I}_j| = b, \forall j$), and let $\mathbb{P}_{n,b}^{(j)} = b^{-1} \sum_{i \in \mathcal{I}_j} \delta_{X_i}$ be the empirical distribution corresponding to subset j. BLB's estimate of $\xi(Q_n(P))$ is then given by

$$s^{-1} \sum_{j=1}^{s} \xi(Q_n(\mathbb{P}_{n,b}^{(j)})). \tag{2.1}$$

Although the terms $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ in (2.1) cannot be computed analytically in general, they can be computed numerically via straightforward Monte Carlo approximation in the manner of the bootstrap: for each term j, repeatedly resample n points i.i.d. from $\mathbb{P}_{n,b}^{(j)}$, compute the estimate on each resample, form the empirical distribution $\mathbb{Q}_{n,j}^{*}$ of the computed estimates, and approximate $\xi(Q_n(\mathbb{P}_{n,b}^{(j)})) \approx \xi(\mathbb{Q}_{n,j}^{*})$.

Now, to realize the substantial computational benefits afforded by BLB, we utilize the following crucial fact: each BLB resample, despite having nominal size n, contains at most b distinct data points. In particular, to generate each resample, it suffices to draw a vector of counts from an n-trial uniform multinomial distribution over b objects. We can then represent each resample by simply maintaining the at most b distinct points present within it, accompanied by corresponding sampled counts (i.e., each resample requires only storage space in $O(b)$). In turn, if the estimator can work directly with this weighted data representation, then the computational requirements of the estimator—with respect to both time and storage space—scale only in b, rather than n. Fortunately, this property does indeed hold for many if not most commonly used estimators, such as general M-estimators. The resulting BLB algorithm, including Monte Carlo resampling, is shown in Algorithm 1.

Thus, BLB only requires repeated computation on small subsets of the original dataset and avoids the bootstrap's problematic need for repeated computation of the estimate on resamples having size comparable to that of the original dataset. A simple and standard calculation [33] shows that each bootstrap resample contains approximately $0.632n$ distinct points, which is large if n is large. In contrast, as discussed above, each BLB resample contains at most b distinct points, and b can be chosen to be much smaller than n or $0.632n$. For example, we might take $b = n^{\gamma}$ where $\gamma \in [0.5, 1]$. More concretely, if $n = 1,000,000$, then each bootstrap resample would contain approximately $632,000$ distinct points, whereas with

Algorithm 1: Bag of Little Bootstraps (BLB)

Input: Data $X_1, \ldots, X_n$ b: subset size
$\hat{\theta}$: estimator of interest s: number of sampled subsets
ξ: estimator quality assessment r: number of Monte Carlo iterations
Output: An estimate of $\xi(Q_n(P))$

for $j \leftarrow 1$ **to** s **do**
  `// Subsample the data`
  Randomly sample a set $\mathcal{I} = \{i_1, \ldots, i_b\}$ of b indices from $\{1, \ldots, n\}$ without replacement
  [or, choose $\mathcal{I}$ to be a disjoint subset of size b from a predefined random partition of $\{1, \ldots, n\}$]
  `// Approximate` $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$
  for $k \leftarrow 1$ **to** r **do**
    Sample $(n_1, \ldots, n_b) \sim \text{Multinomial}(n, \mathbf{1}_b/b)$
    $\mathbb{P}_{n,k}^* \leftarrow n^{-1} \sum_{a=1}^{b} n_a \delta_{X_{i_a}}$
    $\hat{\theta}_{n,k}^* \leftarrow \hat{\theta}(\mathbb{P}_{n,k}^*)$
  end
  $\mathbb{Q}_{n,j}^* \leftarrow r^{-1} \sum_{k=1}^{r} \delta_{\hat{\theta}_{n,k}^*}$
  $\xi_{n,j}^* \leftarrow \xi(\mathbb{Q}_{n,j}^*)$
end
`// Average values of` $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ `computed for different data subsets`
return $s^{-1} \sum_{j=1}^{s} \xi_{n,j}^*$

$b = n^{0.6}$ each BLB subsample and resample would contain at most $3,981$ distinct points. If each data point occupies 1 MB of storage space, then the original dataset would occupy 1 TB, a bootstrap resample would occupy approximately 632 GB, and each BLB subsample or resample would occupy at most 4 GB. As a result, the cost of computing the estimate on each BLB resample is generally substantially lower than the cost of computing the estimate on each bootstrap resample, or on the full dataset. Furthermore, as we show in our simulation study and scalability experiments below, BLB typically requires less total computation (across multiple data subsets and resamples) than the bootstrap to reach comparably high accuracy; fairly modest values of s and r suffice.

Due to its much smaller subsample and resample sizes, BLB is also significantly more amenable than the bootstrap to distribution of different subsamples and resamples and their associated computations to independent compute nodes; therefore, BLB allows for simple distributed and parallel implementations, enabling additional large computational gains. In the large data setting, computing a single full-data point estimate often requires simultaneous distributed computation across multiple compute nodes, among which the observed dataset is partitioned. Given the large size of each bootstrap resample, computing the estimate on

even a single such resample in turn also requires the use of a comparably large cluster of compute nodes; the bootstrap requires repetition of this computation for multiple resamples. Each computation of the estimate is thus quite costly, and the aggregate computational costs of this repeated distributed computation are quite high (indeed, the computation for each bootstrap resample requires use of an entire cluster of compute nodes and incurs the associated overhead).

In contrast, BLB straightforwardly permits computation on multiple (or even all) subsamples and resamples simultaneously in parallel: because BLB subsamples and resamples can be significantly smaller than the original dataset, they can be transferred to, stored by, and processed on individual (or very small sets of) compute nodes. For example, we could naturally leverage modern hierarchical distributed architectures by distributing subsamples to different compute nodes and subsequently using intra-node parallelism to compute across different resamples generated from the same subsample. Thus, relative to the bootstrap, BLB both decreases the total computational cost of assessing estimator quality and allows more natural use of parallel and distributed computational resources. Moreover, even if only a single compute node is available, BLB allows the following somewhat counterintuitive possibility: even if it is prohibitive to actually compute a point estimate for the full observed data using a single compute node (because the full dataset is large), it may still be possible to efficiently assess such a point estimate's quality using only a single compute node by processing one subsample (and the associated resamples) at a time.

Returning to equation (2.1), unlike the plugin approximation $\xi(Q_n(\mathbb{P}_n))$ used by the bootstrap, the plugin approximations $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ used by BLB are based on empirical distributions $\mathbb{P}_{n,b}^{(j)}$ which are more compact and hence, as we have seen, less computationally demanding than the full empirical distribution $\mathbb{P}_n$. However, each $\mathbb{P}_{n,b}^{(j)}$ is inferior to $\mathbb{P}_n$ as an approximation to the true underlying distribution P, and so BLB averages across multiple different realizations of $\mathbb{P}_{n,b}^{(j)}$ to improve the quality of the final result. This procedure yields significant computational benefits over the bootstrap (as discussed above and demonstrated empirically in Section 2.4), while having the same generic applicability and favorable statistical properties as the bootstrap (as shown in the next section), in addition to being more robust than the m out of n bootstrap and subsampling to the choice of subset size (see our simulation study below).

2.3 Statistical Performance

2.3.1 Consistency and Higher-Order Correctness

We now show that BLB has statistical properties—in particular, asymptotic consistency and higher-order correctness—which are identical to those of the bootstrap, under the same conditions that have been used in prior analysis of the bootstrap. Note that if $\hat{\theta}_n$ is consistent (i.e., approaches θ in probability) as $n \to \infty$, then it has a degenerate limiting distribution. Thus, in studying the asymptotics of the bootstrap and related procedures, it is typical

to assume that ξ manipulates the distribution of a centered and normalized version of $\hat{\theta}_n$ (though this distribution is still determined by $Q_n(P)$ and P). Additionally, as in standard analyses of the bootstrap, we do not explicitly account here for error introduced by use of Monte Carlo approximation to compute the individual plugin approximations $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$.

The following theorem states that (under standard assumptions) as $b, n \to \infty$, the estimates $s^{-1}\sum_{j=1}^{s}\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ returned by BLB approach the population value $\xi(Q_n(P))$ in probability. Interestingly, the only assumption about b required for this result is that $b \to \infty$, though in practice we would generally take b to be a slowly growing function of n.

Theorem 1. *Suppose that $\hat{\theta}_n = \phi(\mathbb{P}_n)$ and $\theta = \phi(P)$, where ϕ is Hadamard differentiable at P tangentially to some subspace, with P, $\mathbb{P}_n$, and $\mathbb{P}_{n,b}^{(j)}$ viewed as maps from some Donsker class $\mathcal{F}$ to $\mathbb{R}$ such that $\mathcal{F}_\delta$ is measurable for every $\delta > 0$, where $\mathcal{F}_\delta = \{f - g : f, g \in \mathcal{F}, \rho_P(f - g) < \delta\}$ and $\rho_P(f) = (P(f - Pf)^2)^{1/2}$. Additionally, assume that $\xi(Q_n(P))$ is a function of the distribution of $\sqrt{n}(\phi(\mathbb{P}_n) - \phi(P))$ which is continuous in the space of such distributions with respect to a metric that metrizes weak convergence. Then,*

$$s^{-1}\sum_{j=1}^{s}\xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P)) \xrightarrow{P} 0$$

as $n \to \infty$, for any sequence $b \to \infty$ and for any fixed s.

See the appendix for a proof of this theorem, as well as for proofs of all other results in this section. Note that the assumptions of Theorem 1 are standard in analysis of the bootstrap and in fact hold in many practically interesting cases. For example, M-estimators are generally Hadamard differentiable (under some regularity conditions) [76, 77], and the assumptions on ξ are satisfied if, for example, ξ computes a cdf value. Theorem 1 can also be generalized to hold for sequences $s \to \infty$ and more general forms of ξ, but such generalization appears to require stronger assumptions, such as uniform integrability of the $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$; the need for stronger assumptions in order to obtain more general consistency results has also been noted in prior work on the bootstrap (e.g., see [42]).

Moving beyond analysis of the asymptotic consistency of BLB, we now characterize its higher-order correctness (i.e., the rate of convergence of its output to $\xi(Q_n(P))$). A great deal of prior work has been devoted to showing that the bootstrap is higher-order correct in many cases (e.g., see the seminal book by Hall [44]), meaning that it converges to the true value $\xi(Q_n(P))$ at a rate of $O_P(1/n)$ or faster. In contrast, methods based on analytical asymptotic approximation are generally correct only at order $O_P(1/\sqrt{n})$. The bootstrap converges more quickly due to its ability to utilize the full empirical distribution of the observed data (rather than, for example, only low-order sample moments), which allows it to better capture finite-sample deviations of the distribution of the quantity of interest from its asymptotic limiting distribution.

As shown by the following theorem, BLB shares the same degree of higher-order correctness as the bootstrap, assuming that s and b are chosen to be sufficiently large. Importantly,

sufficiently large values of b here can still be significantly smaller than n, with $b/n \to 0$ as $n \to \infty$. Following prior analyses of the bootstrap, we now make the standard assumption that ξ can be represented via an asymptotic series expansion in powers of $1/\sqrt{n}$. In fact, prior work provides such expansions in a variety of settings. When ξ computes a cdf value, these expansions are termed Edgeworth expansions; if ξ computes a quantile, then the relevant expansions are Cornish-Fisher expansions. See [44] for a full development of such expansions both in generality as well as for specific forms of the estimator, including smooth functions of mean-like statistics and curve estimators.

Theorem 2. *Suppose that $\xi(Q_n(P))$ admits an expansion as an asymptotic series*

$$\xi(Q_n(P)) = z + \frac{p_1}{\sqrt{n}} + \cdots + \frac{p_k}{n^{k/2}} + o\left(\frac{1}{n^{k/2}}\right), \tag{2.2}$$

where z is a constant independent of P and the p_k are polynomials in the moments of P. Additionally, assume that the empirical version of $\xi(Q_n(P))$ for any j admits a similar expansion

$$\xi(Q_n(\mathbb{P}_{n,b}^{(j)})) = z + \frac{\hat{p}_1^{(j)}}{\sqrt{n}} + \cdots + \frac{\hat{p}_k^{(j)}}{n^{k/2}} + o_P\left(\frac{1}{n^{k/2}}\right), \tag{2.3}$$

where z is as defined above and the $\hat{p}_k^{(j)}$ are polynomials in the moments of $\mathbb{P}_{n,b}^{(j)}$ obtained by replacing the moments of P in the p_k with those of $\mathbb{P}_{n,b}^{(j)}$. Then, assuming that $b \leq n$ and $E(\hat{p}_k^{(1)})^2 < \infty$ for $k \in \{1, 2\}$,

$$\left| s^{-1} \sum_{j=1}^{s} \xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P)) \right| = O_P\left(\frac{\sqrt{\mathrm{Var}(\hat{p}_k^{(1)} - p_k | \mathbb{P}_n)}}{\sqrt{ns}} \right) + O_P\left(\frac{1}{n}\right) + O\left(\frac{1}{b\sqrt{n}}\right). \tag{2.4}$$

Therefore, taking $s = \Omega(n \,\mathrm{Var}(\hat{p}_k^{(1)} - p_k | \mathbb{P}_n))$ and $b = \Omega(\sqrt{n})$ yields

$$\left| s^{-1} \sum_{j=1}^{s} \xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P)) \right| = O_P\left(\frac{1}{n}\right),$$

in which case BLB enjoys the same level of higher-order correctness as the bootstrap.

Note that it is natural to assume above that $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ can be expanded in powers of $1/\sqrt{n}$, rather than $1/\sqrt{b}$, because $Q_n(\mathbb{P}_{n,b}^{(j)})$ is the distribution of the estimate computed on n points sampled from $\mathbb{P}_{n,b}^{(j)}$. The fact that only b points are represented in $\mathbb{P}_{n,b}^{(j)}$ enters via the $\hat{p}_k^{(j)}$, which are polynomials in the sample moments of those b points.

Theorem 2 indicates that, like the bootstrap, BLB can converge at rate $O_P(1/n)$ (assuming that s and b grow at a sufficient rate). Additionally, because $\mathrm{Var}(\hat{p}_k^{(1)} - p_k | \mathbb{P}_n)$ is decreasing in probability as b and n increase, s can grow significantly more slowly than n

(indeed, unconditionally, $\hat{p}_k^{(j)} - p_k = O_P(1/\sqrt{b})$). While $\mathrm{Var}(\hat{p}_k^{(1)} - p_k | \mathbb{P}_n)$ can in principle be computed given an observed dataset, as it depends only on $\mathbb{P}_n$ and the form of the estimator under consideration, we can also obtain a general upper bound (in probability) on the rate of decrease of this conditional variance:

Remark 1. *Assuming that* $E(\hat{p}_k^{(1)})^4 < \infty$, $\mathrm{Var}(\hat{p}_k^{(1)} - p_k | \mathbb{P}_n) = O_P(1/\sqrt{n}) + O(1/b)$.

The following result, which applies to the alternative variant of BLB that constrains the s randomly sampled subsets to be disjoint, also highlights the fact that s can grow substantially more slowly than n:

Theorem 3. *Under the assumptions of Theorem 2, and assuming that BLB uses disjoint random subsets of the observed data (rather than simple random subsamples), we have*

$$\left| s^{-1} \sum_{j=1}^{s} \xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P)) \right| = O_P\left(\frac{1}{\sqrt{nbs}}\right) + O\left(\frac{1}{b\sqrt{n}}\right). \tag{2.5}$$

Therefore, if $s \sim (n/b)$ *and* $b = \Omega(\sqrt{n})$, *then*

$$\left| s^{-1} \sum_{j=1}^{s} \xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P)) \right| = O_P\left(\frac{1}{n}\right),$$

in which case BLB enjoys the same level of higher-order correctness as the bootstrap.

Finally, while the assumptions of the two preceding theorems generally require that ξ studentizes the estimator under consideration (which involves dividing by an estimate of standard error), similar results hold even if the estimator is not studentized. In particular, not studentizing slows the convergence rate of both the bootstrap and BLB by the same factor, generally causing the loss of a factor of $O_P(1/\sqrt{n})$ [76].

2.3.2 Simulation Study

We investigate empirically the statistical performance characteristics of BLB and compare to the statistical performance of existing methods via experiments on simulated data. Use of simulated data is necessary here because it allows knowledge of P, $Q_n(P)$, and hence $\xi(Q_n(P))$; this ground truth is required for evaluation of statistical correctness. For different datasets and estimation tasks, we study the convergence properties of BLB as well as the bootstrap, the m out of n bootstrap, and subsampling.

We consider two different settings: regression and classification. For both settings, the data have the form $X_i = (\tilde{X}_i, Y_i) \sim P$, i.i.d. for $i = 1, \ldots, n$, where $\tilde{X}_i \in \mathbb{R}^d$; $Y_i \in \mathbb{R}$ for regression, whereas $Y_i \in \{0, 1\}$ for classification. In each case, $\hat{\theta}_n$ estimates a parameter vector in $\mathbb{R}^d$ for a linear or generalized linear model of the mapping between $\tilde{X}_i$ and Y_i. We define ξ as a procedure that computes a set of marginal 95% confidence intervals, one

for each element of the estimated parameter vector. In particular, given an estimator's sampling distribution Q (or an approximation thereof), ξ computes the boundaries of the relevant confidence intervals as the 2.5th and 97.5th percentiles of the marginal component-wise distributions defined by Q (averaging across ξ's simply consists of averaging these percentile estimates).

To evaluate the various quality assessment procedures on a given estimation task and true underlying data distribution P, we first compute the ground truth $\xi(Q_n(P))$ by generating $2,000$ realizations of datasets of size n from P, computing $\hat{\theta}_n$ on each, and using this collection of $\hat{\theta}_n$'s to form a high-fidelity approximation to $Q_n(P)$. Then, for an independent dataset realization of size n from the true underlying distribution, we run each quality assessment procedure (without parallelization) until it converges and record the estimate of $\xi(Q_n(P))$ produced after each iteration (e.g., after each bootstrap resample or BLB subsample is processed), as well as the cumulative processing time required to produce that estimate. Every such estimate is evaluated based on the average (across dimensions) relative deviation of its component-wise confidence intervals' widths from the corresponding true widths; given an estimated confidence interval width c and a true width c_o, the relative deviation of c from c_o is defined as $|c - c_o|/c_o$. We repeat this process on five independent dataset realizations of size n and average the resulting relative errors and corresponding processing times across these five datasets to obtain a trajectory of relative error versus time for each quality assessment procedure. The relative errors' variances are small relative to the relevant differences between their means, and so these variances are not shown in our plots. Note that we evaluate based on confidence interval widths, rather than coverage probabilities, to control the running times of our experiments: in our experimental setting, even a single run of a quality assessment procedure requires non-trivial time, and computing coverage probabilities would require a large number of such runs. All experiments in this section were implemented and executed using MATLAB on a single processor. To maintain consistency of notation, we refer to the m out of n bootstrap as the b out of n bootstrap throughout the remainder of this section. For BLB, the b out of n bootstrap, and subsampling, we consider $b = n^\gamma$ with $\gamma \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$; we use $r = 100$ in all runs of BLB.

In the regression setting, we generate each dataset from a true underlying distribution P consisting of either a linear model $Y_i = \tilde{X}_i^T \mathbf{1}_d + \epsilon_i$ or a model $Y_i = \tilde{X}_i^T \mathbf{1}_d + \tilde{X}_i^T \tilde{X}_i + \epsilon_i$ having a quadratic term, with $d = 100$ and $n = 20,000$. The $\tilde{X}_i$ and ϵ_i are drawn independently from one of the following pairs of distributions: $\tilde{X}_i \sim \text{Normal}(0, I_d)$ with $\epsilon_i \sim \text{Normal}(0, 10)$; $\tilde{X}_{i,j} \sim \text{StudentT}(3)$ i.i.d. for $j = 1, \ldots, d$ with $\epsilon_i \sim \text{Normal}(0, 10)$; or $\tilde{X}_{i,j} \sim \text{Gamma}(1 + 5(j-1)/\max(d-1, 1), 2) - 2[1 + 5(j-1)/\max(d-1, 1), 2]$ independently for $j = 1, \ldots, d$ with $\epsilon_i \sim \text{Gamma}(1, 2) - 2$. All of these distributions have $E\tilde{X}_i = E\epsilon_i = 0$, and the last $\tilde{X}_i$ distribution has non-zero skewness which varies among the dimensions. In the regression setting under both the linear and quadratic data generating distributions, our estimator $\hat{\theta}_n$ consists of a linear (in $\tilde{X}_i$) least squares regression with a small L_2 penalty on the parameter vector to encourage numerical stability (we set the weight on this penalty term to 10^{-5}). The true average (across dimensions) marginal confidence interval width for the estimated parameter vector is approximately 0.1 under the linear data generating distributions (for all

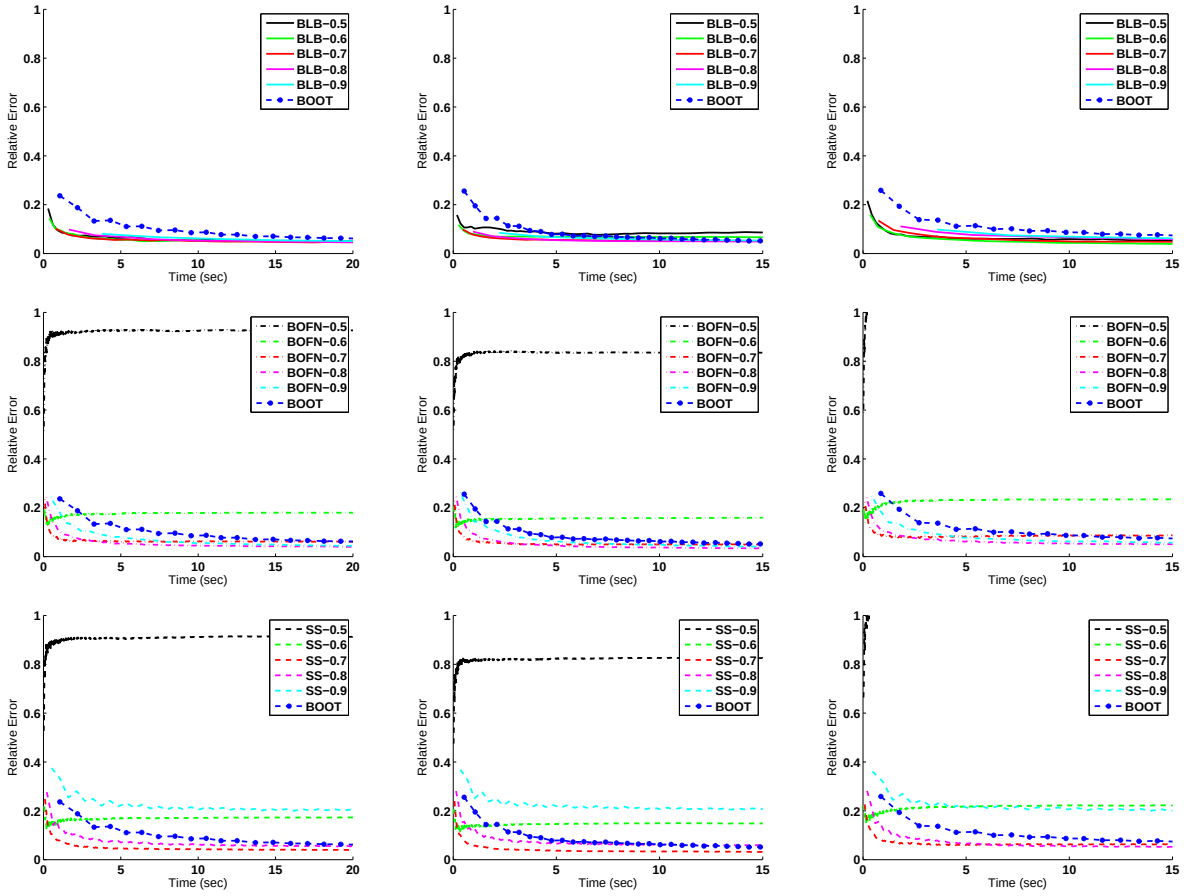


Figure 2.1: Relative error vs. processing time for regression setting. The top row shows BLB with bootstrap (BOOT), the middle row shows b out of n bootstrap (BOFN), and the bottom row shows subsampling (SS). For BLB, BOFN, and SS, $b = n^\gamma$ with the value of γ for each trajectory given in the legend. The left column shows results for linear regression with linear data generating distribution and Gamma $\tilde{X}_i$ distribution. The middle column shows results for linear regression with quadratic data generating distribution and Gamma $\tilde{X}_i$ distribution. The right column shows results for linear regression with linear data generating distribution and StudentT $\tilde{X}_i$ distribution.

$\tilde{X}_i$ distributions) and approximately 1 under the quadratic data generating distributions.

Figure 2.1 shows results for the regression setting under the linear and quadratic data generating distributions with the Gamma and StudentT $\tilde{X}_i$ distributions; similar results hold for the Normal $\tilde{X}_i$ distribution. In all cases, BLB (top row) succeeds in converging to low relative error significantly more quickly than the bootstrap, for all values of b considered. In contrast, the b out of n bootstrap (middle row) fails to converge to low relative error for smaller values of b (below $n^{0.7}$). Additionally, subsampling (bottom row) performs strictly

worse than the b out of n bootstrap, as subsampling fails to converge to low relative error for both smaller and larger values of b (e.g., for $b = n^{0.9}$). Note that fairly modest values of s suffice for convergence of BLB (recall that s values are implicit in the time axes of our plots), with s at convergence ranging from 1-2 for $b = n^{0.9}$ up to 10-14 for $b = n^{0.5}$, in the experiments shown in Figure 2.1; larger values of s are required for smaller values of b, which accords with both intuition and our theoretical analysis. Under the quadratic data generating distribution with StudentT $\tilde{X}_i$ distribution (plots not shown), none of the procedures (including the bootstrap) converge to low relative error, which is unsurprising given the StudentT(3) distribution's lack of moments beyond order two.

For the classification setting, we generate each dataset considered from either a linear model $Y_i \sim \text{Bernoulli}((1 + \exp(-\tilde{X}_i^T \mathbf{1}))^{-1})$ or a model $Y_i \sim \text{Bernoulli}((1 + \exp(-\tilde{X}_i^T \mathbf{1} - \tilde{X}_i^T \tilde{X}_i))^{-1})$ having a quadratic term, with $d = 10$. We use the three different distributions on $\tilde{X}_i$ defined in the regression setting. Our estimator, under both the linear and quadratic data generating distributions, consists of a linear (in $\tilde{X}_i$) logistic regression fit via Newton's method, again using an L_2 penalty term with weight 10^{-5} to encourage numerical stability. For this estimation task with $n = 20,000$, the true average (across dimensions) marginal confidence interval width for the estimated parameter vector is approximately 0.1 under the linear data generating distributions (for all $\tilde{X}_i$ distributions) and approximately 0.02 under the quadratic data generating distributions.

Figure 2.2 shows results for the classification setting under the linear and quadratic data generating distributions with the Gamma and StudentT $\tilde{X}_i$ distributions, and $n = 20,000$ (as in Figure 2.1); results for the Normal $\tilde{X}_i$ distribution are qualitatively similar. Here, the performance of the various procedures is more varied than in the regression setting. The case of the linear data generating distribution with Gamma $\tilde{X}_i$ distribution (left column of Figure 2.2) appears to be the most challenging. In this setting, BLB converges to relative error comparable to that of the bootstrap for $b > n^{0.6}$, while converging to higher relative errors for the smallest values of b considered. For the larger values of b, which are still significantly smaller than n, we again converge to low relative error faster than the bootstrap. We are also once again more robust than the b out of n bootstrap, which fails to converge to low relative error for $b \leq n^{0.7}$. In fact, even for $b \leq n^{0.6}$, BLB's performance is superior to that of the b out of n bootstrap. Qualitatively similar results hold for the other data generating distributions, but with BLB and the b out of n bootstrap both performing better relative to the bootstrap. In the experiments shown in Figure 2.2, the values of s (which are implicit in the time axes of our plots) required for convergence of BLB range from 1-2 for $b = n^{0.9}$ up to 10-20 for $b \leq n^{0.6}$ (for cases in which BLB converges to low relative error). As in the regression setting, subsampling (plots not shown) has performance strictly worse than that of the b out of n bootstrap in all cases.

To further examine the cases in which BLB (when using small values of b) does not converge to relative error comparable to that of the bootstrap, we explore how the various procedures' relative errors vary with n. In particular, for different values of n (and b), we run each procedure as described above and report the relative error that it achieves after it converges (i.e., after it has processed sufficiently many subsets, in the case of BLB, or

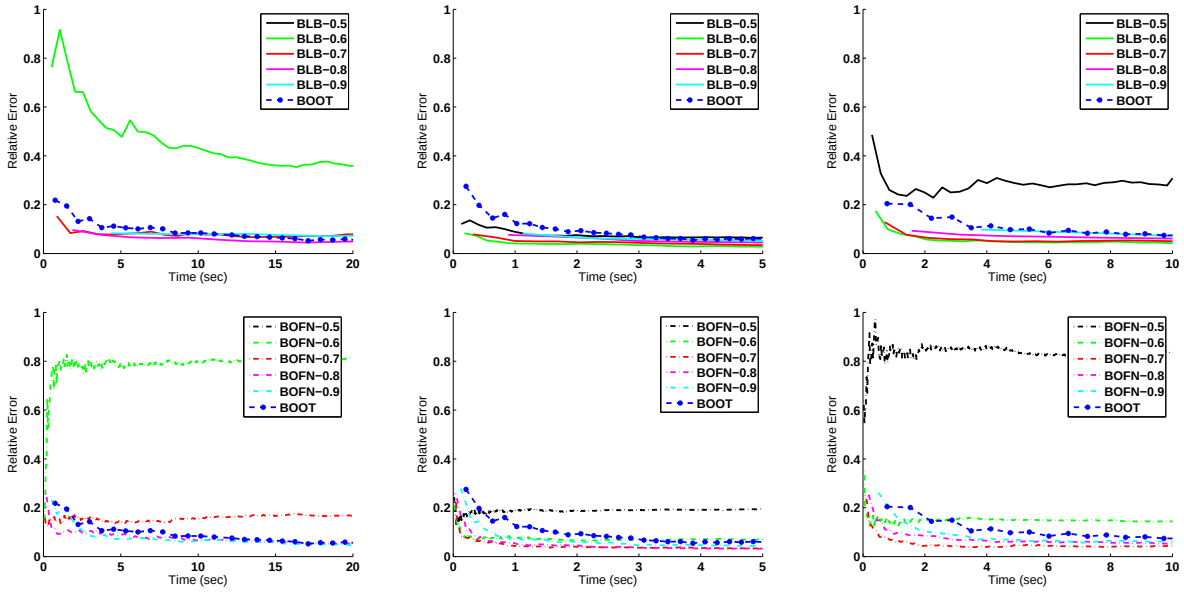


Figure 2.2: Relative error vs. processing time for classification setting with $n = 20,000$. The top row shows BLB with bootstrap (BOOT); bottom row shows b out of n bootstrap (BOFN). For both BLB and BOFN, $b = n^\gamma$ with the value of γ for each trajectory given in the legend. The left column shows results for logistic regression with linear data generating distribution and Gamma $\tilde{X}_i$ distribution. The middle column shows results for logistic regression with quadratic data generating distribution and Gamma $\tilde{X}_i$ distribution. The right column shows results for logistic regression with linear data generating distribution and StudentT $\tilde{X}_i$ distribution.

resamples, in the case of the b out of n bootstrap and the bootstrap, to allow its output to stabilize). Figure 2.3 shows results for the classification setting under the linear data generating distribution with the Gamma and StudentT $\tilde{X}_i$ distributions; qualitatively similar results hold for the Normal $\tilde{X}_i$ distribution. As expected based on our previous results for fixed n, BLB's relative error here is higher than that of the bootstrap for the smallest values of b and n considered. Nonetheless, BLB's relative error decreases to that of the bootstrap as n increases—for all considered values of γ, with $b = n^\gamma$—in accordance with our theoretical analysis; indeed, as n increases, we can set b to progressively more slowly growing functions of n while still achieving low relative error. Furthermore, BLB's relative error is consistently substantially lower than that of the b out of n bootstrap and decreases more quickly to the low relative error of the bootstrap as n increases.

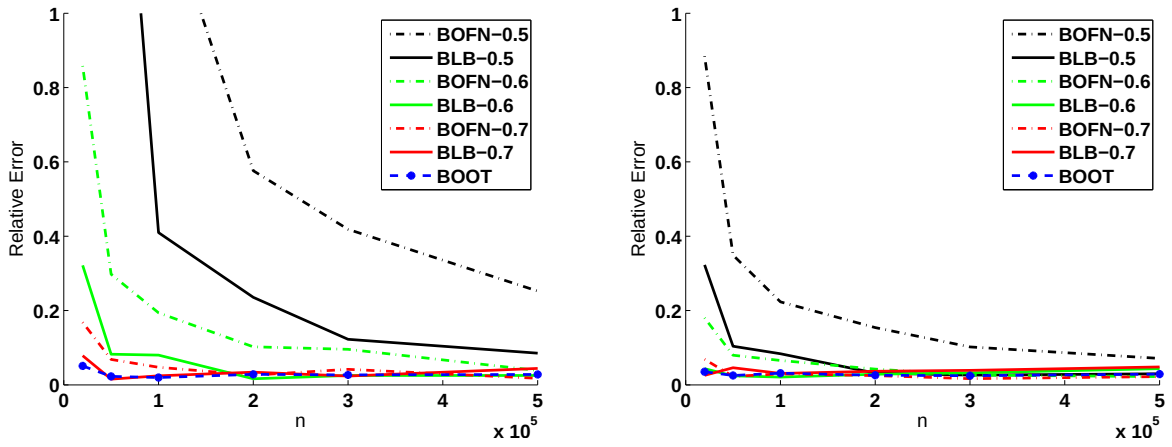


Figure 2.3: Relative error (after convergence) vs. n for BLB, the b out of n bootstrap (BOFN), and the bootstrap (BOOT) in the classification setting. For both BLB and BOFN, $b = n^\gamma$ with the relevant values of γ given in the legend. The left plot shows results for logistic regression with linear data generating distribution and Gamma $\tilde{X}_i$ distribution. The right plot shows results for logistic regression with linear data generating distribution and StudentT $\tilde{X}_i$ distribution.

2.4 Computational Scalability

The experiments of the preceding section, though primarily intended to investigate statistical performance, also provide some insight into computational performance: as seen in Figures 2.1 and 2.2, when computing on a single processor, BLB generally requires less time, and hence less total computation, than the bootstrap to attain comparably high accuracy. Those results only hint at BLB's superior ability to scale computationally to large datasets, which we now demonstrate in full in the following discussion and via large-scale experiments on a distributed computing platform.

As discussed in Section 2.2, modern massive datasets often exceed both the processing and storage capabilities of individual processors or compute nodes, thus necessitating the use of parallel and distributed computing architectures. As a result, the scalability of a quality assessment method is closely tied to its ability to effectively utilize such computing resources.

Recall from our exposition in preceding sections that, due to the large size of bootstrap resamples, the following is the most natural avenue for applying the bootstrap to large-scale data using distributed computing: given data partitioned across a cluster of compute nodes, parallelize the estimate computation on each resample across the cluster, and compute on one resample at a time. This approach, while at least potentially feasible, remains quite problematic. Each computation of the estimate will require the use of an entire cluster of compute nodes, and the bootstrap repeatedly incurs the associated overhead, such as

the cost of repeatedly communicating intermediate data among nodes. Additionally, many cluster computing systems currently in widespread use (e.g., Hadoop MapReduce [41]) store data only on disk, rather than in memory, due to physical size constraints (if the dataset size exceeds the amount of available memory) or architectural constraints (e.g., the need for fault tolerance). In that case, the bootstrap incurs the extreme costs associated with repeatedly reading a very large dataset from disk—reads from disk are orders of magnitude slower than reads from memory. Though disk read costs may be acceptable when (slowly) computing only a single full-data point estimate, they easily become prohibitive when computing many estimates on one hundred or more resamples. Furthermore, as we have seen, executing the bootstrap at scale requires implementing the estimator such that it can be run on data distributed over a cluster of compute nodes.

In contrast, BLB permits computation on multiple (or even all) subsamples and resamples simultaneously in parallel, allowing for straightforward distributed and parallel implementations which enable effective scalability and large computational gains. Because BLB subsamples and resamples can be significantly smaller than the original dataset, they can be transferred to, stored by, and processed independently on individual (or very small sets of) compute nodes. For instance, we can distribute subsamples to different compute nodes and subsequently use intra-node parallelism to compute across different resamples generated from the same subsample. Note that generation and distribution of the subsamples requires only a single pass over the full dataset (i.e., only a single read of the full dataset from disk, if it is stored only on disk), after which all required data (i.e., the subsamples) can potentially be stored in memory. Beyond this significant architectural benefit, we also achieve implementation and algorithmic benefits: we do not need to parallelize the estimator internally to take advantage of the available parallelism, as BLB uses this available parallelism to compute on multiple resamples simultaneously, and exposing the estimator to only b rather than n distinct points significantly reduces the computational cost of estimation, particularly if the estimator computation scales super-linearly.

Given the shortcomings of the m out of n bootstrap and subsampling illustrated in the preceding section, we do not include these methods in the scalability experiments of this section. However, it is worth noting that these procedures have a significant computational shortcoming in the setting of large-scale data: the m out of n bootstrap and subsampling require repeated access to many different random subsets of the original dataset (in contrast to the relatively few, potentially disjoint, subsamples required by BLB), and this access can be quite costly when the data is distributed across a cluster of compute nodes.

We now detail our large-scale experiments on a distributed computing platform. For this empirical study, we use the experimental setup of Section 2.3.2, with some modification to accommodate larger scale and distributed computation. First, we now use $d = 3,000$ and $n = 6,000,000$ so that the size of a full observed dataset is approximately 150 GB. The full dataset is partitioned across a number of compute nodes. We again use simulated data to allow knowledge of ground truth; due to the substantially larger data size and attendant higher running times, we now use 200 independent realizations of datasets of size n to numerically compute the ground truth. As our focus is now computational (rather than statistical)

performance, we present results here for a single data generating distribution which yields representative statistical performance based on the results of the previous section; for a given dataset size, changing the underlying data generating distribution does not alter the computational resources required for storage and processing. For the experiments in this section, we consider the classification setting with StudentT $\tilde{X}_i$ distribution. The mapping between $\tilde{X}_i$ and Y_i remains similar to that of the linear data generating distribution in Section 2.3.2, but with the addition of a normalization factor to prevent degeneracy when using larger d: $Y_i \sim \text{Bernoulli}((1+\exp(-\tilde{X}_i^T \mathbf{1}/\sqrt{d}))^{-1})$. We implement the logistic regression using L-BFGS [63] due to the significantly larger value of d.

We compare the performance of BLB and the bootstrap, both implemented as described above. That is, our implementation of BLB processes all subsamples simultaneously in parallel on independent compute nodes; we use $r = 50$, $s = 5$, and $b = n^{0.7}$. Our implementation of the bootstrap uses all available processors to compute on one resample at a time, with computation of the logistic regression parameter estimates parallelized across the available compute nodes by simply distributing the relevant gradient computations among the different nodes upon which the data is partitioned. We utilize Poisson resampling [77] to generate bootstrap resamples, thereby avoiding the complexity of generating a random multinomial vector of length n in a distributed fashion. Due to high running times, we show results for a single trial of each method, though we have observed little variability in qualitative outcomes during development of these experiments. All experiments in this section are run on Amazon EC2 and implemented in the Scala programming language using the Spark cluster computing framework [82], which provides the ability to either read data from disk (in which case performance is similar to that of Hadoop MapReduce) or cache it in memory across a cluster of compute nodes (provided that sufficient memory is available) for faster repeated access.

In the left plot of Figure 2.4, we show results obtained using a cluster of 10 worker nodes, each having 6 GB of memory and 8 compute cores; thus, the total memory of the cluster is 60 GB, and the full dataset (150 GB) can only be stored on disk (the available disk space is ample and far exceeds the dataset size). As expected, the time required by the bootstrap to produce even a low-accuracy output is prohibitively high, while BLB provides a high-accuracy output quite quickly, in less than the time required to process even a single bootstrap resample. In the right plot of Figure 2.4, we show results obtained using a cluster of 20 worker nodes, each having 12 GB of memory and 4 compute cores; thus, the total memory of the cluster is 240 GB, and we cache the full dataset in memory for faster repeated access. Unsurprisingly, the bootstrap's performance improves significantly with respect to the previous disk-bound experiment. However, the performance of BLB (which also improves), remains substantially better than that of the bootstrap.

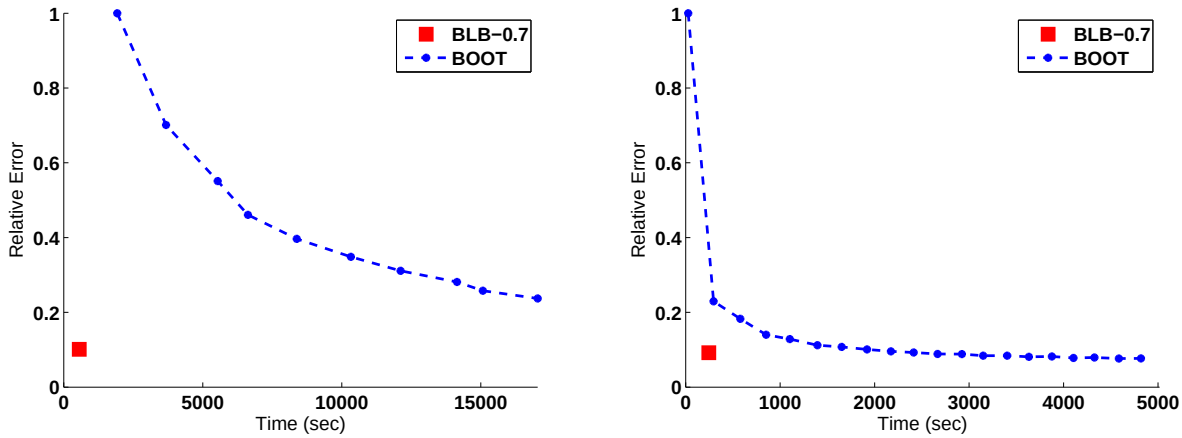


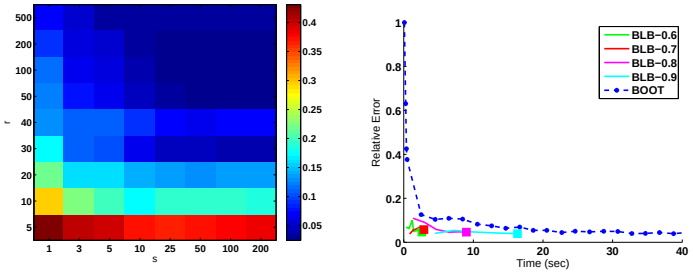
Figure 2.4: Relative error vs. processing time for BLB (with $b = n^{0.7}$) and the bootstrap (BOOT) on 150 GB of data in the classification setting. The left plot shows results with the full dataset stored only on disk; the right plot shows results with the full dataset cached in memory. Because BLB's computation is fully parallelized across all subsamples, we show only the processing time and relative error of BLB's final output.

2.5 Hyperparameter Selection

Like existing resampling-based procedures such as the bootstrap, BLB requires the specification of hyperparameters controlling the number of subsamples and resamples processed. Setting such hyperparameters to be sufficiently large is necessary to ensure good statistical performance; however, setting them to be unnecessarily large results in wasted computation. Prior work on the bootstrap and related procedures—which largely does not address computational issues—generally assumes that a procedure's user will simply select a priori a large, constant number of resamples to be processed (with the exception of [73], which does not provide a general solution for this issue). However, this approach reduces the level of automation of these methods and can be quite inefficient in the large data setting, in which each subsample or resample can require a substantial amount of computation.

Thus, we now examine the dependence of BLB's performance on the choice of r and s, with the goal of better understanding their influence and providing guidance toward achieving adaptive methods for their selection. For any particular application of BLB, we seek to select the minimal values of r and s which are sufficiently large to yield good statistical performance.

Recall that in the simulation study of Section 2.3.2, across all of the settings considered, fairly modest values of r (100 for confidence intervals) and s (from 1-2 for $b = n^{0.9}$ up to 10-20 for $b = n^{0.6}$) were sufficient. The left plot of Figure 2.5 provides further insight into the influence of r and s, giving the relative errors achieved by BLB with $b = n^{0.7}$ for different r, s pairs in the classification setting with linear data generating distribution and StudentT $\tilde{X}_i$



r Stats	CI	STDERR
mean	89.6	67.7
min	50	40
max	150	110

Figure 2.5: Results for BLB hyperparameter selection in the classification setting with linear data generating distribution and StudentT $\tilde{X}_i$ distribution. The left plot shows the relative error achieved by BLB for different values of r and s, with $b = n^{0.7}$. The right plot shows relative error vs. processing time (without parallelization) for BLB using adaptive selection of r and s (the resulting stopping times of the BLB trajectories are marked by large squares) and the bootstrap (BOOT); for BLB, $b = n^{\gamma}$ with the value of γ for each trajectory given in the legend. The table gives statistics of the different values of r selected by BLB's adaptive hyperparameter selection (across multiple subsamples, with $b = n^{0.7}$) when ξ is either our usual confidence interval width-based quality measure (CI), or a component-wise standard error (STDERR); the relative errors achieved by BLB and the bootstrap are comparable in both cases.

distribution. In particular, note that for all but the smallest values of r and s, it is possible to choose these values independently such that BLB achieves low relative error; in this case, selecting $s \geq 3, r \geq 50$ is sufficient.

While these results are useful and provide some guidance for hyperparameter selection, we expect the sufficient values of r and s to change based on the identity of ξ (e.g., we expect a confidence interval to be harder to compute and hence to require larger r than a standard error) and the properties of the underlying data. Thus, to help avoid the need to set r and s to be conservatively and inefficiently large, we now provide a means for adaptive hyperparameter selection, which we validate empirically.

Concretely, to select r adaptively in the inner loop of Algorithm 1, we propose an iterative scheme whereby, for any given subsample j, we continue to process resamples and update $\xi^*_{n,j}$ until it has ceased to change significantly. Noting that the values $\hat{\theta}^*_{n,k}$ used to compute $\xi^*_{n,j}$ are conditionally i.i.d. given a subsample, for most forms of ξ the series of computed $\xi^*_{n,j}$ values will be well behaved and will converge (in many cases at rate $O(1/\sqrt{r})$, though with unknown constant) to a constant target value as more resamples are processed. Therefore, it suffices to process resamples (i.e., to increase r) until we are satisfied that $\xi^*_{n,j}$ has ceased to fluctuate significantly; we propose using Algorithm 2 to assess this convergence. The same scheme can be used to select s adaptively by processing more subsamples (i.e., increasing s) until BLB's output value $s^{-1}\sum_{j=1}^{s}\xi^*_{n,j}$ has stabilized; in this case, one can simultaneously also choose r adaptively and independently for each subsample. When parallelizing across subsamples and resamples, one can simply process batches of subsamples and resamples

Algorithm 2: Convergence Assessment

Input: A series $z^{(1)}, z^{(2)}, \ldots, z^{(t)} \in \mathbb{R}^d$
$w \in \mathbb{N}$: window size ($< t$)
$\epsilon \in \mathbb{R}$: target relative error (> 0)
Output: true if and only if the input series is deemed to have ceased to fluctuate beyond the target relative error

if $\forall j \in [1, w]$, $\frac{1}{d}\sum_{i=1}^{d} \frac{|z_i^{(t-j)} - z_i^{(t)}|}{|z_i^{(t)}|} \leq \epsilon$ **then**
  return *true*
else
  return *false*
end

(with batch size determined by the available parallelism) until the output stabilizes.

The right plot of Figure 2.5 shows the results of applying such adaptive hyperparameter selection in a representative empirical setting from our earlier simulation study (without parallelization). For selection of r we use $\epsilon = 0.05$ and $w = 20$, and for selection of s we use $\epsilon = 0.05$ and $w = 3$. As illustrated in the plot, the adaptive hyperparameter selection allows BLB to cease computing shortly after it has converged (to low relative error), limiting the amount of unnecessary computation that is performed without degradation of statistical performance. Though selected a priori, ϵ and w are more intuitively interpretable and less dependent on the details of ξ and the underlying data generating distribution than r and s. Indeed, the aforementioned specific values of ϵ and w yield results of comparably good quality when also used for the other data generation settings considered in Section 2.3.2, when applied to a variety of real datasets in Section 2.6 below, and when used in conjunction with different forms of ξ (see the table in Figure 2.5, which shows that smaller values of r are selected when ξ is easier to compute). Thus, our scheme significantly helps to alleviate the burden of a priori hyperparameter selection.

Automatic selection of a value of b in a computationally efficient manner would also be desirable but is more difficult due to the inability to easily reuse computations performed for different values of b. One could consider similarly increasing b from some small value until the output of BLB stabilizes (an approach reminiscent of the method proposed in [11] for the m out of n bootstrap); devising a means of doing so efficiently is the subject of future work. Nonetheless, based on our fairly extensive empirical investigation, it seems that $b = n^{0.7}$ is a reasonable and effective choice in many situations.

2.6 Real Data

In this section, we present the results of applying BLB to several different real datasets. In this case, given the absence of ground truth, it is not possible to objectively evaluate the

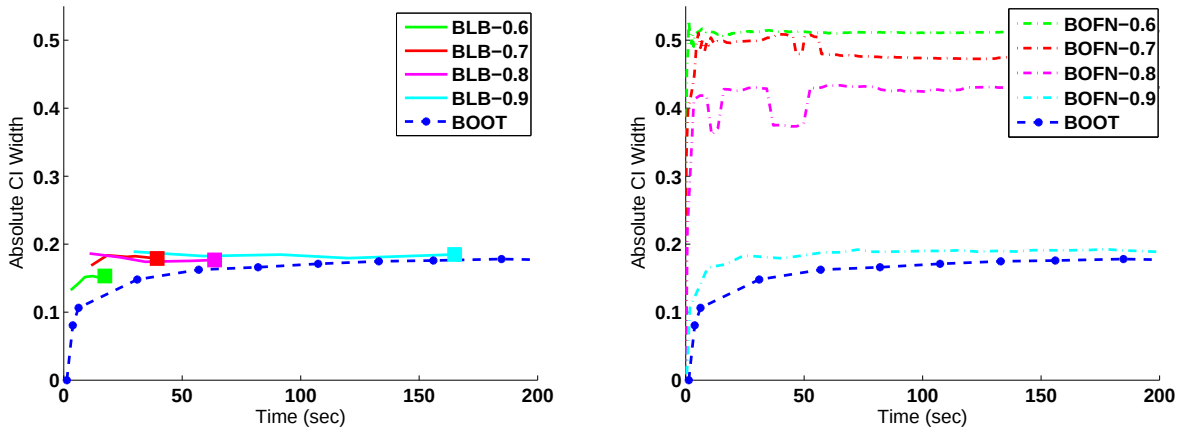


Figure 2.6: Average (across dimensions) absolute confidence interval width vs. processing time on the UCI connect4 dataset (logistic regression, $d = 42, n = 67,557$). The left plot shows results for BLB (using adaptive hyperparameter selection, with the output at convergence marked by large squares) and the bootstrap (BOOT). The right plot shows results for the b out of n bootstrap (BOFN). For both BLB and BOFN, $b = n^\gamma$ with the value of γ for each trajectory given in the legend.

statistical correctness of any particular estimator quality assessment method; rather, we are reduced to comparing the outputs of various methods (in this case, BLB, the bootstrap, and the b out of n bootstrap) to each other. Because we cannot determine the relative error of each procedure's output without knowledge of ground truth, we now instead report the average (across dimensions) absolute confidence interval width yielded by each procedure.

Figure 2.6 shows results for BLB, the bootstrap, and the b out of n bootstrap on the UCI connect4 dataset [34], where the model is logistic regression (as in the classification setting of our simulation study above), $d = 42$, and $n = 67,557$. We select the BLB hyperparameters r and s using the adaptive method described in the preceding section. Notably, the outputs of BLB for all values of b considered, and the output of the bootstrap, are tightly clustered around the same value; additionally, as expected, BLB converges more quickly than the bootstrap. However, the values produced by the b out of n bootstrap vary significantly as b changes, thus further highlighting this procedure's lack of robustness. We have obtained qualitatively similar results on six additional datasets from the UCI dataset repository (ct-slice, magic, millionsong, parkinsons, poker, shuttle) [34] with different estimators (linear regression and logistic regression) and a range of different values of n and d (see the appendix for plots of these results).

2.7 Time Series

While we have focused thus far on the setting of i.i.d. data, variants of the bootstrap—such as the moving block bootstrap and the stationary bootstrap—have been proposed to handle other data analysis settings such as that of time series [33, 45, 50, 54, 66]. These bootstrap variants can be used within BLB, in computing the requisite plugin approximations $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$, to obtain variants of our procedure which are applicable in non-i.i.d. settings. The advantages (e.g., with respect to scalability) of such BLB variants over variants of the bootstrap (and its relatives) remain identical to the advantages discussed above in the context of large-scale i.i.d. data. We briefly demonstrate the extensibility of BLB by combining our procedure with the stationary bootstrap [66] to obtain a "stationary BLB" which is suitable for assessing the quality of estimators applied to large-scale stationary time series data.

To extend BLB in this manner, we must simply alter both the subsample selection mechanism and the resample generation mechanism such that both of these processes respect the underlying data generating process. In particular, for stationary time series data it suffices to select each subsample as a (uniformly) randomly positioned block of length b within the observed time series of length n. Given a subsample of size b, we generate each resample by applying the stationary bootstrap to the subsample to obtain a series of length n. That is, given $p \in [0, 1]$ (a hyperparameter of the stationary bootstrap), we first select uniformly at random a data point in the subsample series and then repeat the following process until we have amassed a new series of length n: with probability $1 - p$ we append to our resample the next point in the subsample series (wrapping around to the beginning if we reach the end of the subsample series), and with probability p we (uniformly at random) select and append a new point in the subsample series. Given subsamples and resamples generated in this manner, we execute the remainder of the BLB procedure as described in Algorithm 1.

We now present simulation results comparing the performance of the bootstrap, BLB, the stationary bootstrap, and stationary BLB. In this experiment, initially introduced in [66], we generate observed data consisting of a stationary time series $X_1, \ldots, X_n \in \mathbb{R}$ where $X_t = Z_t + Z_{t-1} + Z_{t-2} + Z_{t-3} + Z_{t-4}$ and the Z_t are drawn independently from Normal$(0, 1)$. We consider the task of estimating the standard deviation of the rescaled mean $\sum_{i=1}^{n} X_t/\sqrt{n}$, which is approximately 5; we set $p = 0.1$ for the stationary bootstrap and stationary BLB. The results in Table 2.1 (for $n = 5,000$) show the improvement of the stationary bootstrap over the bootstrap, the similar improvement of stationary BLB over BLB, and the fact that the statistical performance of stationary BLB is comparable to that of the stationary bootstrap for $b \geq n^{0.7}$. Note that this exploration of stationary BLB is intended as a proof of concept, and additional investigation would help to further elucidate and perhaps improve the performance characteristics of this BLB extension.

2.A Appendix: Proofs

We provide here full proofs of the theoretical results included in Section 2.3.1 above.

Method	Standard	Stationary
BLB-0.6	$2.2 \pm .1$	$4.2 \pm .1$
BLB-0.7	$2.2 \pm .04$	$4.5 \pm .1$
BLB-0.8	$2.2 \pm .1$	$4.6 \pm .2$
BLB-0.9	$2.2 \pm .1$	$4.6 \pm .1$
BOOT	$2.2 \pm .1$	$4.6 \pm .2$

Table 2.1: Comparison of standard and stationary bootstrap (BOOT) and BLB on stationary time series data with $n = 5,000$. We report the average and standard deviation of estimates (after convergence) of the standard deviation of the rescaled mean aggregated over 10 trials. The true population value of the standard deviation of the rescaled mean is approximately 5.

2.A.1 Consistency

We first define some additional notation, following that used in [77]. Let $l^\infty(\mathcal{F})$ be the set of all uniformly bounded real functions on $\mathcal{F}$, and let $\mathrm{BL}_1(l^\infty(\mathcal{F}))$ denote the set of all functions $h : l^\infty(\mathcal{F}) \to [0,1]$ such that $|h(z_1) - h(z_2)| \leq \|z_1 - z_2\|_\mathcal{F}, \forall z_1, z_2 \in l^\infty(\mathcal{F})$, where $\|\cdot\|_\mathcal{F}$ is the uniform norm for maps from $\mathcal{F}$ to $\mathbb{R}$. We define Pf to be the expectation of $f(X)$ when $X \sim P$; as suggested by this notation, throughout this section we will view distributions such as P, $\mathbb{P}_n$, and $\mathbb{P}_{n,b}^{(j)}$ as maps from some function class $\mathcal{F}$ to $\mathbb{R}$. $E(\cdot)^*$ and $E(\cdot)_*$ denote the outer and inner expectation of $(\cdot)$, respectively, and we indicate outer probability via P^*. $X \overset{d}{=} Y$ denotes that the random variables X and Y are equal in distribution, and $\mathcal{F}_\delta$ is defined as the set $\{f - g : f, g \in \mathcal{F}, \rho_P(f - g) < \delta\}$, where $\rho_P(\cdot)$ is the variance semimetric: $\rho_P(f) = (P(f - Pf)^2)^{1/2}$.

Following prior analyses of the bootstrap [36, 77], we first observe that, conditioning on $\mathbb{P}_{n,b}^{(j)}$ for any j as $b, n \to \infty$, resamples from the subsampled empirical distribution $\mathbb{P}_{n,b}^{(j)}$ behave asymptotically as though they were drawn directly from P, the true underlying distribution:

Lemma 1. *Given $\mathbb{P}_{n,b}^{(j)}$ for any j, let $X_1^*, \ldots, X_n^* \sim \mathbb{P}_{n,b}^{(j)}$ i.i.d., and define $\mathbb{P}_{n,b}^* = n^{-1} \sum_{i=1}^n \delta_{X_i^*}$. Additionally, we define the resampled empirical process $\mathbb{G}_{n,b}^* = \sqrt{n}(\mathbb{P}_{n,b}^* - \mathbb{P}_{n,b}^{(j)})$. Then, for $\mathcal{F}$ a Donsker class of measurable functions such that $\mathcal{F}_\delta$ is measurable for every $\delta > 0$,*

$$\sup_{h \in BL_1(l^\infty(\mathcal{F}))} \left| E_{\mathbb{P}_{n,b}^{(j)}} h(\mathbb{G}_{n,b}^*) - Eh(\mathbb{G}_P) \right| \overset{P^*}{\to} 0,$$

as $n \to \infty$, for any sequence $b \to \infty$, where $E_{\mathbb{P}_{n,b}^{(j)}}$ denotes expectation conditional on the contents of the subscript and $\mathbb{G}_P$ is a P-Brownian bridge process. "Furthermore, the sequence $E_{\mathbb{P}_{n,b}^{(j)}} h(\mathbb{G}_{n,b}^*)^* - E_{\mathbb{P}_{n,b}^{(j)}} h(\mathbb{G}_{n,b}^*)_*$ converges to zero in probability for every $h \in BL_1(l^\infty(\mathcal{F}))$. If $P^*\|f - Pf\|_\mathcal{F}^2 < \infty$, then the convergence is also outer almost surely." [77]

Proof. Note that $\mathbb{P}_{n,b}^{(j)} \overset{d}{=} \mathbb{P}_b$. Hence, applying Theorem 3.6.3 in [77] with the identification $(n, k_n) \leftrightarrow (b, n)$ yields the desired result. □

Lemma 1 states that, conditionally on the sequence $\mathbb{P}_{n,b}^{(j)}$, the sequence of processes $\mathbb{G}_{n,b}^*$ converges in distribution to the P-Brownian bridge process $\mathbb{G}_P$, in probability. Noting that the empirical process $\mathbb{G}_n = \sqrt{n}(\mathbb{P}_n - P)$ also converges in distribution to $\mathbb{G}_P$ (recall that $\mathcal{F}$ is a Donsker class by assumption), it follows that size n resamples generated from $\mathbb{P}_{n,b}^{(j)}$ behave asymptotically as though they were drawn directly from P. Under standard assumptions, it then follows that $\xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P)) \xrightarrow{P} 0$:

Lemma 2. *Under the assumptions of Theorem 1, for any j,*

$$\xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P)) \xrightarrow{P} 0$$

as $n \to \infty$, for any sequence $b \to \infty$.

Proof. Let R be the random element to which $\sqrt{n}(\phi(\mathbb{P}_n) - \phi(P))$ converges in distribution; note that the functional delta method [76] provides the form of R in terms of ϕ and P. The delta method for the bootstrap (see Theorem 23.9 in [76]) in conjunction with Lemma 1 implies that, under our assumptions, $\sqrt{n}(\phi(\mathbb{P}_{n,b}^*) - \phi(\mathbb{P}_{n,b}^{(j)}))$ also converges conditionally in distribution to R, given $\mathbb{P}_{n,b}^{(j)}$, in probability. Thus, the distribution of $\sqrt{n}(\phi(\mathbb{P}_n) - \phi(P))$ and the distribution of $\sqrt{n}(\phi(\mathbb{P}_{n,b}^*) - \phi(\mathbb{P}_{n,b}^{(j)}))$, the latter conditionally on $\mathbb{P}_{n,b}^{(j)}$, have the same asymptotic limit in probability. As a result, given the assumed continuity of ξ, it follows that $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ and $\xi(Q_n(P))$ have the same asymptotic limit, in probability. □

The above lemma indicates that each individual $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ is asymptotically consistent as $b, n \to \infty$. Theorem 1 immediately follows:

Proof of Theorem 1. Lemma 2 in conjunction with the continuous mapping theorem [76] implies the desired result. □

2.A.2 Higher-Order Correctness

We first prove two supporting lemmas.

Lemma 3. *Assume that $X_1, \ldots, X_b \sim P$ are i.i.d., and let $\hat{p}_k(X_1, \ldots, X_b)$ be the sample version of p_k based on $X_1, \ldots, X_b$, as defined in Theorem 2. Then, assuming that $E[\hat{p}_k(X_1, \ldots, X_b)^2] < \infty$,*

$$\mathrm{Var}(\hat{p}_k(X_1, \ldots, X_b) - p_k) = \mathrm{Var}(\hat{p}_k(X_1, \ldots, X_b)) = O(1/b).$$

Proof. By definition, the $\hat{p}_k$ are simply polynomials in sample moments. Thus, we can write

$$\hat{p}_k = \hat{p}_k(X_1, \ldots, X_b) = \sum_{\beta=1}^{B} c_\beta \prod_{\alpha=1}^{A_\beta} \left(b^{-1} \sum_{i=1}^{b} g_\alpha^{(\beta)}(X_i) \right), \tag{2.6}$$

where each $g_\alpha^{(\beta)}$ raises its argument to some power. Now, observe that for any β,

$$V_\beta = \prod_{\alpha=1}^{A_\beta} \left(b^{-1} \sum_{i=1}^{b} g_\alpha^{(\beta)}(X_i) \right)$$

is a V-statistic of order A_β applied to the b observations $X_1, \ldots, X_b$. Let $h_\beta(x_1, \ldots, x_{A_\beta})$ denote the kernel of this V-statistic, which is a symmetrized version of $\prod_{\alpha=1}^{A_\beta} g_\alpha^{(\beta)}(x_\alpha)$. It follows that $\hat{p}_k = \sum_{\beta=1}^{B} c_\beta V_\beta$ is itself a V-statistic of order $A = \max_\beta A_\beta$ with kernel $h(x_1, \ldots, x_A) = \sum_{\beta=1}^{B} c_\beta h_\beta(x_1, \ldots, x_{A_\beta})$, applied to the b observations $X_1, \ldots, X_b$. Let U denote the corresponding U-statistic having kernel h. Then, using Proposition 3.5(ii) and Corollary 3.2(i) in [70], we have

$$\mathrm{Var}(\hat{p}_k - p_k) = \mathrm{Var}(\hat{p}_k) = \mathrm{Var}(U) + O(b^{-2}) \leq \frac{A}{b} \mathrm{Var}(h(X_1, \ldots, X_A)) + O(b^{-2}) = O(1/b).$$

□

Lemma 4. *Assume that $X_1, \ldots, X_b \sim P$ are i.i.d., and let $\hat{p}_k(X_1, \ldots, X_b)$ be the sample version of p_k based on $X_1, \ldots, X_b$, as defined in Theorem 2. Then, assuming that $E|\hat{p}_k(X_1, \ldots, X_b)| < \infty$,*

$$|E[\hat{p}_k(X_1, \ldots, X_b)] - p_k| = O(1/b).$$

Proof. As noted in the proof of Lemma 3, we can write

$$\hat{p}_k(X_1, \ldots, X_b) = \sum_{\beta=1}^{B} c_\beta \prod_{\alpha=1}^{A_\beta} \left(b^{-1} \sum_{i=1}^{b} g_\alpha^{(\beta)}(X_i) \right),$$

where each $g_\alpha^{(\beta)}$ raises its argument to some power. Similarly,

$$p_k = \sum_{\beta=1}^{B} c_\beta \prod_{\alpha=1}^{A_\beta} E g_\alpha^{(\beta)}(X_1),$$

and so

$$\begin{aligned} |E[\hat{p}_k(X_1, \ldots, X_b)] - p_k| &= \left| \sum_{\beta=1}^{B} c_\beta \prod_{\alpha=1}^{A_\beta} \left(b^{-1} \sum_{i=1}^{b} g_\alpha^{(\beta)}(X_i) \right) - \sum_{\beta=1}^{B} c_\beta \prod_{\alpha=1}^{A_\beta} E g_\alpha^{(\beta)}(X_1) \right| \\ &\leq \sum_{\beta=1}^{B} |c_\beta| \cdot \left| E \left[\prod_{\alpha=1}^{A_\beta} \left(b^{-1} \sum_{i=1}^{b} g_\alpha^{(\beta)}(X_i) \right) \right] - \prod_{\alpha=1}^{A_\beta} E g_\alpha^{(\beta)}(X_1) \right|. \end{aligned}$$

Given that the number of terms in the outer sum on the right-hand side is constant with respect to b, to prove the desired result it is sufficient to show that, for any β,

$$\Delta_\beta = \left| E\left[\prod_{\alpha=1}^{A_\beta}\left(b^{-1}\sum_{i=1}^{b} g_\alpha^{(\beta)}(X_i)\right)\right] - \prod_{\alpha=1}^{A_\beta} E g_\alpha^{(\beta)}(X_1)\right| = O\left(\frac{1}{b}\right).$$

Observe that

$$E\left[\prod_{\alpha=1}^{A_\beta}\left(b^{-1}\sum_{i=1}^{b} g_\alpha^{(\beta)}(X_i)\right)\right] = b^{-A_\beta} E\left[\sum_{i_1,\ldots,i_{A_\beta}=1}^{b}\ \prod_{\alpha=1}^{A_\beta} g_\alpha^{(\beta)}(X_{i_\alpha})\right]. \tag{2.7}$$

If $i_1,\ldots,i_{A_\beta}$ are all distinct, then $E\prod_{\alpha=1}^{A_\beta} g_\alpha^{(\beta)}(X_{i_\alpha}) = \prod_{\alpha=1}^{A_\beta} E g_\alpha^{(\beta)}(X_1)$ because $X_1,\ldots,X_b$ are i.i.d.. Additionally, the right-hand summation in (2.7) has $b!/(b-A_\beta)!$ terms in which $i_1,\ldots,i_{A_\beta}$ are all distinct; correspondingly, there are $b^{A_\beta} - b!/(b-A_\beta)!$ terms in which $\exists\alpha,\alpha'$ s.t. $i_\alpha = i_{\alpha'}$. Therefore, it follows that

$$E\left[\prod_{\alpha=1}^{A_\beta}\left(b^{-1}\sum_{i=1}^{b} g_\alpha^{(\beta)}(X_i)\right)\right] = b^{-A_\beta}\left[\frac{b!}{(b-A_\beta)!}\prod_{\alpha=1}^{A_\beta} E g_\alpha^{(\beta)}(X_1) \quad + \sum_{\substack{1\le i_1,\ldots,i_{A_\beta}\le b \\ \exists\alpha,\alpha' \text{ s.t. } i_\alpha = i_{\alpha'}}} E\prod_{\alpha=1}^{A_\beta} g_\alpha^{(\beta)}(X_{i_\alpha})\right]$$

and

$$\begin{aligned}
\Delta_\beta &= \left| E\left[\prod_{\alpha=1}^{A_\beta}\left(b^{-1}\sum_{i=1}^{b} g_\alpha^{(\beta)}(X_i)\right)\right] - \prod_{\alpha=1}^{A_\beta} E g_\alpha^{(\beta)}(X_1)\right| \\
&= b^{-A_\beta}\left|\left(\frac{b!}{(b-A_\beta)!} - b^{A_\beta}\right)\prod_{\alpha=1}^{A_\beta} E g_\alpha^{(\beta)}(X_1) \quad + \sum_{\substack{1\le i_1,\ldots,i_{A_\beta}\le b \\ \exists\alpha,\alpha' \text{ s.t. } i_\alpha = i_{\alpha'}}} E\prod_{\alpha=1}^{A_\beta} g_\alpha^{(\beta)}(X_{i_\alpha})\right| \\
&\le b^{-A_\beta}\left|\frac{b!}{(b-A_\beta)!} - b^{A_\beta}\right| \cdot \left|\prod_{\alpha=1}^{A_\beta} E g_\alpha^{(\beta)}(X_1)\right| \ + \ b^{-A_\beta}\left|b^{A_\beta} - \frac{b!}{(b-A_\beta)!}\right| C,
\end{aligned} \tag{2.8}$$

where

$$C \quad = \max_{\substack{1\le i_1,\ldots,i_{A_\beta}\le b \\ \exists\alpha,\alpha' \text{ s.t. } i_\alpha = i_{\alpha'}}} \left| E\prod_{\alpha=1}^{A_\beta} g_\alpha^{(\beta)}(X_{i_\alpha})\right|.$$

Note that C is a constant with respect to b. Also, simple algebraic manipulation shows that

$$\frac{b!}{(b-A_\beta)!} = b^{A_\beta} - \kappa b^{A_\beta - 1} + O(b^{A_\beta - 2})$$

for some $\kappa > 0$. Thus, plugging into equation (2.8), we obtain the desired result:

$$\Delta_\beta \leq b^{-A_\beta} \left| \kappa b^{A_\beta - 1} + O(b^{A_\beta - 2}) \right| \cdot \left| \prod_{\alpha=1}^{A_\beta} E g_\alpha^{(\beta)}(X_1) \right| + b^{-A_\beta} \left| \kappa b^{A_\beta - 1} + O(b^{A_\beta - 2}) \right| C = O\left(\frac{1}{b}\right).$$

□

We now provide full proofs of Theorem 2, Remark 1, and Theorem 3.

Proof of Theorem 2. Summing the expansion (2.3) over j, we have

$$s^{-1} \sum_{j=1}^{s} \xi(Q_n(\mathbb{P}_{n,b}^{(j)})) = z + n^{-1/2} s^{-1} \sum_{j=1}^{s} \hat{p}_1^{(j)} + n^{-1} s^{-1} \sum_{j=1}^{s} \hat{p}_2^{(j)} + o_P\left(\frac{1}{n}\right).$$

Subtracting the corresponding expansion (2.2) for $\xi(Q_n(P))$, we then obtain

$$\left| s^{-1} \sum_{j=1}^{s} \xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P)) \right| \leq n^{-1/2} \left| s^{-1} \sum_{j=1}^{s} \hat{p}_1^{(j)} - p_1 \right| + n^{-1} \left| s^{-1} \sum_{j=1}^{s} \hat{p}_2^{(j)} - p_2 \right| + o_P\left(\frac{1}{n}\right). \tag{2.9}$$

We now further analyze the first two terms on the right-hand side of the above expression; for the remainder of this proof, we assume that $k \in \{1, 2\}$. Observe that, for fixed k, the $\hat{p}_k^{(j)}$ are conditionally i.i.d. given $X_1, \ldots, X_n$ for all j, and so

$$\mathrm{Var}\left(s^{-1} \sum_{j=1}^{s} \left([\hat{p}_k^{(j)} - p_k] - E[\hat{p}_k^{(1)} - p_k | \mathbb{P}_n] \right) \middle| \mathbb{P}_n \right) = \frac{\mathrm{Var}(\hat{p}_k^{(1)} - p_k | \mathbb{P}_n)}{s},$$

where we denote by $E[\hat{p}_k^{(1)} - p_k | \mathbb{P}_n]$ and $\mathrm{Var}(\hat{p}_k^{(1)} - p_k | \mathbb{P}_n)$ the expectation and variance of $\hat{p}_k^{(1)} - p_k$ over realizations of $\mathbb{P}_{n,b}^{(1)}$ conditionally on $X_1, \ldots, X_n$. Now, given that $\hat{p}_k^{(j)}$ is a permutation-symmetric function of size b subsets of $X_1, \ldots, X_n$, $E[\hat{p}_k^{(1)} - p_k | \mathbb{P}_n]$ is a U-statistic of order b. Hence, we can apply Corollary 3.2(i) in [70] in conjunction with Lemma 3 to find that

$$\mathrm{Var}\left(E[\hat{p}_k^{(1)} - p_k | \mathbb{P}_n] - E[\hat{p}_k^{(1)} - p_k] \right) = \mathrm{Var}\left(E[\hat{p}_k^{(1)} - p_k | \mathbb{P}_n] \right) \leq \frac{b}{n} \mathrm{Var}(\hat{p}_k^{(1)} - p_k) = O\left(\frac{1}{n}\right).$$

From the result of Lemma 4, we have

$$|E[\hat{p}_k^{(1)} - p_k]| = O\left(\frac{1}{b}\right).$$

Combining the expressions in the previous three panels, we find that

$$\left| s^{-1} \sum_{j=1}^{s} \hat{p}_k^{(j)} - p_k \right| = O_P\left(\frac{\sqrt{\mathrm{Var}(\hat{p}_k^{(1)} - p_k | \mathbb{P}_n)}}{\sqrt{s}} \right) + O_P\left(\frac{1}{\sqrt{n}}\right) + O\left(\frac{1}{b}\right).$$

Finally, plugging into equation (2.9) with $k = 1$ and $k = 2$, we obtain the desired result. □

Proof of Remark 1. Observe that

$$\mathrm{Var}(\hat{p}_k^{(1)} - p_k|\mathbb{P}_n) \leq E[(\hat{p}_k^{(1)} - p_k)^2|\mathbb{P}_n] = E[(\hat{p}_k^{(1)} - p_k)^2|\mathbb{P}_n] - E[(\hat{p}_k^{(1)} - p_k)^2] + E[(\hat{p}_k^{(1)} - p_k)^2].$$

Given that $\hat{p}_k^{(1)}$ is a polynomial in the moments of $\mathbb{P}_{n,b}^{(1)}$, $\hat{q}_k^{(1)} = (\hat{p}_k^{(1)} - p_k)^2$ is also a polynomial in the moments of $\mathbb{P}_{n,b}^{(1)}$. Hence, Lemma 3 applies to $\hat{q}_k^{(1)}$. Additionally, $\hat{q}_k^{(1)}$ is a permutation-symmetric function of size b subsets of $X_1, \ldots, X_n$, and so $E[\hat{q}_k^{(1)}|\mathbb{P}_n]$ is a U-statistic of order b. Therefore, applying Corollary 3.2(i) in [70] in conjunction with Lemma 3, we find that

$$\mathrm{Var}\left(E[(\hat{p}_k^{(1)} - p_k)^2|\mathbb{P}_n] - E[(\hat{p}_k^{(1)} - p_k)^2]\right) = \mathrm{Var}\left(E[\hat{q}_k^{(1)}|\mathbb{P}_n]\right) \leq \frac{b}{n}\mathrm{Var}(\hat{q}_k^{(1)}) = O\left(\frac{1}{n}\right).$$

Now,

$$E[(\hat{p}_k^{(1)} - p_k)^2] = \mathrm{Var}(\hat{p}_k^{(1)} - p_k) + E[\hat{p}_k^{(1)} - p_k]^2.$$

By Lemmas 3 and 4, $\mathrm{Var}(\hat{p}_k^{(1)} - p_k) = O(1/b)$ and $E[\hat{p}_k^{(1)} - p_k]^2 = O(1/b^2)$. Combining with the expressions in the previous three panels, we obtain the desired result:

$$\mathrm{Var}(\hat{p}_k^{(1)} - p_k|\mathbb{P}_n) = O_P\left(\frac{1}{\sqrt{n}}\right) + O\left(\frac{1}{b}\right) + O\left(\frac{1}{b^2}\right) = O_P\left(\frac{1}{\sqrt{n}}\right) + O\left(\frac{1}{b}\right).$$

□

Proof of Theorem 3. As noted in the proof of Theorem 2,

$$\left|s^{-1}\sum_{j=1}^{s}\xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P))\right| \leq n^{-1/2}\left|s^{-1}\sum_{j=1}^{s}\hat{p}_1^{(j)} - p_1\right| + n^{-1}\left|s^{-1}\sum_{j=1}^{s}\hat{p}_2^{(j)} - p_2\right| + o_P\left(\frac{1}{n}\right). \tag{2.10}$$

Throughout this proof, we assume that $k \in \{1, 2\}$. Under the assumptions of this theorem, the $\mathbb{P}_{n,b}^{(j)}$ are based on disjoint subsets of the n observations and so are i.i.d.. Hence, for any k, the $\hat{p}_k^{(j)}$ are i.i.d. for all j, and so using Lemma 3,

$$\mathrm{Var}\left(\left[s^{-1}\sum_{j=1}^{s}\hat{p}_k^{(j)} - p_k\right] - E[\hat{p}_k^{(1)} - p_k]\right) = \frac{\mathrm{Var}(\hat{p}_k^{(1)} - p_k)}{s} = O\left(\frac{1}{bs}\right).$$

Additionally, from the result of Lemma 4, we have

$$|E[\hat{p}_k^{(1)} - p_k]| = O\left(\frac{1}{b}\right).$$

Combining the expressions in the previous two panels, we find that

$$\left|s^{-1}\sum_{j=1}^{s}\hat{p}_k^{(j)} - p_k\right| = O_P\left(\frac{1}{\sqrt{bs}}\right) + O\left(\frac{1}{b}\right).$$

Finally, plugging into equation (2.10) with $k = 1$ and $k = 2$, we obtain the desired result. □

2.B Appendix: Additional Real Data Results

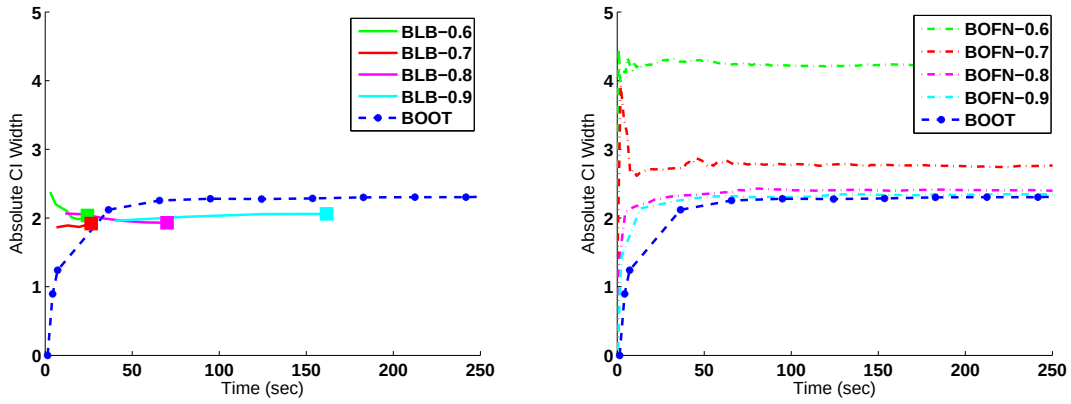


Figure 2.7: Average (across dimensions) absolute confidence interval width vs. processing time on the UCI ct-slice dataset (linear regression, $d = 385$, $n = 53,500$). The left plot shows results for BLB (using adaptive hyperparameter selection, with the output at convergence marked by large squares) and the bootstrap (BOOT). The right plot shows results for the b out of n bootstrap (BOFN).

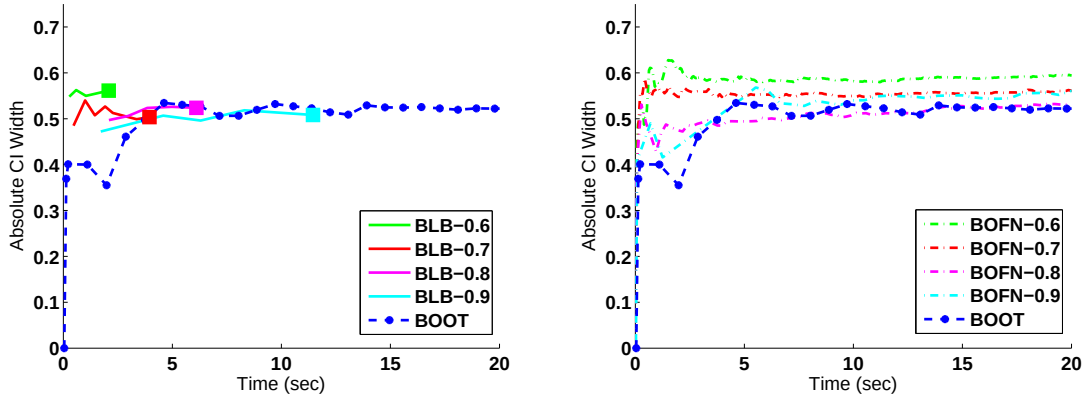


Figure 2.8: Average (across dimensions) absolute confidence interval width vs. processing time on the UCI magic dataset (logistic regression, $d = 10$, $n = 19,020$). The left plot shows results for BLB (using adaptive hyperparameter selection, with the output at convergence marked by large squares) and the bootstrap (BOOT). The right plot shows results for the b out of n bootstrap (BOFN).

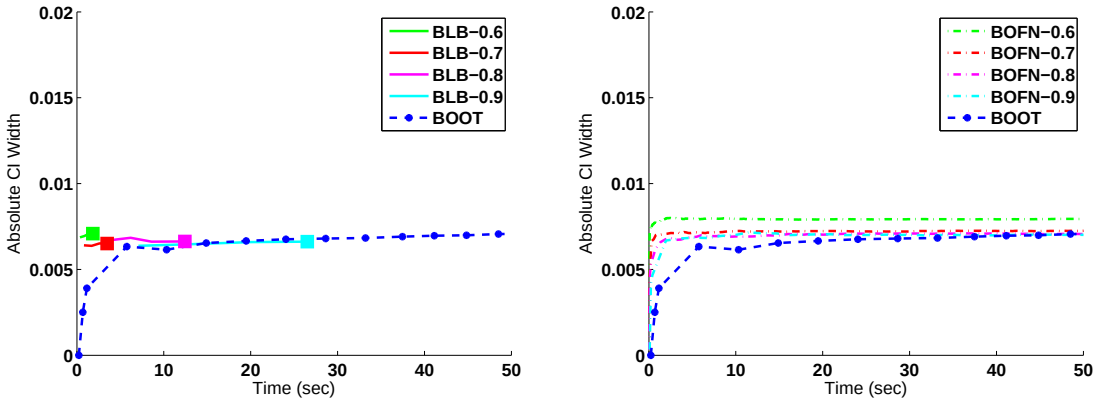


Figure 2.9: Average (across dimensions) absolute confidence interval width vs. processing time on the UCI millionsong dataset (linear regression, $d = 90$, $n = 50,000$). The left plot shows results for BLB (using adaptive hyperparameter selection, with the output at convergence marked by large squares) and the bootstrap (BOOT). The right plot shows results for the b out of n bootstrap (BOFN).

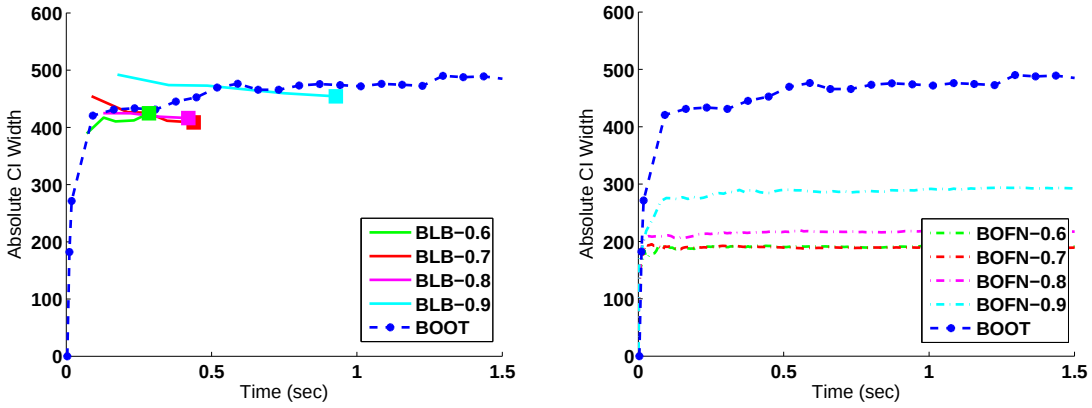


Figure 2.10: Average (across dimensions) absolute confidence interval width vs. processing time on the UCI parkinsons dataset (linear regression, $d = 16$, $n = 5,875$). The left plot shows results for BLB (using adaptive hyperparameter selection, with the output at convergence marked by large squares) and the bootstrap (BOOT). The right plot shows results for the b out of n bootstrap (BOFN).

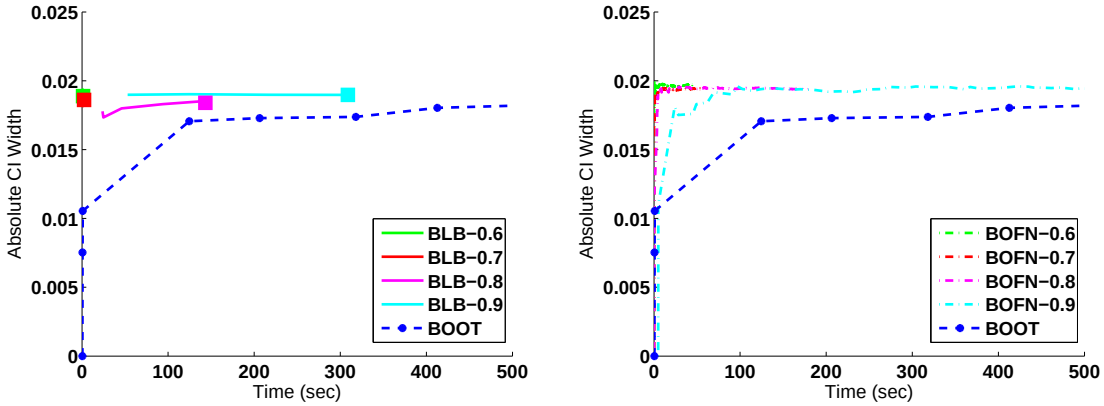


Figure 2.11: Average (across dimensions) absolute confidence interval width vs. processing time on the UCI poker dataset (logistic regression, $d = 10$, $n = 50,000$). The left plot shows results for BLB (using adaptive hyperparameter selection, with the output at convergence marked by large squares) and the bootstrap (BOOT). The right plot shows results for the b out of n bootstrap (BOFN).

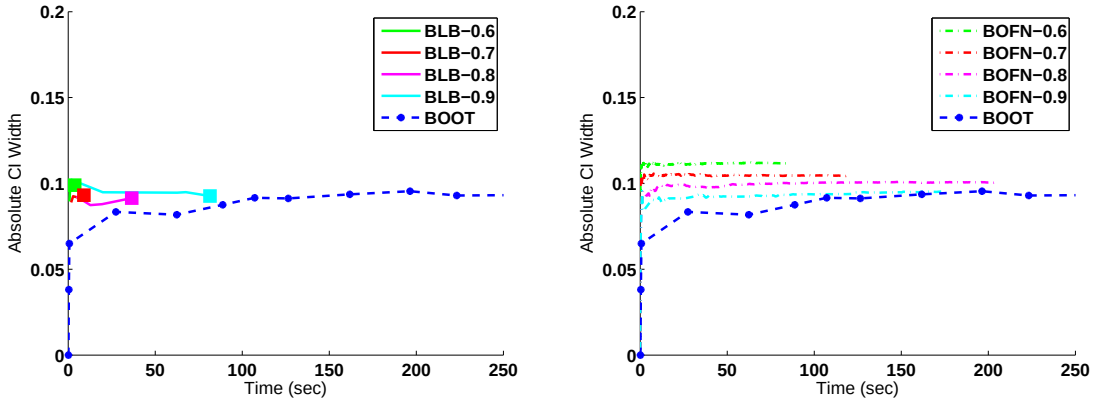


Figure 2.12: Average (across dimensions) absolute confidence interval width vs. processing time on the UCI shuttle dataset (logistic regression, $d = 9$, $n = 43,500$). The left plot shows results for BLB (using adaptive hyperparameter selection, with the output at convergence marked by large squares) and the bootstrap (BOOT). The right plot shows results for the b out of n bootstrap (BOFN).

Chapter 3

A General Bootstrap Performance Diagnostic

3.1 Introduction

Modern datasets are growing rapidly in size and are increasingly subjected to diverse, rapidly evolving sets of complex and exploratory queries, often crafted by non-statisticians. These developments render generic applicability and automation of data analysis methodology particularly desirable, both to allow the statistician to work more efficiently and to allow the non-statistician to correctly and effectively utilize more sophisticated inferential techniques. For example, the development of generic techniques for training classifiers and evaluating their generalization ability has allowed this methodology to spread well beyond the boundaries of the machine learning and statistics research community, to great practical benefit. More generally, estimation techniques for a variety of settings have been rendered generically usable. However, except in some restricted settings, the fundamental inferential problem of assessing the quality of estimates based upon finite data has eluded a highly automated solution.

Assessment of an estimate's quality—for example, its variability (e.g., in the form of a confidence region), its bias, or its risk—is essential to both its interpretation and use. Indeed, such quality assessments underlie a variety of core statistical tasks, such as calibrated inference regarding parameter values, bias correction, and hypothesis testing. Beyond simply enabling other statistical methodology, however, estimator quality assessments can also have more direct utility, whether by improving human interpretation of inferential outputs or by allowing more efficient management of data collection and processing resources. For instance, we might seek to collect or process only as much data as is required to yield estimates of some desired quality, thereby avoiding the cost (e.g., in time or money) of collecting or processing more data than is necessary. Such an approach in fact constitutes an active line of work in research on large database systems, which seeks to answer queries on massive datasets quickly by only applying them to subsamples of the total available data [2, 52]. The result of

applying a query to only a subsample is in fact an estimate of the query's output if applied to the full dataset, and effective implementation of a system using this technique requires an automatic ability to accurately assess the quality of such estimates for generic queries.

In recent decades, the bootstrap [30, 33] has emerged as a powerful and widely used means of assessing estimator quality, with its popularity due in no small part to its relatively generic applicability. Unlike classical methods—which have generally relied upon analytic asymptotic approximations requiring deep analysis of specific classes of estimators in specific settings [67]—the bootstrap can be straightforwardly applied, via a simple computational mechanism, to a broad range of estimators. Since its inception, theoretical work has shown that the bootstrap is broadly consistent [8, 36, 77] and can be higher-order correct [44]. As a result, the bootstrap (and its various relatives and extensions) provides perhaps the most promising avenue for obtaining a generically applicable, automated estimator quality assessment capability.

Unfortunately, however, while the bootstrap is relatively automatic in comparison to its classical predecessors, it remains far from being truly automatically usable, as evaluating and ensuring its accuracy is often a challenge even for experts in the methodology. Indeed, like any inferential procedure, despite its excellent theoretical properties and frequently excellent empirical performance, the bootstrap is not infallible. For example, it may fail to be consistent in particular settings (i.e., for particular pairs of estimators and data generating distributions) [69, 9]. While theoretical conditions yielding consistency are well known, they can be non-trivial to verify analytically and provide little useful guidance in the absence of manual analysis. Furthermore, even if consistent, the bootstrap may exhibit poor performance on finite samples.

Thus, it would be quite advantageous to have some means of diagnosing poor performance or failure of the bootstrap in an automatic, data-driven fashion, without requiring significant manual analysis. That is, we would like a diagnostic procedure which is analogous to the manner in which we evaluate performance in the setting of supervised learning (e.g., classification), in which we directly and empirically evaluate generalization error (e.g., via a held-out validation set or cross-validation). Unfortunately, prior work on bootstrap diagnostics (see [19] for a comprehensive survey) does not provide a satisfactory solution, as existing diagnostic methods target only specific bootstrap failure modes, are often brittle or difficult to apply, and generally lack substantive empirical evaluations. For example, a theoretical result of Beran [6] regarding bootstrap asymptotics has been proposed as the basis of a diagnostic for bootstrap inconsistency; however, it is unclear how to reliably construct and interpret the diagnostic plots required by this proposal, and the limited existing empirical evaluation reveals it to be of questionable practical utility [19]. Other work has sought to diagnose bootstrap failure specifically due to incorrect standardization of the quantity being bootstrapped (which could occur if an estimator's convergence rate is unknown or incorrectly determined), use of an incorrect resampling model (if, for example, the data has a correlation structure that is not fully known a priori), or violation of an assumption of pivotality of the quantity being bootstrapped [19]. Additionally, jackknife-after-bootstrap and bootstrap-after-bootstrap calculations have been proposed as a means of evaluating the

stability of the bootstrap's outputs [32, 19]; while such procedures can be useful data analysis tools, their utility as the basis of a diagnostic remains limited, as, among other things, it is unclear whether they will behave correctly in settings where the bootstrap is inconsistent.

In contrast to prior work, we present here a general bootstrap performance diagnostic which does not target any particular bootstrap failure mode but rather directly and automatically determines whether or not the bootstrap is performing satisfactorily (i.e., providing sufficiently accurate outputs) when applied to a given dataset and estimator. The key difficulty in evaluating the accuracy of the bootstrap's (or any estimator quality assessment procedure's) outputs is the lack of ready availability of even approximate comparisons to ground truth estimate quality. While comparisons to ground truth labels are readily obtained in the case of supervised learning via use of a held-out validation set or cross-validation, comparing to ground truth in the context of estimator quality assessment requires access to the (unknown) sampling distribution of the estimator in question. We surmount this difficulty by constructing a proxy to ground truth for various small sample sizes (smaller than that of our full observed dataset) and comparing the bootstrap's outputs to this proxy, requiring that they converge to the ground truth proxy as the sample size is increased. This approach is enabled by the increasing availability of large datasets and more powerful computational resources. We show via an extensive empirical evaluation, on a variety of estimators and simulated and real data, that the resulting diagnostic is effective in determining—fully automatically—whether or not the bootstrap is performing satisfactorily in a given setting.

In Section 3.2, we formalize our statistical setting and notation. We introduce our diagnostic in full detail in Section 3.3. Sections 3.4 and 3.5 present the results of our evaluations on simulated and real data, respectively.

3.2 Setting and Notation

We assume that we observe n data points $\mathcal{D} = (X_1, \ldots, X_n)$ sampled i.i.d. from some unknown distribution P; let $\mathbb{P}_n = n^{-1} \sum_{i=1}^{n} \delta_{X_i}$ be the empirical distribution of the observed data. Based upon this dataset, we form an estimate $\hat{\theta}(\mathcal{D})$ of some parameter $\theta(P)$ of P; note that, unlike $\theta(P)$, $\hat{\theta}(\mathcal{D})$ is a random quantity due to its dependence on the data $\mathcal{D}$. We then seek to form an assessment $\xi(P, n)$ of the quality of the estimate $\hat{\theta}(\mathcal{D})$, which consists of a summary of the distribution Q_n of some quantity $u(\mathcal{D}, P)$. Our choice of summary and form for u depends upon our inferential goals and our knowledge of the properties of $\hat{\theta}$. For instance, $\xi(P, n)$ might compute an interquantile range for $u(\mathcal{D}, P) = \hat{\theta}(\mathcal{D})$, the expectation of $u(\mathcal{D}, P) = \hat{\theta}(\mathcal{D}) - \theta(P)$ (i.e., the bias), or a confidence interval based on the distribution of $u(\mathcal{D}, P) = n^{1/2}(\hat{\theta}(\mathcal{D}) - \theta(P))$. Unfortunately, we cannot compute $\xi(P, n)$ directly because P and Q_n are unknown, and so we must resort to estimating $\xi(P, n)$ based upon a single observed dataset $\mathcal{D}$.

The bootstrap addresses this problem by estimating the unknown $\xi(P, n)$ via the plug-in approximation $\xi(\mathbb{P}_n, n)$. Although computing $\xi(\mathbb{P}_n, n)$ exactly is typically intractable, we

can obtain an accurate approximation using a simple Monte Carlo procedure: repeatedly form simulated datasets $\mathcal{D}^*$ of size n by sampling n points i.i.d. from $\mathbb{P}_n$, compute $u(\mathcal{D}^*, \mathbb{P}_n)$ for each simulated dataset, form the empirical distribution $\mathbb{Q}_n$ of the computed values of u, and return the desired summary of this distribution. We overload notation somewhat by referring to this final bootstrap output as $\xi(\mathbb{Q}_n, n)$, allowing ξ to take as its first argument either a data generating distribution or a distribution of u values.

For ease of exposition, we assume below that ξ is real-valued, though the proposed methodology can be straightforwardly generalized (e.g., to contexts in which ξ produces elements of a vector space).

3.3 The Diagnostic

We frame the task of evaluating whether or not the bootstrap is performing satisfactorily in a given setting as a decision problem: for a given estimator, data generating distribution P, and dataset size n, is the bootstrap's output sufficiently likely to be sufficiently near the ground truth value $\xi(P, n)$? This formulation avoids the difficulty of producing uniformly precise quantifications of the bootstrap's accuracy by requiring only that a decision be rendered based upon some definition of "sufficiently likely" and "sufficiently near the ground truth." Nonetheless, in developing a diagnostic procedure to address this decision problem, we face the key difficulties of determining the distribution of the bootstrap's outputs on datasets of size n and of obtaining even an approximation to the ground truth value against which to evaluate this distribution.

Ideally, we might approximate $\xi(P, n)$ for a given value of n by observing many independent datasets, each of size n. For each dataset, we would compute the corresponding value of u, and the resulting collection of u values would approximate the distribution Q_n, which would in turn yield a direct approximation of the ground truth value $\xi(P, n)$. Furthermore, we could approximate the distribution of bootstrap outputs by simply running the bootstrap on each dataset of size n. Unfortunately, however, in practice we only observe a single set of n data points, rendering this approach an unachievable ideal.

To surmount this difficulty, our diagnostic (Algorithm 3) executes this ideal procedure for dataset sizes smaller than n. That is, for a given $p \in \mathcal{N}$ and $b \leq \lfloor n/p \rfloor$, we randomly sample p disjoint subsets of the observed dataset $\mathcal{D}$, each of size b. For each subset, we compute the value of u; the resulting collection of u values approximates the distribution Q_b, in turn yielding a direct approximation of $\xi(P, b)$, the ground truth value for the smaller dataset size b. Additionally, we run the bootstrap on each of the p subsets of size b, and comparing the distribution of the resulting p bootstrap outputs to our ground truth approximation, we can determine whether or not the bootstrap performs acceptably well at sample size b.

It then remains to use this ability to evaluate the bootstrap's performance at smaller sample sizes to determine whether or not it is performing satisfactorily at the full sample size n. To that end, we evaluate the bootstrap's performance at multiple smaller sample sizes to determine whether or not the distribution of its outputs is in fact converging to the

Algorithm 3: Bootstrap Performance Diagnostic

Input: $\mathcal{D} = (X_1, \ldots, X_n)$: observed data
ξ: estimator quality assessment
u: quantity whose distribution is summarized by ξ
p: number of disjoint subsamples used to compute ground truth approximations (e.g., 100)
$b_1, \ldots, b_k$: increasing sequence of subsample sizes for which ground truth approximations are computed, with $b_k \leq \lfloor n/p \rfloor$ (e.g., $b_i = \lfloor n/(p2^{k-i}) \rfloor$ with $k = 3$)
$c_1 \geq 0$: tolerance for decreases in absolute relative deviation of mean bootstrap output (e.g., 0.2)
$c_2 \geq 0$: tolerance for decreases in relative standard deviation of bootstrap output (e.g., 0.2)
$c_3 \geq 0, \alpha \in [0, 1]$: desired probability α that bootstrap output at sample size n has absolute relative deviation from ground truth less than or equal to c_3 (e.g., $c_3 = 0.5, \alpha = 0.95$)

Output: *true* if bootstrap is deemed to be performing satisfactorily, *false* otherwise

$\mathbb{P}_n \leftarrow n^{-1} \sum_{i=1}^{n} \delta_{X_i}$
for $i \leftarrow 1$ **to** k **do**
  $\mathcal{D}_{i1}, \ldots, \mathcal{D}_{ip} \leftarrow$ random disjoint subsets of $\mathcal{D}$, each containing b_i data points
  for $j \leftarrow 1$ **to** p **do**
    $u_{ij} \leftarrow u(\mathcal{D}_{ij}, \mathbb{P}_n)$
    $\xi^*_{ij} \leftarrow bootstrap(\xi, u, b_i, \mathcal{D}_{ij})$
  end
  `// Compute ground truth approximation for sample size` b_i
  $\mathbb{Q}_{b_i} \leftarrow \sum_{j=1}^{p} \delta_{u_{ij}}$
  $\tilde{\xi}_i \leftarrow \xi(\mathbb{Q}_{b_i}, b_i)$
  `// Compute absolute relative deviation of mean of bootstrap outputs`
  `// and relative standard deviation of bootstrap outputs for sample`
  `// size` b_i
  $\Delta_i \leftarrow \left| \frac{mean(\xi^*_{i1}, \ldots, \xi^*_{ip}) - \tilde{\xi}_i}{\tilde{\xi}_i} \right|$ $\qquad$ $\sigma_i \leftarrow \left| \frac{stddev(\xi^*_{i1}, \ldots, \xi^*_{ip})}{\tilde{\xi}_i} \right|$
end
return true *if all of the following hold, and* false *otherwise:*

$$\Delta_{i+1} < \Delta_i \quad OR \quad \Delta_{i+1} \leq c_1, \quad \forall i = 1, \ldots, k, \tag{3.1}$$

$$\sigma_{i+1} < \sigma_i \quad OR \quad \sigma_{i+1} \leq c_2, \quad \forall i = 1, \ldots, k, \tag{3.2}$$

$$\frac{\#\left\{ j \in 1, \ldots, p : \left| \frac{\xi^*_{kj} - \tilde{\xi}_k}{\tilde{\xi}_k} \right| \leq c_3 \right\}}{p} \geq \alpha \tag{3.3}$$

ground truth as the sample size increases, thereby allowing us to generalize our conclusions regarding performance from smaller to larger sample sizes. Indeed, determining whether or not the bootstrap is performing satisfactorily for a single smaller sample size b alone is inadequate for our purposes, as the bootstrap's performance may degrade as sample size increases, so that it fails at sample size n despite appearing to perform sufficiently well at smaller sample size b. Conversely, the bootstrap may exhibit mediocre performance for small sample sizes but improve as it is applied to more data.

Thus, our diagnostic compares the distribution of bootstrap outputs to the ground truth approximation for an increasing sequence of sample sizes $b_1, \ldots, b_k$, with $b_k \leq \lfloor n/p \rfloor$; subsamples of each of these sizes are constructed and processed in the outer for loop of Algorithm 3. In order to conclude that the bootstrap is performing satisfactorily at sample size n, the diagnostic requires that the distribution of its outputs converges monotonically to the ground truth approximation for all of the smaller sample sizes $b_1, \ldots, b_k$. Convergence is assessed based on absolute relative deviation of the mean of the bootstrap outputs from the ground truth approximation (which must decrease with increasing sample size), and size of the standard deviation of the bootstrap outputs relative to the ground truth approximation (which must also decrease with increasing sample size). In Algorithm 3, this convergence assessment is performed by conditions (3.1) and (3.2). As a practical matter, these conditions do not require continuing decreases in the absolute relative mean deviation Δ_i or relative standard deviation σ_i when these quantities are below some threshold (given by c_1 and c_2, respectively) due to inevitable stochastic error in their estimation: when these quantities are sufficiently small, stochastic error due to the fact that we have only used p subsamples prevents reliable determination of whether or not decreases are in fact occurring. We have found that $c_1 = c_2 = 0.2$ is a reasonable choice of the relevant thresholds.

Progressive convergence of the bootstrap's outputs to the ground truth is not alone sufficient, however; although the bootstrap's performance may be improving as sample size increases, a particular value of n may not be sufficiently large to yield satisfactory performance. Therefore, beyond the convergence assessment discussed above, we must also determine whether or not the bootstrap is in fact performing sufficiently well for the user's purposes at sample size n. We define "sufficiently well" as meaning that with probability at least $\alpha \in [0, 1]$, the output of the bootstrap when run on a dataset of size n will have absolute relative deviation from ground truth of at most c_3 (the absolute relative deviation of a quantity γ from a quantity γ_o is defined as $|\gamma - \gamma_o|/|\gamma_o|$); the constants α and c_3 are specified by the user of the diagnostic procedure based on the user's inferential goals. Because we can only directly evaluate the bootstrap's performance at smaller sample sizes (and not at the full sample size n), we take a conservative approach, motivated by the assumption that a false positive (incorrectly concluding that the bootstrap is performing satisfactorily) is substantially less desirable than a false negative. In particular, as embodied in condition (3.3) of Algorithm 3, we require that the bootstrap is performing sufficiently well under the aforementioned definition at the sample size b_k. Satisfying this condition, in conjunction with satisfying the preceding conditions indicating continuing convergence to the ground truth, is taken to imply that the bootstrap will continue to perform satisfactorily when applied to the

full sample size n (in fact, the bootstrap's performance at sample size n will likely exceed that implied by α and c_3 due to the diagnostic's conservatism).

It is worth noting that this diagnostic procedure reposes on the availability in modern data analysis of both substantial quantities of data and substantial computational resources. For example, with $p = 100$ (an empirically effective choice), using $b_k = 1,000$ or $b_k = 10,000$ requires $n \geq 10^5$ or $n \geq 10^6$, respectively. Fortuitously, datasets of such sizes are now commonplace. Regarding its computational requirements, our procedure benefits from the modern shift toward parallel and distributed computing, as the vast majority of the required computation occurs in the inner for loop of Algorithm 3, the iterations of which are independent and individually process only small data subsets. Additionally, we have sought to reduce the procedure's computational costs by using an identical number of subsamples p for each subsample size $b_1, \ldots, b_k$; one could presumably improve statistical performance by using larger numbers of subsamples for smaller subsample sizes.

The guidelines given in Algorithm 3 for setting the diagnostic procedure's hyperparameters are motivated by the procedure's structure and have proven to be empirically effective. We recommend exponential spacing of the $b_1, \ldots, b_k$ to help ensure that reliable comparisons of bootstrap performance can be made across adjacent sample sizes b_i and b_{i+1}. However, by construction, setting the $b_1, \ldots, b_k$ to be too close together should primarily cause an increase in the false negative rate (the probability that the diagnostic incorrectly concludes that the bootstrap is not performing satisfactorily), rather than a less desirable increase in the false positive rate. Similarly, setting c_1 or c_2 to be too low should also primarily result in an increase in the false negative rate. Regarding c_3 and α, these hyperparameters should be determined by the user's bootstrap performance desiderata. We nonetheless expect that fairly lenient settings of c_3—such as $c_3 = 0.5$, which corresponds to allowing the bootstrap to deviate from ground truth by up to 50%—to be reasonable in many cases. This expectation stems from the fact that the actual or targeted quality of estimators on fairly large datasets is frequently high, leading to estimator quality assessments, such as interquantile ranges, which are small in absolute value; in these cases, it follows that a seemingly large relative error in bootstrap outputs (e.g., 50%) corresponds to a small absolute error.

As we demonstrate via an extensive empirical evaluation on both synthetic and real data in the following two sections, our proposed bootstrap performance diagnostic is quite effective, with false positive rates that are generally extremely low or zero and false negative rates that generally approach zero as the subsample sizes $b_1, \ldots, b_k$ are increased. Of course, like any inferential procedure, our procedure does have some unavoidable limitations, such as in cases where the data generating distribution has very fine-grained adverse features which cannot be reliably observed in datasets of size b_k; we discuss these issues further below.

3.4 Simulation Study

We first evaluate the diagnostic's effectiveness on data generated from a variety of different synthetic distributions paired with a variety of different estimators. Using simulated data

here allows direct knowledge of the ground truth value $\xi(P, n)$, and by selecting different synthetic distributions, we can design settings that pose different challenges to the diagnostic procedure. For each distribution-estimator pair and sample size n considered, we perform multiple independent runs of the diagnostic on independently generated datasets of size n to compute the Diagnostic True Rate (DTR), the probability that the diagnostic outputs *true* in that setting. We then evaluate this DTR against the bootstrap's actual performance on datasets of size n; because the underlying data generating distributions here are known, we can also compare to known theoretical expectations of bootstrap consistency.

More precisely, we consider the following data generating distributions: Normal(0,1), Uniform(0,10), StudentT(1.5), StudentT(3), Cauchy(0,1), 0.95Normal(0,1) + 0.05Cauchy(0,1), and 0.99Normal(0,1) + 0.01Cauchy(10^4,1). In our plots, we denote these distributions using the following abbreviations: Normal, Uniform, StuT(1.5), StuT(3), Cauchy, Mixture1, and Mixture2. We also consider the following estimators $\hat{\theta}$ (abbreviations, if any, are given in parentheses): mean, median (med), variance (var), standard deviation (std), sample maximum (max), and 95th percentile (perc). The estimator quality assessment ξ in all experiments computes the interquantile range between the 0.025 and 0.975 quantiles of the distribution of $u(\mathcal{D}, P) = \hat{\theta}(\mathcal{D})$. For all runs of the bootstrap, we use between 200 and 500 resamples, with the precise number of resamples determined by the adaptive hyperparameter selection procedure given in Section 2.5 above. All runs of the diagnostic use the hyperparameter guidelines given in Algorithm 3: $p = 100, k = 3, b_i = \lfloor n/(p2^{k-i}) \rfloor, c_1 = 0.2, c_2 = 0.2, c_3 = 0.5, \alpha = 0.95$. We consider sample sizes $n = 10^5$ and $n = 10^6$.

For each distribution-estimator pair and sample size n, we first compute the ground truth value $\xi(P, n)$ as the interquantile range of the u values for 5,000 independently generated datasets of size n. We also approximate the distribution of bootstrap outputs on datasets of size n by running the bootstrap on 100 independently generated datasets of this size. Whether or not this distribution of bootstrap outputs satisfies the performance criterion defined by c_3, α—that is, whether or not the α quantile of the absolute relative deviation of bootstrap outputs from $\xi(P, n)$ is less than or equal to c_3—determines the ground truth conclusion regarding whether or not the bootstrap is performing satisfactorily in a given setting. To actually evaluate the diagnostic's effectiveness, we then run it on 100 independently generated datasets of size n and estimate the DTR as the fraction of these datasets for which the diagnostic returns *true*. If the ground truth computations deemed the bootstrap to be performing satisfactorily in a given setting, then the DTR would ideally be 1, and otherwise it would ideally be 0.

Figure 3.1 presents our results for all distribution-estimator pairs and both sample sizes n considered. In these plots, dark blue indicates cases in which the ground truth computations on datasets of size n deemed the bootstrap to be performing satisfactorily and the bootstrap is expected theoretically to be consistent (i.e., the DTR should ideally be 1); red indicates cases in which neither of these statements is true (i.e., the DTR should ideally be 0); and light purple indicates cases in which the ground truth computations on datasets of size n deemed the bootstrap to be performing satisfactorily but the bootstrap is not expected theoretically to be consistent (i.e., the DTR should ideally be 1).

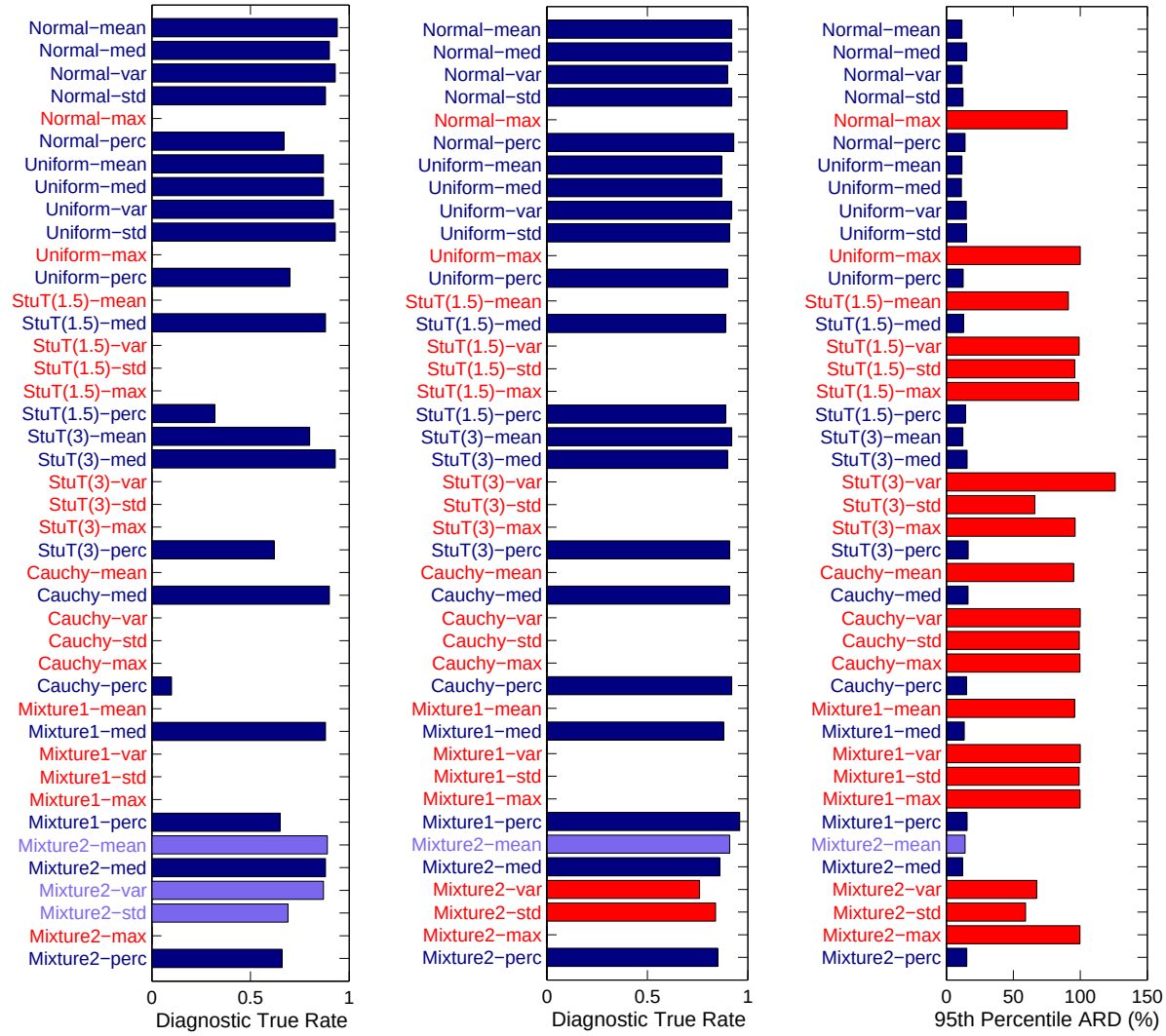


Figure 3.1: Diagnostic and bootstrap performance on simulated data. Dark blue indicates cases where bootstrap is performing satisfactorily on datasets of size n (based on ground truth computations) and is expected theoretically to be consistent; red indicates cases where neither of these statements is true; light purple indicates cases where bootstrap is performing satisfactorily on datasets of size n (based on ground truth computations) but is not expected theoretically to be consistent. **(left and middle)** For each distribution-estimator pair, fraction of 100 independent trials for which the diagnostic outputs *true*. For the left plot, $n = 10^5$; for the middle plot, $n = 10^6$. **(right)** For each distribution-estimator pair, 95th percentile of absolute relative deviation of bootstrap output from ground truth, over 100 independent trials on datasets of size $n = 10^6$.

As seen in the lefthand and middle plots (which show DTRs for $n = 10^5$ and $n = 10^6$, respectively), our proposed diagnostic performs quite well across a range of data generating distributions and estimators, and its performance improves as it is provided with more data. For the smaller sample size $n = 10^5$, in the dark blue and light purple cases, the DTR is generally markedly greater than 0.5; furthermore, when the sample size is increased to $n = 10^6$, the DTRs in all of the dark blue and light purple cases increase to become uniformly near 1, indicating low false negative rates (i.e., the diagnostic nearly always deems the bootstrap to be performing satisfactorily when it is indeed performing satisfactorily). In the red cases, for both sample sizes, the DTR is nearly always zero, indicating that false positive rates are nearly always zero (i.e., the diagnostic only rarely deems the bootstrap to be performing satisfactorily when it is in fact not performing satisfactorily). Mixture2-var and Mixture2-std with $n = 10^6$ provide the only exceptions to this result, which is unsurprising given that Mixture2 was specifically designed to include a small heavy-tailed component which is problematic for the bootstrap but cannot be reliably detected at the smaller sample sizes $b_1, \ldots, b_k$; nonetheless, even in these cases, the righthand plot indicates that the ground truth computations very nearly deemed the bootstrap to be performing satisfactorily. Interestingly, the bootstrap's finite sample performance for the settings considered nearly always agrees with theoretical expectations regarding consistency; disagreement occurs only when Mixture2 is paired with the estimators mean, var, or std, which is again unsurprising given the properties of Mixture2.

3.5 Real Data

We also evaluate the diagnostic's effectiveness on three real datasets obtained from Conviva, Inc. [23], which describe different attributes of large numbers of video streams viewed by Internet users. These datasets are routinely subjected to a variety of different analyses by practitioners and are the subject of ongoing efforts to improve the computational efficiency of database systems by processing only data subsamples and quantifying the resulting estimation error [2].

We designate the three (scalar-valued) datasets as follows, with their sizes (i.e., numbers of constituent data points) given in parentheses: Conviva1 (30,470,092), Conviva2 (1,111,798,565), and Conviva3 (2,952,651,449). Histograms of the three datasets are given in Figure 3.2; note that the datasets are heavily skewed and also contain large numbers of repeated values. Due to privacy considerations, we are unable to provide the precise values and corresponding frequencies represented in the data, but the histograms nonetheless convey the shapes of the datasets' empirical distributions.

To circumvent the fact that ground truth values for individual real datasets cannot be obtained, we do not directly apply our diagnostic to these three datasets. Rather, we treat the empirical distribution of each dataset as an underlying data generating distribution which is used to generate the datasets used in our experiments. With this setup, our experiments on the real datasets proceed identically to the experiments in Section 3.4 above, but now with

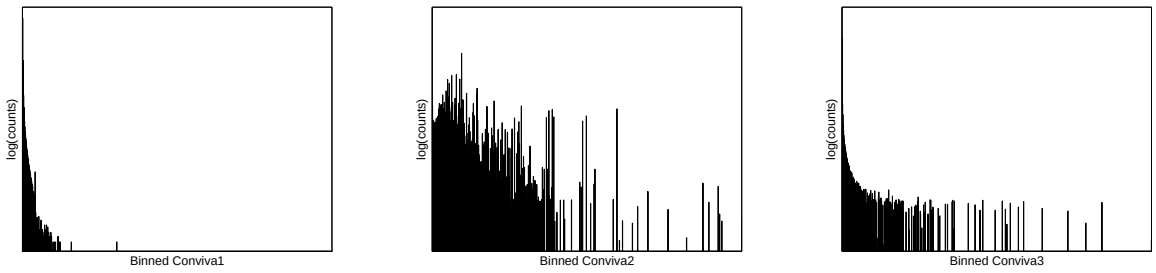


Figure 3.2: Histograms for the real datasets Conviva1, Conviva2, and Conviva3. Note that the y axes give frequencies on a log scale.

data sampled from the aforementioned empirical distributions rather than from synthetic distributions.

Figure 3.3 presents the results of our experiments on the Conviva data. The color scheme used in these plots is identical to that in Figure 3.1, with the addition of magenta, which indicates cases in which the ground truth computations on datasets of size n deemed the bootstrap to not be performing satisfactorily but the bootstrap is expected theoretically to be consistent (i.e., the DTR should ideally be 0). Given that the data generating distributions used in these experiments all have finite support, the bootstrap is expected theoretically to be consistent for all estimators considered except the sample maximum. However, as seen in the righthand plot of Figure 3.3, the bootstrap's finite sample performance is often quite poor even in cases where consistency is expected; in this regard (as well as in other ways), the real data setting of this section differs substantially from the synthetic data setting considered in Section 3.4 above.

The lefthand and middle plots of Figure 3.3 demonstrate that our diagnostic procedure again performs quite well. Indeed, the DTR is again nearly always zero (or is quite small if positive) in the red and magenta cases, indicating false positive rates that are nearly always zero. The dark blue cases generally have DTRs markedly greater than 0.5 for $n = 10^5$ (lefthand plot), with DTRs in these cases generally increasing to become nearly 1 for $n = 10^6$, indicating low false negative rates; no light purple cases occur for the real data. Beyond these broad conclusions, it is worth noting that the Conviva2-max, Conviva2-perc, and Conviva3-med settings exhibit rather surprising behavior relative to our other results, in that the diagnostic's performance seems to degrade when the sample size is increased. We believe that this behavior is related to the particularly high redundancy (i.e., degree of repetition of values) in Conviva2 and Conviva3, and it will be the subject of future work.

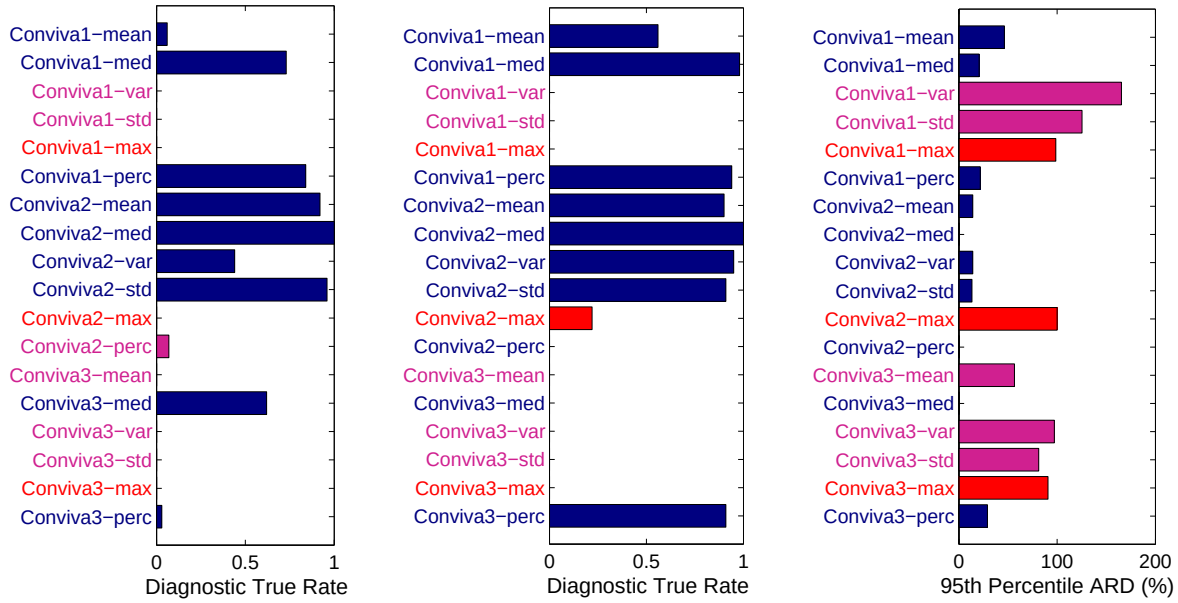


Figure 3.3: Diagnostic and bootstrap performance on real data. Color scheme is identical to that in Figure 3.1, with the addition of magenta indicating cases where the bootstrap is not performing satisfactorily on datasets of size n (based on ground truth computations) but is expected theoretically to be consistent. **(left and middle)** For each distribution-estimator pair, fraction of 100 independent trials for which the diagnostic outputs *true*. For the left plot, $n = 10^5$; for the middle plot, $n = 10^6$. **(right)** For each distribution-estimator pair, 95th percentile of absolute relative deviation of bootstrap output from ground truth, over 100 independent trials on datasets of size $n = 10^6$.

Chapter 4

Random Conic Pursuit for Semidefinite Programming

4.1 Introduction

Many difficult problems have been shown to admit elegant and tractably computable representations via optimization over the set of positive semidefinite (PSD) matrices. As a result, semidefinite programs (SDPs) have appeared as the basis for many procedures in machine learning and statistics, such as sparse PCA [25], distance metric learning [80], nonlinear dimensionality reduction [78], multiple kernel learning [51], multitask learning [64], and matrix completion [15].

While SDPs can be solved in polynomial time, they remain computationally challenging. General-purpose solvers, often based on interior point methods, do exist and readily provide high-accuracy solutions. However, their memory requirements do not scale well with problem size, and they typically do not allow a fine-grained tradeoff between optimization accuracy and speed, which is often a desirable tradeoff in machine learning and statistical problems that are based on random data. Furthermore, SDPs in machine learning and statistics frequently arise as convex relaxations of problems that are originally computationally intractable, in which case even an exact solution to the SDP yields only an approximate solution to the original problem, and an approximate SDP solution can once again be quite useful. Although some SDPs do admit tailored solvers which are fast and scalable (e.g., [59, 17, 24]), deriving and implementing these methods is often challenging, and an easily usable solver that alleviates these issues has been elusive. This is partly the case because generic first-order methods do not apply readily to general SDPs.

In this work, we present Random Conic Pursuit, a randomized solver for general SDPs that is simple, easily implemented, scalable, and of inherent interest due to its novel construction. We consider general SDPs over $\mathbb{R}^{d\times d}$ of the form

$$\min_{X\succeq 0} \quad f(X) \quad \text{s.t.} \quad g_j(X) \leq 0, \qquad j = 1\ldots k, \tag{4.1}$$

where f and the g_j are convex real-valued functions, and $\succeq$ denotes the ordering induced by the PSD cone. Random Conic Pursuit minimizes the objective function iteratively, repeatedly randomly sampling a PSD matrix and optimizing over the random two-dimensional subcone given by this matrix and the current iterate. This construction maintains feasibility while avoiding the computational expense of deterministically finding feasible directions or of projecting into the feasible set. Furthermore, each iteration is computationally inexpensive, though in exchange we generally require a relatively large number of iterations. In this regard, Random Conic Pursuit is similar in spirit to algorithms such as online gradient descent and sequential minimal optimization [65] which have illustrated that in the machine learning setting, algorithms that take a large number of simple, inexpensive steps can be surprisingly successful.

The resulting algorithm, despite its simplicity and randomized nature, converges fairly quickly to useful approximate solutions. Unlike interior point methods, Random Conic Pursuit does not excel in producing highly exact solutions. However, it is more scalable and provides the ability to trade off computation for more approximate solutions. In what follows, we present our algorithm in full detail and demonstrate its empirical behavior and efficacy on various SDPs that arise in machine learning; we also provide analytical results that yield insight into its behavior and convergence properties.

4.2 Random Conic Pursuit

Random Conic Pursuit (Algorithm 4) solves SDPs of the general form (4.1) via a sequence of simple two-variable optimizations (4.2). At each iteration, the algorithm considers the two-dimensional cone spanned by the current iterate, X_t, and a random rank one PSD matrix, Y_t. It selects as its next iterate, X_{t+1}, the point in this cone that minimizes the objective f subject to the constraints $g_j(X_{t+1}) \leq 0$ in (4.1). The distribution of the random matrices is periodically updated based on the current iterate (e.g., to match the current iterate in expectation); these updates yield random matrices that are better matched to the optimum of the SDP at hand.

The two-variable optimization (4.2) can be solved quickly in general via a two-dimensional bisection search. As a further speedup, for many of the problems that we considered, the two-variable optimization can be altogether short-circuited with a simple check that determines whether the solution $X_{t+1} = X_t$, with $\hat{\beta} = 1$ and $\hat{\alpha} = 0$, is optimal. Additionally, SDPs with a trace constraint $\operatorname{tr} X = 1$ force $\alpha + \beta = 1$ and therefore require only a one-dimensional optimization.

Random Conic Pursuit can also readily benefit from the use of parallel and distributed computational resources (though our experiments below are all performed in the serial single-processor setting). In particular, during each iteration, we can use multiple processors to simultaneously draw multiple different instantiations of Y_t and select that which yields the greatest decrease in the objective function; this approach can potentially yield substantial benefits, particularly during later iterations in which finding feasible descent directions is

Algorithm 4: Random Conic Pursuit
[brackets contain a particular, generally effective, sampling scheme]

Input: A problem of the form (4.1) $n \in \mathbb{N}$: number of iterations
X_0: a feasible initial iterate [$\kappa \in (0,1)$: numerical stability parameter]
Output: An approximate solution X_n to (4.1)

$p \leftarrow$ a distribution over $\mathbb{R}^d$ [$p \leftarrow \mathcal{N}(0, \Sigma)$ with $\Sigma = (1-\kappa)X_0 + \kappa I_d$]
for $t \leftarrow 1$ **to** n **do**
  Sample x_t from p and set $Y_t \leftarrow x_t x_t'$
  Set $\hat{\alpha}, \hat{\beta}$ to the optimizer of

$$\begin{aligned} \min_{\alpha,\beta \in \mathbb{R}} \; & f(\alpha Y_t + \beta X_{t-1}) \\ \text{s.t. } & g_j(\alpha Y_t + \beta X_{t-1}) \leq 0, \quad j = 1 \dots k \\ & \alpha, \beta \geq 0 \end{aligned} \tag{4.2}$$

  Set $X_t \leftarrow \hat{\alpha} Y_t + \hat{\beta} X_{t-1}$
  if $\hat{\alpha} > 0$ **then** Update p based on X_t [$p \leftarrow \mathcal{N}(0, \Sigma)$ with $\Sigma = (1-\kappa)X_t + \kappa I_d$]
end
return X_n

more difficult so that often $\hat{\alpha} = 0$. Furthermore, optimization of (4.2), including the underlying basic matrix operations (which are usually simple operations such as matrix additions or inner products rather than more complicated operations such as inversions), can often be straightforwardly parallelized.

Two simple guarantees for Random Conic Pursuit are immediate. First, its iterates are feasible for (4.1) because each iterate is a positive sum of two PSD matrices, and because the constraints g_j of (4.2) are also those of (4.1). Second, the objective values decrease monotonically because $\beta = 1, \alpha = 0$ is a feasible solution to (4.2). We must also note two limitations of Random Conic Pursuit: it does not admit general equality constraints, and it requires a feasible starting point. Nonetheless, for many of the SDPs that appear in machine learning and statistics, feasible points are easy to identify, and equality constraints are either absent or fortuitously pose no difficulty.

We can gain further intuition by observing that Random Conic Pursuit's iterates, X_t, are positive weighted sums of random rank one matrices and so lie in the random polyhedral cones

$$\mathcal{F}_t^x := \left\{ \sum_{i=1}^{t} \gamma_i x_t x_t' : \gamma_i \geq 0 \right\} \subset \{X : X \succeq 0\}. \tag{4.3}$$

Thus, Random Conic Pursuit optimizes the SDP (4.1) by greedily optimizing f with respect to the g_j constraints within an expanding sequence of random cones $\{\mathcal{F}_t^x\}$. These cones yield successively better inner approximations of the PSD cone (a basis for which is the set of all

rank one matrices) while allowing us to easily ensure that the iterates remain PSD.

In light of this discussion, one might consider approximating the original SDP by sampling a random cone $\mathcal{F}_n^x$ in one shot and replacing the constraint $X \succeq 0$ in (4.1) with the simpler linear constraints $X \in \mathcal{F}_n^x$. For sufficiently large n, $\mathcal{F}_n^x$ would approximate the PSD cone well (see Theorem 5 below), yielding an inner approximation that upper bounds the original SDP; the resulting problem would be easier than the original (e.g., it would become a linear program if the g_j were linear). However, we have found empirically that a very large n is required to obtain good approximations, thus negating any potential performance improvements (e.g., over interior point methods). Random Conic Pursuit successfully resolves this issue by iteratively expanding the random cone $\mathcal{F}_t^x$. As a result, we are able to much more efficiently access large values of n, though we compute a greedy solution within $\mathcal{F}_n^x$ rather than a global optimum over the entire cone. This tradeoff is ultimately quite advantageous.

4.3 Applications and Experiments

We assess the practical convergence and scaling properties of Random Conic Pursuit by applying it to three different machine learning and statistical tasks that rely on SDPs: distance metric learning, sparse PCA, and maximum variance unfolding. For each, we compare the performance of Random Conic Pursuit (implemented in MATLAB) to that of a standard and widely used interior point solver, `SeDuMi` [71] (via `cvx` [40]), and to the best available solver which has been customized for each problem.

To evaluate convergence, we first compute a ground-truth solution X^* for each problem instance by running the interior point solver with extremely low tolerance. Then, for each algorithm, we plot the normalized objective value errors $[f(X_t) - f(X^*)]/|f(X^*)|$ of its iterates X_t as a function of the amount of time required to generate each iterate. Additionally, for each problem, we plot the value of an application-specific metric for each iterate. These metrics provide a measure of the practical implications of obtaining SDP solutions which are suboptimal to varying degrees. We evaluate scaling with problem dimensionality by running the various solvers on problems of different dimensionalities and computing various metrics on the solver runs as described below for each experiment. Unless otherwise noted, we use the bracketed sampling scheme given in Algorithm 4 with $\kappa = 10^{-4}$ for all runs of Random Conic Pursuit.

4.3.1 Metric Learning

Given a set of datapoints in $\mathbb{R}^d$ and a pairwise similarity relation over them, metric learning extracts a Mahalanobis distance $d_A(x, y) = \sqrt{(x-y)'A(x-y)}$ under which similar points are nearby and dissimilar points are far apart [80]. Let $\mathcal{S}$ be the set of similar pairs of datapoints, and let $\bar{\mathcal{S}}$ be its complement. The metric learning SDP, for $A \in \mathbb{R}^{d \times d}$ and

$C = \sum_{(i,j)\in\mathcal{S}}(x_i - x_j)(x_i - x_j)'$, is

$$\min_{A\succeq 0} \quad \text{tr}(CA) \quad \text{s.t.} \quad \sum_{(i,j)\in\bar{\mathcal{S}}} d_A(x_i, x_j) \geq 1. \tag{4.4}$$

To apply Random Conic Pursuit, X_0 is set to a feasible scaled identity matrix. We solve the two-variable optimization (4.2) via a double bisection search: at each iteration, α is optimized out with a one-variable bisection search over α given fixed β, yielding a function of β only. This resulting function is itself then optimized using a bisection search over β.

As the application-specific metric for this problem, we measure the extent to which the metric learning goal has been achieved: similar datapoints should be near each other, and dissimilar datapoints should be farther away. We adopt the following metric of quality of a solution matrix X, where $\zeta = \sum_i |\{j : (i,j) \in \mathcal{S}\}| \cdot |\{l : (i,l) \in \bar{\mathcal{S}}\}|$ and $1[\cdot]$ is the indicator function: $Q(X) = \frac{1}{\zeta}\sum_i \sum_{j:(i,j)\in\mathcal{S}} \sum_{l:(i,l)\in\bar{\mathcal{S}}} 1[d_{ij}(X) < d_{il}(X)]$.

To examine convergence behavior, we first apply the metric learning SDP to the UCI ionosphere dataset, which has $d = 34$ and 351 datapoints with two distinct labels ($\mathcal{S}$ contains pairs with identical labels). We selected this dataset from among those used in [80] because it is among the datasets which have the largest dimensionality and experience the greatest impact from metric learning in that work's clustering application. Because the interior point solver scales prohibitively badly in the number of datapoints, we subsampled the dataset to yield $4 \times 34 = 136$ datapoints.

To evaluate scaling, we use synthetic data in order to allow variation of d. To generate a d-dimensional data set, we first generate mixture centers by applying a random rotation to the elements of $\mathcal{C}_1 = \{(-1, 1), (-1, -1)\}$ and $\mathcal{C}_2 = \{(1, 1), (1, -1)\}$. We then sample each datapoint $x_i \in \mathbb{R}^d$ from $\mathcal{N}(0, I_d)$ and assign it uniformly at random to one of two clusters. Finally, we set the first two components of x_i to a random element of $\mathcal{C}_k$ if x_i was assigned to cluster $k \in \{1, 2\}$; these two components are perturbed by adding a sample from $\mathcal{N}(0, 0.25I_2)$.

The best known customized solver for the metric learning SDP is a projected gradient algorithm [80], for which we used code available from the author's website.

Figure 4.1 shows the results of our experiments on an ionosphere data problem instance. The two trajectory plots show that Random Conic Pursuit converges to a very high-quality solution (with high Q and negligible objective value error) significantly faster than interior point. Additionally, our performance is comparable to that of the projected gradient method which has been customized for this task. Table 4.1 illustrates scaling for increasing d. Interior point scales badly in part because parsing the SDP becomes impracticably slow for d significantly larger than 100. Nonetheless, Random Conic Pursuit scales well beyond that point, continuing to return solutions with high Q in reasonable time. On this synthetic data, projected gradient appears to reach high Q somewhat more quickly, though Random Conic Pursuit consistently yields significantly better objective values, indicating better-quality solutions.

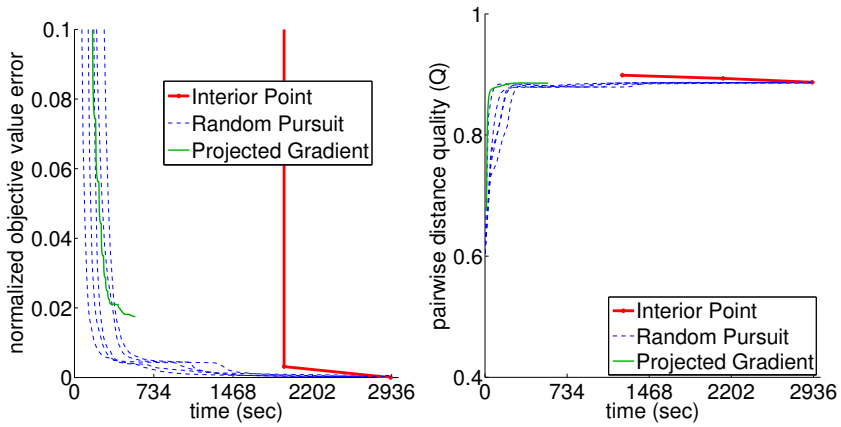


Figure 4.1: Results for metric learning on UCI ionosphere data: trajectories of objective value error (left) and Q (right).

d	alg	f after 2 hrs*	time to $Q > 0.99$
100	IP	3.7e-9	636.3
100	**RCP**	2.8e-7, 3.0e-7	142.7, 148.4
100	PG	1.1e-5	42.3
200	**RCP**	5.1e-8, 6.1e-8	529.1, 714.8
200	PG	1.6e-5	207.7
300	**RCP**	5.4e-8, 6.5e-8	729.1, 1774.7
300	PG	2.0e-5	1095.8
400	**RCP**	7.2e-8, 1.0e-8	2128.4, 2227.2
400	PG	2.4e-5	1143.3

Table 4.1: Results for metric learning scaling experiments on synthetic data (IP = interior point, RCP = Random Conic Pursuit, PG = projected gradient), with two trials per d for Random Conic Pursuit and times in seconds. *For $d = 100$, third column shows f after 20 minutes.

4.3.2 Sparse PCA

Sparse PCA seeks to find a sparse unit length vector that maximizes $x'Ax$ for a given data covariance matrix A. This problem can be relaxed to the following SDP [25], for $X, A \in \mathbb{R}^{d\times d}$:

$$\min_{X\succeq 0} \quad \rho\mathbf{1}'|X|\mathbf{1} - \mathrm{tr}(AX) \quad \text{s.t.} \quad tr(X) = 1, \tag{4.5}$$

where the scalar $\rho > 0$ controls the solution's sparsity. A subsequent rounding step returns the dominant eigenvector of the SDP's solution, yielding a sparse principal component.

We use the colon cancer dataset [3] that has been used frequently in past studies of sparse PCA and contains 2,000 microarray readings for 62 subjects. The goal is to identify a small number of microarray cells that capture the greatest variance in the dataset. We vary d by

subsampling the readings and use $\rho = 0.2$ (large enough to yield sparse solutions) for all experiments.

To apply Random Conic Pursuit, we set $X_0 = A/\operatorname{tr}(A)$. The trace constraint (4.5) implies that $\operatorname{tr}(X_{t-1}) = 1$ and so $\operatorname{tr}(\alpha Y_t + \beta X_{t-1}) = \alpha \operatorname{tr}(Y_t) + \beta = 1$ in (4.2). Thus, we can simplify the two-variable optimization (4.2) to a one-variable optimization, which we solve by bisection search.

The fastest available customized solver for the sparse PCA SDP is an adaptation of Nesterov's smooth optimization procedure [25] (denoted by DSPCA), for which we used a MATLAB implementation with heavy MEX optimizations that is downloadable from the author's web site.

We compute two application-specific metrics which capture the two goals of sparse PCA: high captured variance and high sparsity. Given the top eigenvector u of a solution matrix X, its captured variance is $u'Au$, and its sparsity is given by $\frac{1}{d}\sum_j 1[|u_j| < \tau]$; we take $\tau = 10^{-3}$ based on qualitative inspection of the raw microarray data covariance matrix A.

The results of our experiments are shown in Figure 4.2 and Table 4.2. As seen in the plots, on a problem instance with $d = 100$, Random Conic Pursuit quickly achieves an objective value within 4% of optimal and thereafter continues to converge, albeit more slowly; we also quickly achieve fairly high sparsity (compared to that of the exact SDP optimum). In contrast, interior point is able to achieve lower objective value and even higher sparsity within the timeframe shown, but, unlike Random Conic Pursuit, it does not provide the option of spending less time to achieve a solution which is still relatively sparse. All of the solvers quickly achieve very similar captured variances, which are not shown. DSPCA is extremely efficient, requiring much less time than its counterparts to find nearly exact solutions. However, that procedure is highly customized (via several pages of derivation and an optimized implementation), whereas Random Conic Pursuit and interior point are general-purpose.

Table 4.2 illustrates scaling by reporting achieved objecive values and sparsities after the solvers have each run for 4 hours. Interior point fails due to memory requirements for $d > 130$, whereas Random Conic Pursuit continues to function and provide useful solutions, as seen from the achieved sparsity values, which are much larger than those of the raw data covariance matrix. Again, DSPCA continues to be extremely efficient.

4.3.3 Maximum Variance Unfolding (MVU)

MVU searches for a kernel matrix that embeds high-dimensional input data into a lower-dimensional manifold [78]. Given m data points and a neighborhood relation $i \sim j$ between them, it forms their centered and normalized Gram matrix $G \in \mathbb{R}^{m \times m}$ and the squared Euclidean distances $d_{ij}^2 = G_{ii} + G_{jj} - 2G_{ij}$. The desired kernel matrix is the solution of the following SDP, where $X \in \mathbb{R}^{m \times m}$ and the scalar $\nu > 0$ controls the dimensionality of the

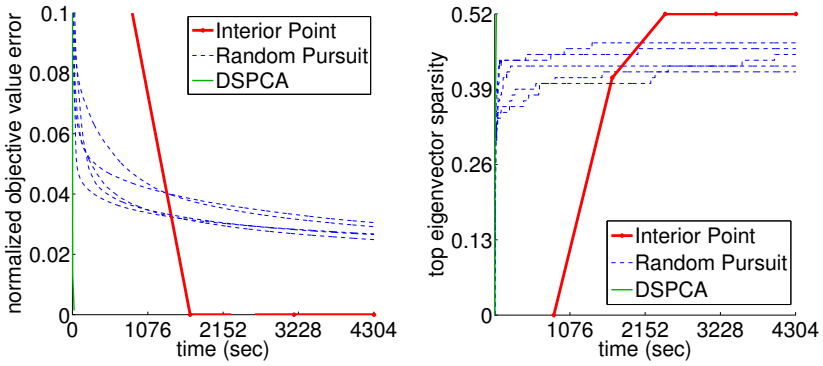


Figure 4.2: Results for sparse PCA: trajectories of objective value error (left) and sparsity (right), for a problem with $d = 100$. All solvers quickly yield similar captured variance (not shown here).

d	alg	f after 4 hrs	sparsity after 4 hrs
120	IP	-10.25	0.55
120	**RCP**	-9.98, -10.02	0.47, 0.45
120	DSPCA	-10.24	0.55
200	IP	failed	failed
200	**RCP**	-10.30, -10.27	0.51, 0.50
200	DSPCA	-11.07	0.64
300	IP	failed	failed
300	**RCP**	-9.39, -9.29	0.51, 0.51
300	DSPCA	-11.52	0.69
500	IP	failed	failed
500	**RCP**	-6.95, -6.54	0.53, 0.50
500	DSPCA	-11.61	0.78

Table 4.2: Results for sparse PCA scaling experiments (IP = interior point, RCP = Random Conic Pursuit), with two trials per d for Random Conic Pursuit.

resulting embedding:

$$\max_{X \succeq 0} \quad \operatorname{tr}(X) - \nu \sum_{i \sim j} (X_{ii} + X_{jj} - 2X_{ij} - d_{ij}^2)^2 \quad \text{s.t.} \quad \mathbf{1}'X\mathbf{1} = 0. \tag{4.6}$$

To apply Random Conic Pursuit, we set $X_0 = G$ and use the general sampling formulation in Algorithm 4 by setting $p = \mathcal{N}(0, \Pi(\nabla f(X_t)))$ in the initialization (i.e., $t = 0$) and update steps, where Π truncates negative eigenvalues of its argument to zero. This scheme empirically yields improved performance for the MVU problem as compared to the bracketed sampling scheme in Algorithm 4. To handle the equality constraint, each Y_t is first transformed to $\breve{Y}_t = (I - \mathbf{1}\mathbf{1}'/m)Y_t(I - \mathbf{1}\mathbf{1}'/m)$, which preserves PSDness and ensures feasibility. The two-variable optimization (4.2) proceeds as before on $\breve{Y}_t$ and becomes a two-variable quadratic program, which can be solved analytically.

MVU also admits a gradient descent algorithm, which serves as a straw-man large-scale solver for the MVU SDP. At each iteration, the step size is picked by a line search, and the spectrum of the iterate is truncated to maintain PSDness. We use G as the initial iterate.

To generate data, we randomly sample m points from the surface of a synthetic swiss roll [78]; we set $\nu = 1$. To quantify the amount of time required for a solver to converge, we run it until its objective curve appears qualitatively flat and declare the convergence point to be the earliest iterate whose objective value is within 1% of the best objective value seen so far (which we denote by $\hat{f}$).

Figure 4.3 and Table 4.3 illustrate that Random Conic Pursuit's objective values converge quickly, and on problems where the interior point solver achieves the optimum, Random Conic Pursuit nearly achieves that optimum. The interior point solver runs out of memory when $m > 400$ and also fails on smaller problems if its tolerance parameter is not tuned. Random Conic Pursuit easily runs on larger problems for which interior point fails, and for smaller problems its running time is within a small factor of that of the interior point solver. The gradient descent solver is orders of magnitude slower than the other solvers and failed to converge to a meaningful solution for $m \geq 400$ even after 2000 iterations (which took 8 hours).

4.4 Analysis

Analysis of Random Conic Pursuit is complicated by the procedure's use of randomness and its handling of the constraints $g_j \leq 0$ explicitly in the sub-problem (4.2), rather than via penalty functions or projections. Nonetheless, we are able to obtain useful insights by first analyzing a simpler setting having only a PSD constraint. We thus obtain a bound on the rate at which the objective values of Random Conic Pursuit's iterates converge to the SDP's optimal value when the problem has no constraints of the form $g_j \leq 0$:

Theorem 4 (Convergence rate of Random Conic Pursuit when f is weakly convex and $k = 0$)**.** *Let $f : \mathbb{R}^{d \times d} \to \mathbb{R}$ be a convex differentiable function with L-Lipschitz gradients such*

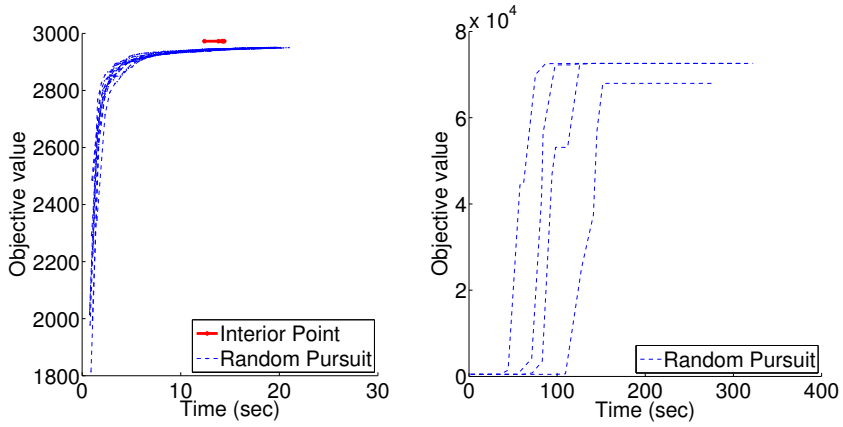


Figure 4.3: Results for MVU: trajectories of objective value for problems with $m = 200$ (left) and $m = 800$ (right).

m	alg	f after convergence	seconds to $f > 0.99\hat{f}$
40	IP	23.4	0.4
40	**RCP**	22.83 (0.03)	0.5 (0.03)
40	GD	23.2	5.4
200	IP	2972.6	12.4
200	**RCP**	2921.3 (1.4)	6.6 (0.8)
200	GD	2943.3	965.4
400	IP	12255.6	97.1
400	**RCP**	12207.96 (36.58)	26.3 (9.8)
800	IP	failed	failed
800	**RCP**	71231.1 (2185.7)	115.4 (29.2)

Table 4.3: Results for MVU scaling experiments showing convergence as a function of m (IP = interior point, RCP = Random Conic Pursuit, GD = gradient descent). Standard deviations over 10 runs of Random Conic Pursuit are shown in parentheses.

that the minimum of the following optimization problem is attained at some X^*:

$$\min_{X \succeq 0} \; f(X). \tag{4.7}$$

Let $X_1 \dots X_t$ be the iterates of Algorithm 4 when applied to this problem starting at iterate X_0 (using the bracketed sampling scheme given in the algorithm specification), and suppose that $\|X_t - X^*\|$ is bounded. Then,

$$Ef(X_t) - f(X^*) \leq \frac{1}{t} \cdot \max(\Gamma L, f(X_0) - f(X^*)), \tag{4.8}$$

for some constant Γ that does not depend on t.

See the appendix for proof. Despite the extremely simple and randomized nature of Random Conic Pursuit, the theorem guarantees that its objective values converge at the rate $O(1/t)$ on an important subclass of SDPs. We omit here some readily available extensions: for example, the probability that a trajectory of iterates violates the above rate can be bounded by noting that the iterates' objective values behave as a finite difference submartingale. Additionally, the theorem and proof could be generalized to hold for a broader class of sampling schemes.

Directly characterizing the convergence of Random Conic Pursuit on problems with constraints appears to be significantly more difficult and seems to require introduction of new quantities depending on the constraint set (e.g., condition number of the constraint set and its overlap with the PSD cone) whose implications for the algorithm are difficult to explicitly characterize with respect to d and the properties of the g_j, X^*, and the Y_t sampling distribution. Indeed, it would be useful to better understand the limitations of Random Conic Pursuit. As noted above, the procedure cannot readily accommodate general equality constraints; furthermore, for some constraint sets, sampling only a rank one Y_t at each iteration could conceivably cause the iterates to become trapped at a sub-optimal boundary point (this could be alleviated by sampling higher rank Y_t). A more general analysis is the subject of continuing work, though our experiments confirm empirically that we realize usefully fast convergence of Random Conic Pursuit even when it is applied to a variety of constrained SDPs.

We obtain a different analytical perspective by recalling that Random Conic Pursuit computes a solution within the random polyhedral cone $\mathcal{F}_n^x$, defined in (4.3) above. The distance between this cone and the optimal matrix X^* is closely related to the quality of solutions produced by Random Conic Pursuit. The following theorem characterizes the distance between a sampled cone $\mathcal{F}_n^x$ and any fixed X^* in the PSD cone:

Theorem 5. *Let $X^* \succ 0$ be a fixed positive definite matrix, and let $x_1, \ldots, x_n \in \mathbb{R}^d$ be drawn i.i.d. from $\mathcal{N}(0, \Sigma)$ with $\Sigma \succ X^*$. Then, for any $\delta > 0$, with probability at least $1 - \delta$,*

$$\min_{X \in \mathcal{F}_n^x} \|X - X^*\| \leq \frac{1 + \sqrt{2} \log \frac{1}{\delta}}{\sqrt{n}} \cdot \frac{2}{e} \sqrt{\left|\Sigma X^{*-1}\right|} \left\| \left(X^{*-1} - \Sigma^{-1}\right)^{-1} \right\|_2$$

See the appendix for proof. As expected, $\mathcal{F}_n^x$ provides a progressively better approximation to the PSD cone (with high probability) as n grows. Furthermore, the rate at which this occurs depends on X^* and its relationship to Σ; as the latter becomes better matched to the former, smaller values of n are required to achieve an approximation of given quality.

The constant Γ in Theorem 4 can hide a dependence on the dimensionality d of the problem, though the proof of Theorem 5 helps to elucidate the dependence of Γ on d and X^* for the particular case when Σ does not vary over time (the constants in Theorem 5 arise from bounding $\|\gamma_t(x_t) x_t x_t'\|$). A potential concern regarding both of the above theorems is the possibility of extremely adverse dependence of their constants on the dimensionality d and the properties (e.g., condition number) of X^*. However, our empirical results in Section 4.3 show that Random Conic Pursuit does indeed decrease the objective function

usefully quickly on real problems with relatively large d and solution matrices X^* which are rank one, a case predicted by the analysis to be among the most difficult.

4.5 Related Work

Random Conic Pursuit and the analyses above are related to a number of existing optimization and sampling algorithms.

Our procedure is closely related to feasible direction methods [72], which move along descent directions in the feasible set defined by the constraints at the current iterate. Cutting plane methods [47], when applied to some SDPs, solve a linear program obtained by replacing the PSD constraint with a polyhedral constraint. Random Conic Pursuit overcomes the difficulty of finding feasible descent directions or cutting planes, respectively, by sampling directions randomly and also allowing the current iterate to be rescaled.

Pursuit-based optimization methods [22, 49] return a solution within the convex hull of an a priori-specified convenient set of points $\mathcal{M}$. At each iteration, they refine their solution to a point between the current iterate and a point in $\mathcal{M}$. The main burden in these methods is to select a near-optimal point in $\mathcal{M}$ at each iteration. For SDPs having only a trace equality constraint and with $\mathcal{M}$ the set of rank one PSD matrices, Hazan [46] shows that such points in $\mathcal{M}$ can be found via an eigenvalue computation, thereby obtaining a convergence rate of $O(1/t)$. In contrast, our method selects steps randomly and still obtains a rate of $O(1/t)$ in the unconstrained case.

The Hit-and-Run algorithm for sampling from convex bodies can be combined with simulated annealing to solve SDPs [55]. In this configuration, similarly to Random Conic Pursuit, it conducts a search along random directions whose distribution is adapted over time.

Finally, whereas Random Conic Pursuit utilizes a randomized polyhedral inner approximation of the PSD cone, the work of Calafiore and Campi [18] yields a randomized outer approximation to the PSD cone obtained by replacing the PSD constraint $X \succeq 0$ with a set of sampled linear inequality constraints. It can be shown that for linear SDPs, the dual of the interior LP relaxation is identical to the exterior LP relaxation of the dual of the SDP. Empirically, however, this outer relaxation requires impractically many sampled constraints to ensure that the problem remains bounded and yields a good-quality solution.

4.A Appendix: Proofs

Proof of Theorem 4. We prove that equation (4.8) holds in general for any X^*, and thus for the optimizer of f in particular. The convexity of f implies the following linear lower bound on $f(X)$ for any X and Y:

$$f(X) \geq f(Y) + \langle \partial f(Y), X - Y \rangle. \tag{4.9}$$

The Lipschitz assumption on the gradient of f implies the following quadratic upper bound on $f(X)$ for any X and Y [60]:

$$f(X) \leq f(Y) + \langle \partial f(Y), X - Y \rangle + \tfrac{L}{2}\|X - Y\|^2. \tag{4.10}$$

Define the random variable $\tilde{Y}_t := \gamma_t(Y_t)Y_t$ with γ_t a positive function that ensures $E\tilde{Y}_t = X^*$. It suffices to set $\gamma_t = q(Y)/\breve{p}(Y)$, where $\breve{p}$ is the distribution of Y_t and q is any distribution with mean X^*. In particular, the choice $\tilde{Y}_t := \gamma_t(x_t)x_t x_t'$ with $\gamma_t(x) = \mathcal{N}(x|0, X^*)/\mathcal{N}(x|0, \Sigma_t)$ satisfies this.

At iteration t, Algorithm 4 produces α_t and β_t so that $X_{t+1} := \alpha_t Y_t + \beta_t X_t$ minimizes $f(X_{t+1})$. We will bound the difference $f(X_{t+1}) - f(X^*)$ at each iteration by sub-optimally picking $\hat{\alpha}_t = 1/t$, $\hat{\beta}_t = 1 - 1/t$, and $\hat{X}_{t+1} = \hat{\beta}_t X_t + \hat{\alpha}_t \gamma_t(Y_t)Y_t = \hat{\beta}_t X_t + \hat{\alpha}_t \tilde{Y}_t$. Conditioned on X_t, we have

$$\begin{aligned}
Ef(X_{t+1}) - f(X^*) &\leq Ef(\hat{\beta}_t X_t + \hat{\alpha}_t \tilde{Y}_t) - f(X^*) && (4.11)\\
&= Ef\left(X_t - \tfrac{1}{t}(X_t - \tilde{Y}_t)\right) - f(X^*) && (4.12)\\
&\leq f(X_t) - f(X^*) + E\left\langle \partial f(X_t), \tfrac{1}{t}(\tilde{Y}_t - X_t)\right\rangle + \tfrac{L}{2t^2}E\|X_t - \tilde{Y}_t\|^2 && (4.13)\\
&= f(X_t) - f(X^*) + \tfrac{1}{t}\langle \partial f(X_t), X^* - X_t\rangle + \tfrac{L}{2t^2}E\|X_t - \tilde{Y}_t\|^2 && (4.14)\\
&\leq f(X_t) - f(X^*) + \tfrac{1}{t}\left(f(X^*) - f(X_t)\right) + \tfrac{L}{2t^2}E\|X_t - \tilde{Y}_t\|^2 && (4.15)\\
&= \left(1 - \tfrac{1}{t}\right)\left(f(X_t) - f(X^*)\right) + \tfrac{L}{2t^2}E\|X_t - \tilde{Y}_t\|^2. && (4.16)
\end{aligned}$$

The first inequality follows by the suboptimality of $\hat{\alpha}_t$ and $\hat{\beta}_t$, the second by Equation (4.10), and the third by (4.9).

Define $e_t := Ef(X_t) - f(X^*)$. The term $E\|\tilde{Y}_t - X_t\|^2$ is bounded above by some absolute constant Γ because $E\|\tilde{Y}_t - X_t\|^2 = E\|\tilde{Y}_t - X^*\|^2 + \|X_t - X^*\|^2$. The first term is bounded because it is the variance of $\tilde{Y}_t$, and the second term is bounded by assumption. Taking expectation over X_t gives the bound $e_{t+1} \leq \left(1 - \frac{1}{t}\right)e_t + \frac{L\Gamma}{2t^2}$, which is solved by $e_t = \frac{1}{t} \cdot \max(\Gamma L, f(X_0) - f(X^*))$ [58]. □

Proof of Theorem 5. We wish to bound the tails of the random variable

$$\min_{\gamma_1 \cdots \gamma_n \geq 0} \left\| \sum_{i=1}^{n} \gamma_i x_i x_i' - X^* \right\|. \tag{4.17}$$

We first simplify the problem by eliminating the minimization over γ. Define a function $\gamma(x; X^*) : \mathbb{R}^d \to \mathbb{R}_+$ that satisfies

$$E_{x \sim \mathcal{N}(0,\Sigma)} \gamma(x) x x' = X^*. \tag{4.18}$$

The choice

$$\gamma(x) = \frac{\mathcal{N}(x|0, X^*)}{\mathcal{N}(x|0, \Sigma)} = |\Sigma|^{1/2}\left|X^*\right|^{-1/2} \exp\left(-\tfrac{x'(X^{*-1} - \Sigma^{-1})x}{2}\right) \tag{4.19}$$

works, since $E_{x\sim\mathcal{N}(0,\Sigma)}\gamma(x)xx' = E_{x\sim\mathcal{N}(0,X^*)}xx' = X^*$. Setting sub-optimally the coefficients γ_i to $\gamma(x_i)$ gives

$$\min_{\gamma\geq 0}\left\|\sum_{i=1}^{n}\gamma_i x_i x_i' - X^*\right\| \leq \left\|\frac{1}{n}\sum_{i=1}^{n}\gamma(x_i)x_i x_i' - E\gamma(x)xx'\right\| \tag{4.20}$$

$$= \left\|\frac{1}{n}\sum_{i=1}^{n} z_i - Ez\right\|. \tag{4.21}$$

Thus, it suffices to bound the tails of the deviation of an empirical average of i.i.d. random variables $z_i := \gamma(x_i)x_i x_i'$ from its expectation, $Ez = X^*$. We proceed using McDiarmid's inequality.

The scalar random variables $\|z_i\|$ are bounded because for all x, we have:

$$\|z\| = \|\gamma(x)xx'\| = \|xx'\||\Sigma|^{\frac{1}{2}}\left|X^*\right|^{-\frac{1}{2}}\exp\left(-\tfrac{x'(X^{*-1}-\Sigma^{-1})x}{2}\right) \tag{4.22}$$

$$\leq |\Sigma|^{\frac{1}{2}}\left|X^*\right|^{-\frac{1}{2}}\|x\|_2^2\exp\left(-\tfrac{\lambda_{min}(X^{*-1}-\Sigma^{-1})\|x\|_2^2}{2}\right) \tag{4.23}$$

$$\leq \frac{2|\Sigma|^{\frac{1}{2}}\left|X^*\right|^{-\frac{1}{2}}}{e\lambda_{min}(X^{*-1}-\Sigma^{-1})} \tag{4.24}$$

$$= \frac{2}{e}|\Sigma|^{\frac{1}{2}}\left|X^*\right|^{-\frac{1}{2}}\left\|\left(X^{*-1}-\Sigma^{-1}\right)^{-1}\right\|_2 \tag{4.25}$$

$$=: \Delta. \tag{4.26}$$

Equation (4.24) follows because the function $f(y) = ye^{-\alpha y}$ is bounded above by $\frac{1}{e\alpha}$.

The expectation of $\left\|\frac{1}{n}\sum_{i=1}^{n} z_i - Ez\right\|$, whose tails we wish to bound, is the standard deviation of $\frac{1}{n}\sum_{i=1}^{n} z_i$, and can be bounded in the standard way in a Hilbert space:

$$\left(E\left\|\frac{1}{n}\sum_{i=1}^{n} z_i - Ez\right\|\right)^2 \leq E\left\|\frac{1}{n}\sum_{i=1}^{n} z_i - Ez\right\|^2 = \frac{1}{n}\left(E\|z\|^2 - \|Ez\|^2\right), \tag{4.27}$$

which yields

$$E\left\|\frac{1}{n}\sum_{i=1}^{n} z_i - Ez\right\| \leq \frac{\Delta}{\sqrt{n}}. \tag{4.28}$$

Using Equations (4.21) and (4.28), and the fact that $\|z_i\| \leq \Delta$, McDiarmid's inequality gives

$$\Pr\left[\min_{\gamma\geq 0}\left\|\sum_{i=1}^{n}\gamma_i x_i x_i' - X^*\right\| > \frac{\Delta}{\sqrt{n}} + \epsilon\right] \tag{4.29}$$

$$\leq \Pr\left[\left\|\tfrac{1}{n}\sum_{i=1}^{n} z_i - Ez\right\| > \frac{\Delta}{\sqrt{n}} + \epsilon\right] \tag{4.30}$$

$$\leq \Pr\left[\left\|\tfrac{1}{n}\sum_{i=1}^{n} z_i - Ez\right\| > E\left\|\tfrac{1}{n}\sum_{i=1}^{n} z_i - Ez\right\| + \epsilon\right]$$

$$\leq \exp\left(-\frac{n\epsilon^2}{2\Delta^2}\right). \tag{4.31}$$

In other words, for any $\delta > 0$, with probability at least $1 - \delta$,

$$\min_{\gamma\geq 0}\left\|\sum_{i=1}^{n}\gamma_i x_i x_i' - X^*\right\| < \frac{\Delta}{\sqrt{n}}\left(1 + \sqrt{2}\log\tfrac{1}{\delta}\right). \tag{4.32}$$

□

Chapter 5

Conclusion

In the preceding chapters, we have proposed novel algorithms—the Bag of Little Bootstraps (BLB), a general bootstrap performance diagnostic, and Random Conic Pursuit—which advance the state of the art for two important classes of problems in machine learning and statistics: estimator quality assessment and semidefinite programming.

BLB provides a powerful new alternative for automatic, accurate assessment of estimator quality that is well suited to large-scale data and modern parallel and distributed computing architectures. Our procedure shares the favorable statistical properties (i.e., consistency and higher-order correctness) and generic applicability of the bootstrap, while typically having a markedly better computational profile, as we have demonstrated via large-scale experiments on a distributed computing platform. Additionally, BLB is consistently more robust than the m out of n bootstrap and subsampling to the choice of subset size and does not require the use of analytical corrections. To enhance our procedure's computational efficiency and render it more automatically usable, we have introduced a means of adaptively selecting its hyperparameters. We have also applied BLB to several real datasets and presented an extension to non-i.i.d. time series data.

A number of open questions and possible extensions remain for BLB. Though we have constructed an adaptive hyperparameter selection method based on the properties of the subsampling and resampling processes used in BLB, as well as empirically validated the method, it would be useful to develop a more precise theoretical characterization of its behavior. Additionally, as discussed in Section 2.5, it would be beneficial to develop a computationally efficient means of adaptively selecting b. It may also be possible to further reduce r by using methods that have been proposed for reducing the number of resamples required by the bootstrap [31, 33].

Furthermore, it is worth noting that averaging the plugin approximations $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ computed by BLB implicitly corresponds to minimizing the squared error of BLB's output. It would be possible to specifically optimize for other losses on estimator quality assessments—thereby improving statistical performance with respect to such losses—by combining the $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ in other ways (e.g., by using medians rather than averages).

While BLB shares the statistical strengths of the bootstrap, we conversely do not expect

our procedure to be applicable in cases in which the bootstrap fails [9]. Indeed, it was such edge cases that originally motivated development of the m out of n bootstrap and subsampling, which are consistent in various settings that are problematic for the bootstrap. It would be interesting to investigate the performance of BLB in such settings and perhaps use ideas from the m out of n bootstrap and subsampling to improve the applicability of BLB in these edge cases while maintaining computational efficiency and robustness.

Although our development of BLB has focused on scalability as the number of available data points increases, various modern data analysis problems exhibit simultaneous growth in number of data points, data dimensionality, and number of parameters to be estimated. As a result, various work in recent years has sought to characterize the statistical performance of inferential procedures in this high-dimensional scaling regime. That research has particularly focused on procedures for point estimation, such as the Lasso [74], and generally makes assumptions (e.g., regarding sparsity of parameters) which allow effective estimation even as the number of parameters to be estimated increases with the number of available data points. Theoretical and empirical investigation of the performance of resampling-based methods for estimator quality assessment (e.g., the bootstrap, BLB, and the m out of n bootstrap) in the high-dimensional scaling regime, and determination of appropriate assumptions under which these techniques are effective (perhaps with some modification) in this setting, would be both interesting and useful.

In addition to addressing the issue of scalability in estimator quality assessment via BLB, we have presented a general diagnostic procedure which permits automatic determination of whether or not the bootstrap is performing satisfactorily when applied to a given dataset and estimator; we have demonstrated the effectiveness of our procedure via an empirical evaluation on a variety of estimators and simulated and real data. A number of avenues of potential future work remain in this vein. Additional study of the influence of the diagnostic's various hyperparameters would be useful. It would also be interesting to evaluate the diagnostic's effectiveness on yet more data generating distributions, estimators, and estimator quality assessments.

Furthermore, it would be interesting to apply our diagnostic procedure to other estimator quality assessment methods such as BLB, the m out of n bootstrap [9], and subsampling [67]. It would also be fairly straightforward to devise extensions of the diagnostic which are suitable for variants of the bootstrap designed to handle non-i.i.d. data [33, 45, 50, 54, 66]. For such bootstrap variants, the diagnostic might aid in selecting a resampling mechanism which respects the dependency structure of the underlying data generating distribution (e.g., by helping to select an appropriate block size when resampling stationary time series).

It should also be possible to characterize theoretically the consistency of our diagnostic procedure, showing that its false positive and false negative rates approach zero as $b_1, \ldots, b_k, p \to \infty$ and $c_1, c_2 \to 0$, under some assumptions (e.g., monotonicity of the bootstrap's convergence to ground truth in cases where it is performing satisfactorily). It would be interesting to make such a result precise.

Finally, we have presented Random Conic Pursuit, a simple, easily implemented randomized solver for general semidefinite programs (SDPs). Unlike interior point methods,

our procedure does not excel at producing highly exact solutions. However, it is more scalable and provides useful approximate solutions fairly quickly, characteristics that are often desirable in machine learning and statistical applications. This fact is illustrated by our experiments on three different machine learning and statistical tasks based on SDPs; we have also provided an analysis yielding further insight into Random Conic Pursuit.

In potential future work, it would be interesting to study the use of other matrix sampling distributions (beyond those considered in our work) in Random Conic Pursuit. Additionally, Random Conic Pursuit can readily benefit from the use of parallel and distributed computational resources, and it would be interesting to empirically evaluate the resulting performance gains. Finally, further analysis of Random Conic Pursuit, particularly in the setting of SDPs with general constraints, would be of interest.

Bibliography

[1] D. Agarwal, R. Agrawal, R. Khanna, and N. Kota. Estimating rates of rare events with multiple hierarchies through scalable log-linear models. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[2] S. Agarwal, A. Panda, B. Mozafari, S. Madden, and I. Stoica. BlinkDB: Queries with bounded errors and bounded response times on very large data. Technical Report 1203.5485, ArXiv, June 2012.

[3] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the USA*, 96:6745–6750, June 1999.

[4] A. Asuncion, P. Smyth, and M. Welling. Asynchronous distributed learning of topic models. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[5] S. Baker, J. Berger, P. Brady, K. Borne, S. Glotzer, R. Hanisch, D. Johnson, A. Karr, D. Keyes, B. Pate, and H. Prosper. Data-enabled science in the mathematical and physical sciences, 2010. Workshop funded by the National Science Foundation.

[6] R. Beran. Diagnosing bootstrap success. *Annals of the Institute of Statistical Mathematics*, 49(1):1–24, 1997.

[7] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., 1989.

[8] P. J. Bickel and D. A. Freedman. Some asymptotic theory for the bootstrap. *Annals of Statistics*, 9(6):1196–1217, 1981.

[9] P. J. Bickel, F. Götze, and W. R. van Zwet. Resampling fewer than n observations: Gains, losses, and remedies for losses. *Statistica Sinica*, 7:1–31, 1997.

[10] P. J. Bickel and A. Sakov. Extrapolation and the bootstrap. *Sankhya: The Indian Journal of Statistics*, 64:640–652, 2002.

[11] P. J. Bickel and A. Sakov. On the choice of m in the m out of n bootstrap and confidence bounds for extrema. *Statistica Sinica*, 18:967–985, 2008.

[12] P. J. Bickel and J. A. Yahav. Richardson extrapolation and the bootstrap. *Journal of the American Statistical Association*, 83(402):387–393, 1988.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[14] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[16] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.

[17] S. Burer and R. D. C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

[18] G. Calafiore and M. C. Campi. Uncertain convex programs: Randomized solutions and confidence levels. *Mathematical Programming*, 102(1):25–46, 2005.

[19] A. J. Canty, A. C. Davison, D. V. Hinkley, and V. Ventura. Bootstrap diagnostics and remedies. *The Canadian Journal of Statistics*, 34(1):5–27, 2006.

[20] L. Chin, W. C. Hahn, G. Getz, and M. Meyerson. Making sense of cancer genomic data. *Genes and Development*, 25:534–555, 2011.

[21] C. Chu, S. K. Kim, Y. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *International Conference on Machine Learning (ICML)*, 2006.

[22] K. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. In *Symposium on Discrete Algorithms (SODA)*, 2008.

[23] Conviva, Inc. http://www.conviva.com, November 2012.

[24] A. d'Aspremont. Subsampling algorithms for semidefinite programming. Technical Report 0803.1990, ArXiv, 2009.

[25] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007.

[26] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[27] P. Diaconis and B. Efron. Computer-intensive methods in statistics. *Scientific American*, 248:96–108, 1983.

[28] F. Doshi-Velez, D. Knowles, S. Mohamed, and Z. Ghahramani. Large scale nonparametric bayesian inference: Data parallelisation in the Indian buffet process. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[29] J. Duchi, A. Agarwal, and M. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 57(3):592–606, March 2012.

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

[31] B. Efron. More efficient bootstrap computations. *Journal of the American Statistical Association*, 85(409):79–89, 1988.

[32] B. Efron. Jackknife-after-bootstrap standard errors and influence functions. *Journal of the Royal Statistical Society, Series B*, 54(1):83–127, 1992.

[33] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[34] A. Frank and A. Asuncion. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2010.

[35] M. H. Fritz, R. Leinonen, G. Cochrane, and E. Birney. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Research*, 21:734–740, 2011.

[36] E. Giné and J. Zinn. Bootstrapping general empirical measures. *Annals of Probability*, 18(2):851–869, 1990.

[37] J. Gonzalez, Y. Low, A. Gretton, and C. Guestrin. Parallel Gibbs sampling: From colored fields to thin junction trees. In *Artificial Intelligence and Statistics (AISTATS)*, 2011.

[38] J. Gonzalez, Y. Low, and C. Guestrin. Residual splash for optimally parallelizing belief propagation. In *Artificial Intelligence and Statistics (AISTATS)*, 2009.

[39] J. Gonzalez, Y. Low, C. Guestrin, and D. O'Hallaron. Distributed parallel inference on large factor graphs. In *Uncertainty in Artificial Intelligence (UAI)*, 2009.

[40] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. `http://cvxr.com/cvx`, May 2010.

[41] Apache Hadoop. http://hadoop.apache.org, April 2012.

[42] J. Hahn. Bootstrapping quantile regression estimators. *Econometric Theory*, 11(1):105–121, 1995.

[43] K. B. Hall, S. Gilpin, and G. Mann. MapReduce/Bigtable for distributed optimization. In *NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*, 2010.

[44] P. Hall. *The Bootstrap and Edgeworth Expansion*. Springer-Verlag New York, Inc., 1992.

[45] P. Hall and E. Mammen. On general resampling algorithms and their performance in distribution estimation. *Annals of Statistics*, 22(4):2011–2030, 1994.

[46] E. Hazan. Sparse approximate solutions to semidefinite programs. In *Latin American Conference on Theoretical Informatics*, pages 306–316, 2008.

[47] C. Helmberg. A cutting plane algorithm for large scale semidefinite relaxations. In Martin Grötschel, editor, *The Sharpest Cut*, chapter 15. MPS/SIAM Series on Optimization, 2001.

[48] M. Hoffman, D. Blei, and F. Bach. Online learning for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[49] L. K. Jones. A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *Annals of Statistics*, 20(1):608–613, March 1992.

[50] H. R. Kunsch. The jackknife and the bootstrap for general stationary observations. *Annals of Statistics*, 17(3):1217–1241, 1989.

[51] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, December 2004.

[52] N. Laptev, K. Zeng, and C. Zaniolo. Early accurate results for advanced analytics on MapReduce. In *Proceedings of the VLDB Endowment*, volume 5, pages 1028–1039, 2012.

[53] P. Liang and D. Klein. Online EM for unsupervised models. In *North American Association for Computational Linguistics (NAACL)*, 2009.

[54] R. Y. Liu and K. Singh. Moving blocks jackknife and bootstrap capture weak dependence. In R. LePage and L. Billard, editors, *Exploring the Limits of the Bootstrap*, pages 225–248. Wiley, 1992.

[55] L. Lovász and S. Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *Foundations of Computer Science (FOCS)*, 2006.

[56] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. GraphLab: A new framework for parallel machine learning. In *Uncertainty in Artificial Intelligence (UAI)*, 2010.

[57] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Distributed GraphLab: A framework for machine learning and data mining in the cloud. In *Proceedings of Very Large Data Bases (PVLDB)*, 2012.

[58] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[59] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, May 2005.

[60] Y. Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, 110(2):245–259, July 2007.

[61] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed inference for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[62] F. Niu, B. Recht, C. Re, and S. J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[63] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[64] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, pages 1573–1375, 2009.

[65] J. Platt. Using sparseness and analytic QP to speed training of Support Vector Machines. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

[66] D. N. Politis and J. P. Romano. The stationary bootstrap. *Journal of the American Statistical Association*, 89(428):1303–1313, 1994.

[67] D. N. Politis, J. P. Romano, and M. Wolf. *Subsampling*. Springer, 1999.

[68] A. Pollack. DNA sequencing caught in deluge of data. *The New York Times*, November 2011.

[69] H. Putter and W. R. van Zwet. Resampling: Consistency of substitution estimators. *Annals of Statistics*, 24(6):2297–2318, 1996.

[70] J. Shao. *Mathematical Statistics*. Springer, second edition, 2003.

[71] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

[72] W. Sun and Y. Yuan. *Optimization Theory and Methods: Nonlinear Programming*. Springer, 2006.

[73] R. Tibshirani. How many bootstraps? Technical report, Department of Statistics, Stanford University, Stanford, CA, 1985.

[74] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

[75] J. Uszkoreit, J. M. Ponte, A. C. Popat, and M. Dubiner. Large scale parallel document mining for machine translation. In *International Conference on Computational Linguistics*, 2010.

[76] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.

[77] A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes*. Springer-Verlag New York, Inc., 1996.

[78] K. Q. Weinberger, F. Sha, Q. Zhu, and L. K. Saul. Graph Laplacian regularization for large-scale semidefinite programming. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[79] J. Wolfe, A. Haghighi, and D. Klein. Fully distributed EM for very large datasets. In *International Conference on Machine Learning (ICML)*, 2008.

[80] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[81] F. Yan, N. Xu, and Y. Qi. Parallel inference for latent Dirichlet allocation on graphics processing units. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[82] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *USENIX NSDI*, 2012.

[83] M. Zinkevich, M. Weimer, A. Smola, and L. Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.