Communication-Efficient Algorithms for Statistical Optimization

Yuchen Zhang[1] **John C. Duchi**[1] **Martin Wainwright**[1,2]
[1]Department of Electrical Engineering and Computer Science and [2]Department of Statistics
University of California, Berkeley
Berkeley, CA 94720
{yuczhang,jduchi,wainwrig}@eecs.berkeley.edu

Abstract

We study two communication-efficient algorithms for distributed statistical optimization on large-scale data. The first algorithm is an averaging method that distributes the N data samples evenly to m machines, performs separate minimization on each subset, and then averages the estimates. We provide a sharp analysis of this average mixture algorithm, showing that under a reasonable set of conditions, the combined parameter achieves mean-squared error that decays as $\mathcal{O}(N^{-1} + (N/m)^{-2})$. Whenever $m \leq \sqrt{N}$, this guarantee matches the best possible rate achievable by a centralized algorithm having access to all N samples. The second algorithm is a novel method, based on an appropriate form of the bootstrap. Requiring only a single round of communication, it has mean-squared error that decays as $\mathcal{O}(N^{-1} + (N/m)^{-3})$, and so is more robust to the amount of parallelization. We complement our theoretical results with experiments on large-scale problems from the internet search domain. In particular, we show that our methods efficiently solve an advertisement prediction problem from the Chinese SoSo Search Engine, which consists of $N \approx 2.4 \times 10^8$ samples and $d \geq 700,000$ dimensions.

1 Introduction

Many problems in machine learning are based on a form of (regularized) empirical risk minimization. Given the current explosion in the size and amount of data, a central challenge in machine learning is to design efficient algorithms for solving large-scale problem instances. In a centralized setting, there are many procedures for solving empirical risk minimization problems, including standard convex programming approaches [3] as well as various types of stochastic approximation [19, 8, 14]. When the size of the dataset becomes extremely large, however, it may be infeasible to store all of the data on a single computer, or at least to keep the data in memory. Accordingly, the focus of this paper is the theoretical analysis and empirical evaluation of some distributed and communication-efficient procedures for empirical risk minimization.

Recent years have witnessed a flurry of research on distributed approaches to solving very large-scale statistical optimization problems (e.g., see the papers [13, 17, 9, 5, 4, 2, 18] and references therein). It can be difficult within a purely optimization-theoretic setting to show explicit benefits arising from distributed computation. In statistical settings, however, distributed computation can lead to gains in statistical efficiency, as shown by Dekel et al. [4] and extended by other authors [2, 18]. Within the family of distributed algorithms, there can be significant differences in communication complexity: different computers must be synchronized, and when the dimensionality of the data is high, communication can be prohibitively expensive. It is thus interesting to study distributed inference algorithms that require limited synchronization and communication while still enjoying the statistical power guaranteed by having a large dataset.

With this context, perhaps the simplest algorithm for distributed statistical inference is what we term the *average mixture* (AVGM) algorithm. This approach has been studied for conditional random fields [10], for perceptron-type algorithms [12], and for certain stochastic approximation methods [23]. It is an appealingly simple method: given m different machines and a dataset of size $N = nm$, give each machine a (distinct) dataset of size $n = N/m$, have each machine i compute the empirical minimizer θ_i on its fraction of the data, then average all the parameters θ_i across the network. Given an empirical risk minimization algorithm that works on one machine, the procedure is straightforward to implement and is extremely communication efficient (requiring only one round of communication); it is also relatively robust to failure and slow machines, since there is no repeated synchronization. To the best of our knowledge, however, no work has shown theoretically that the AVGM procedure has greater statistical efficiency than the naive approach of using n samples on a single machine. In particular, Mann et al. [10] prove that the AVGM approach enjoys a variance reduction relative to the single processor solution, but they only prove that the final mean-squared error of their estimator is $\mathcal{O}(1/n)$, since they do not show a reduction in the bias of the estimator. Zinkevich et al. [23] propose a parallel stochastic gradient descent (SGD) procedure, which runs SGD independently on k machines for T iterations, averaging the outputs. The algorithm enjoys good practical performance, but their main result [23, Theorem 12] guarantees a convergence rate of $\mathcal{O}(\log k/T)$, which is no better than sequential SGD on a single machine processing T samples.

This paper makes two main contributions. First, we provide a sharp analysis of the AVGM algorithm, showing that under a reasonable set of conditions on the statistical risk function, it can indeed achieve substantially better rates. More concretely, we provide bounds on the mean-squared error that decay as $\mathcal{O}((nm)^{-1}+n^{-2})$. Whenever the number of machines m is less than the number of samples n per machine, this guarantee matches the best possible rate achievable by a centralized algorithm having access to all $N = nm$ samples. This conclusion is non-trivial and requires a surprisingly careful analysis. Our second contribution is to develop a novel extension of simple averaging; it is based on an appropriate form of bootstrap [6, 7], which we refer to *bootstrap average mixture* (BAVGM) approach. At a high level, the BAVGM algorithm distributes samples evenly among m processors or computers as before, but instead of simply returning the empirical minimizer, each processor further subsamples its own dataset in order to estimate the bias of its local estimate, returning a bootstrap-corrected estimate. We then prove that the BAVGM algorithm has mean-squared error decaying as $\mathcal{O}(m^{-1}n^{-1} + n^{-3})$. Thus, as long as $m < n^2$, the bootstrap method matches the centralized gold standard up to higher order terms. Finally, we complement our theoretical results with experiments on simulated data and a large-scale logistic regression experiment that arises from the problem of predicting whether a user of a search engine will click on an advertisement. Our experiments show that the resampling and correction of the BAVGM method provide substantial performance benets over naive solutions as well as the averaging algorithm AVGM.

2 Problem set-up and methods

Let $\{f(\cdot; x), x \in \mathcal{X}\}$ be a collection of convex loss functions with domain containing the convex set $\Theta \subseteq \mathbb{R}^d$. Let P be a probability distribution over the sample space $\mathcal{X}$, and define the *population risk function* $F_0 : \Theta \to \mathbb{R}$ via

$$F_0(\theta) := \mathbb{E}_P[f(\theta; X)] = \int_{\mathcal{X}} f(\theta; x) dP(x).$$

We wish to estimate the risk-minimizing parameter $\theta^* = \operatorname{argmin}_{\theta \in \Theta} F_0(\theta) = \int_{\mathcal{X}} f(\theta; x) dP(x)$, which we assume to be unique. In practice, the population distribution P is unknown to us, but we have access to a collection S of samples from the distribution P. In empirical risk minimization, one estimates the vector θ^* by solving the optimization problem $\widehat{\theta} \in \operatorname{argmin}_{\theta \in \Theta} \frac{1}{|S|} \sum_{x \in S} f(\theta; x)$. Throughout the paper, we impose some standard regularity conditions on the parameter space and its relationship to the optimal parameter θ^*.

Assumption A (Parameters)**.** *The parameter space $\Theta \subset \mathbb{R}^d$ is closed convex with $\theta^* \in \operatorname{int} \Theta$.*

We use $R = \sup_{\theta \in \Theta} \|\theta - \theta^*\|_2$ to denote the ℓ_2-diameter of the parameter space with respect to the optimum. In addition, the risk function is required to have some amount of curvature:

Assumption B (Local strong convexity)**.** *There exists a $\lambda > 0$ such that the population Hessian matrix $\nabla^2 F_0(\theta^*) \succeq \lambda I_{d \times d}$.*

Here $\nabla^2 F_0(\theta^*)$ denotes the Hessian of the population objective F_0 evaluated at θ^*. Note that this local condition is milder than a global strong convexity condition and is required to hold only for the population risk F_0. It is of course well-known that some type of curvature is required to consistently estimate the parameters θ^*.

We now describe our methods. In the distributed setting, we are given a dataset of $N = mn$ samples i.i.d. according to the initial distribution P, which we divide evenly amongst m processors or inference procedures. Let S_j, $j \in \{1, 2, \ldots, m\}$, denote a subsampled dataset of size n, and define the (local) empirical distribution P_1 and empirical objective F_1 via

$$P_{1,j} := \frac{1}{|S_j|} \sum_{x \in S_j} \delta_x \quad \text{and} \quad F_{1,j}(\theta) := \frac{1}{|S_j|} \sum_{x \in S_j} f(\theta; x) = \int_{\mathcal{X}} f(\theta; x) dP_{1,j}(x).$$

The AVGM procedure operates as follows: for $j \in \{1, \ldots, m\}$, machine j uses its dataset S_j to compute a vector $\theta_{1,j} \in \operatorname{argmin}_{\theta \in \Theta} F_{1,j}(\theta)$. AVGM combines these m estimates by averaging:

$$\overline{\theta}_1 := \frac{1}{m} \sum_{j=1}^{m} \theta_{1,j}. \tag{1}$$

The bootstrap average mixture (BAVGM) procedure is based on an additional level of random sampling. In particular, for a parameter $r \in (0, 1]$, each machine j draws a subset $S_{2,j}$ of size $\lceil rn \rceil$ by sampling uniformly at random *without replacement* from its local data set S_j. In addition to computing the empirical minimizer $\theta_{1,j}$ based on S_j, BAVGM also computes the empirical minimizer $\theta_{2,j}$ of the function $F_{2,j}(\theta) := \frac{1}{|S_{2,j}|} \sum_{x \in S_{2,j}} f(\theta; x)$, constructing the bootstrap average $\overline{\theta}_2 := \frac{1}{m} \sum_{j=1}^{m} \theta_{2,j}$ and returning the estimate

$$\overline{\theta}_{\text{BAVGM}} := \frac{\overline{\theta}_1 - r\overline{\theta}_2}{1 - r}. \tag{2}$$

The parameter $r \in (0, 1)$ is a user-defined quantity. The purpose of the weighted estimate (2) is to perform a form of bootstrap bias correction [6, 7]. In rough terms, if $b_0 = \theta^* - \theta_1$ is the bias of the first estimator, then we may approximate b_0 by the bootstrap estimate of bias $b_1 = \theta_1 - \theta_2$. Then, since $\theta^* = \theta_1 + b_0$, we use the fact that $b_1 \approx b_0$ to argue that $\theta^* = \theta_1 + b_0 \approx \theta_1 + b_1$.[1]

3 Main results

3.1 Bounds for simple averaging

To guarantee good estimation properties of our algorithms, we require regularity conditions on the empirical risks F_1 and F_2. It is simplest to state these in terms of the sample functions f, and we note that, as with Assumption B, we require these to hold only locally around the optimal point θ^*.

Assumption C. *For some $\rho > 0$, there exists a neighborhood $U = \{\theta \in \mathbb{R}^d : \|\theta^* - \theta\|_2 \le \rho\} \subseteq \Theta$ such that for arbitrary $x \in \mathcal{X}$, the gradient and the Hessian of f exist and satisfy the bounds*

$$\|\nabla f(\theta; x)\|_2 \le G \quad \text{and} \quad \left|\!\left|\!\left|\nabla^2 f(\theta; x)\right|\!\right|\!\right|_2 \le H.$$

for finite constants G, H. For $x \in \mathcal{X}$, the Hessian matrix $\nabla^2 f(\theta; x)$ is Lipschitz continuous for $\theta \in U$: there is a constant L such that $\left|\!\left|\!\left|\nabla^2 f(v; x) - \nabla^2 f(w; x)\right|\!\right|\!\right|_2 \le L \|v - w\|_2$ for $v, w \in U$.

While Assumption C may appear strong, some smoothness of $\nabla^2 f$ is *necessary* for averaging methods to work, as we now demonstrate by an example. (In fact, this example underscores the difficulty of proving that the AVGM algorithm achieves better mean-squared error than single-machine strategies.) Consider a distribution $\{0, 1\}$ with $P(X = 0) = P(X = 1) = 1/2$, and use the loss

$$f(\theta; x) = \begin{cases} \theta^2 - \theta & \text{if } x = 0 \\ \theta^2 1(\theta \le 0) + \theta & \text{if } x = 1. \end{cases} \tag{3}$$

The associated population risk is $F_0(w) = \frac{1}{2}(w^2 + w^2 1_{(w \le 0)})$, which is strongly convex and smooth, since $|F_0'(w) - F_0'(v)| \le 2|w - v|$, but has discontinuous second derivative. Evidently $\theta^* = 0$, and by an asymptotic expansion we have that $\mathbb{E}[\theta_1] = \Omega(n^{-\frac{1}{2}})$ (see the long version of our paper [22, Appendix D] for this asymptotic result). Consequently, the bias of $\overline{\theta}_1$ is $\Omega(n^{-\frac{1}{2}})$, and the AVGM

[1] When the index j is immaterial, we use the shorthand notation θ_1 and θ_2 to denote $\theta_{1,j}$ and $\theta_{2,j}$, respectively, and similarly with other quantities.

algorithm using $N = mn$ observations must suffer mean squared error $\mathbb{E}[(\overline{\theta}_1 - \theta^*)^2] = \Omega(n^{-1})$. Some type of smoothness is necessary for fast rates.

That being said, Assumptions B and C are somewhat innocuous for practical problems. Both hold for logistic and linear regression problems so long as the *population* data covariance matrix is not rank deficient and the data is bounded; moreover, in the linear regression case, we have $L = 0$.

Our assumptions in place, we present our first theorem on the convergence of the AVGM procedure. We provide the proof of Theorem 1—under somewhat milder assumptions—and its corollaries in the full version of this paper [22].

Theorem 1. *For each $i \in \{1, \ldots, m\}$, let S_i be a dataset of n independent samples, and let*

$$\theta_{1,i} \in \operatorname*{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{x_j \in S_i} f(\theta; x_j)$$

denote the minimizer of the empirical risk for the dataset S_i. Define $\overline{\theta}_1 = \frac{1}{m} \sum_{i=1}^m \theta_{1,i}$ and let θ^* denote the population risk minimizer. Then under Assumptions A–C, we have

$$\begin{aligned} \mathbb{E}\left[\left\|\overline{\theta}_1 - \theta^*\right\|_2^2\right] &\le \frac{2}{nm} \mathbb{E}\left[\left\|\nabla^2 F_0(\theta^*)^{-1} \nabla f(\theta^*; X)\right\|_2^2\right] \\ &+ \frac{5}{\lambda^2 n^2}\left(H^2 \log d + \mathbb{E}\left[\left\|\nabla^2 F_0(\theta^*)^{-1} \nabla f(\theta^*; X)\right\|_2^2\right]\right) \mathbb{E}\left[\left\|\nabla^2 F_0(\theta^*)^{-1} \nabla f(\theta^*; X)\right\|_2^2\right] \\ &+ \mathcal{O}(m^{-1} n^{-2}) + \mathcal{O}(n^{-3}). \end{aligned} \tag{4}$$

A simple corollary of Theorem 1 makes it somewhat easier to parse, though we prefer the general form in the theorem as its dimension dependence is somewhat stronger. Specifically, note that by definition of the operator norm, $|||Ax|||_2 \le |||A|||\, \|x\|_2$ for any matrix A and vector x. Consequently,

$$\left\|\nabla^2 F_0(\theta^*)^{-1} \nabla f(\theta^*; x)\right\|_2 \le \left|\left|\left|\nabla^2 F_0(\theta^*)^{-1}\right|\right|\right|_2 \left\|\nabla f(\theta^*; x)\right\|_2 \le \frac{1}{\lambda} \left\|\nabla f(\theta^*; x)\right\|_2,$$

where for the last inequality we used Assumption B. In general, this upper bound may be quite loose, and in many statistical applications (such as linear regression) multiplying $\nabla f(\theta^*; X)$ by the inverse Hessian standardizes the data. Assumption C implies $\mathbb{E}[\|\nabla f(\theta^*; X)\|_2^2] \le G^2$, so that we arrive at the following:

Corollary 1. *Under the same conditions as Theorem 1, we have*

$$\mathbb{E}\left[\left\|\overline{\theta}_1 - \theta^*\right\|_2^2\right] \le \frac{2G^2}{\lambda^2 nm} + \frac{5G^2}{\lambda^4 n^2}\left(H^2 \log d + \frac{G^2}{\lambda^2}\right) + \mathcal{O}(m^{-1} n^{-2}) + \mathcal{O}(n^{-3}).$$

A comparison of Theorem 1's conclusions with classical statistical results is also informative. If the loss $f(\cdot; x) : \Theta \to \mathbb{R}$ is the negative log-likelihood $\ell(x \mid \theta)$ for a parametric model $P(\cdot \mid \theta^*)$, then under suitable smoothness conditions on the log likelihood [21], we can define the Fisher Information matrix

$$I(\theta^*) := \mathbb{E}_{\theta^*}\left[\nabla \ell(X \mid \theta^*) \nabla \ell(X \mid \theta^*)^\top\right] = \mathbb{E}_{\theta^*}[\nabla^2 \ell(X \mid \theta^*)],$$

where $\mathbb{E}_{\theta^*}$ denotes expectation under the model $P(\cdot \mid \theta^*)$. Let $N = mn$ denote the total number of samples available. Then under our assumptions, we have the minimax result [21, Theorem 8.11] that for any estimator $\widehat{\theta}_N$ based on N samples,

$$\sup_{M < \infty} \liminf_{N \to \infty} \sup_{\|\delta\| \le M/\sqrt{N}} \mathbb{E}_{\theta^* + \delta}\left[\left\|\widehat{\theta}_N - \theta^* - \delta\right\|_2^2\right] \ge \operatorname{tr}(I(\theta^*)^{-1}). \tag{5}$$

In connection with Theorem 1, we obtain the comparative result

Corollary 2. *Let the assumptions of Theorem 1 hold, and assume that the loss functions $f(\cdot; x)$ are the negative log-likelihood $\ell(x \mid \theta)$ for a parametric model $P(\cdot \mid \theta^*)$. Let $N = mn$. Then*

$$\mathbb{E}\left[\left\|\overline{\theta}_1 - \theta^*\right\|_2^2\right] \le \frac{2}{N} \operatorname{tr}(I(\theta^*)^{-1}) + \frac{5m^2 \operatorname{tr}(I(\theta^*)^{-1})}{\lambda^2 N^2}\left(H^2 \log d + \operatorname{tr}(I(\theta^*)^{-1})\right) + \mathcal{O}(m^{-1} n^{-2}).$$

Except for the factor of 2 in the bound, Corollary 2 shows that Theorem 1 essentially achieves the best possible result. The important aspect of our bound, however, is that we obtain this convergence rate without calculating an estimate on all $N = mn$ data samples x_i; we calculate m independent estimators and average them to attain the convergence guarantee.

3.2 Bounds for bootstrap mixture averaging

As shown in Theorem 1 and the immediately preceding corollary, for small m, the convergence rate of the AVGM algorithm is mainly determined by the first term in the bound (4), which is at worst $\frac{G^2}{\lambda^2 mn}$. When the number of processors m grows, however, the second term in the bound (4) may have non-negligible effect in spite of being $\mathcal{O}(n^{-2})$. In addition, when the population risk's local strong convexity parameter λ is close to zero or the Lipschitz continuity constant H of $\nabla f(\theta; x)$ is large, the n^{-2} term in the bound (4) and Corollary 1 may dominate the leading term. This concern motivates our development of the bootstrap average mixture (BAVGM) algorithm and analysis.

Due the additional randomness introduced by the bootstrap algorithm BAVGM, its analysis requires an additional smoothness condition. In particular, we require that in a neighborhood of the optimal point θ^*, the loss function f is smooth through its third derivatives.

Assumption D. *For a $\rho > 0$, there exists a neighborhood $U = \{\theta \in \mathbb{R}^d : \|\theta^* - \theta\|_2 \le 2\rho\} \subseteq \Theta$ such that the smoothness conditions of Assumption C hold. For $x \in \mathcal{X}$, the third derivatives of f are Lipschitz continuous: there is a constant $M \ge 0$ such that for $v, w \in U$ and $u \in \mathbb{R}^d$,*

$$\left\| \left(\nabla^3 f(v; x) - \nabla^3 f(w; x)\right)(u \otimes u)\right\|_2 \le M \|v - w\|_2 \, \|u \otimes u\|_2 = M \|v - w\|_2 \, \|u\|_2^2 .$$

Note that Assumption D holds for linear regression (in fact, with $M = 0$); it also holds for logistic regression problems with finite M as long as the data is bounded.

We now state our second main theorem, which shows that the use of bootstrap samples to reduce the bias of the AVGM algorithm yields improved performance. (Again, see [22] for a proof.)

Theorem 2. *Let Assumptions A–D hold. Then the output $\overline{\theta}_{\text{BAVGM}} = \frac{\overline{\theta}_1 - r\overline{\theta}_2}{1-r}$ of the bootstrap* BAVGM *algorithm satisfies*

$$\mathbb{E}\left[\left\|\overline{\theta}_{\text{BAVGM}} - \theta^*\right\|_2^2\right] \le \frac{2+3r}{(1-r)^2} \cdot \frac{1}{nm} \mathbb{E}\left[\left\|\nabla^2 F_0(\theta^*)^{-1} \nabla f(\theta^*; X)\right\|_2^2\right] + \mathcal{O}\left(\frac{1}{(1-r)^2} m^{-1} n^{-2} + \frac{1}{r(1-r)^2} n^{-3}\right) \tag{6}$$

Comparing the conclusions of Theorem 2 to those of Theorem 1, we see that the the $\mathcal{O}(n^{-2})$ term in the bound (4) has been eliminated. The reason for this elimination is that resampling at a rate r reduces the bias of the BAVGM algorithm to $\mathcal{O}(n^{-3})$; the bias of the AVGM algorithm induces terms of order n^{-2} in Theorem 1. Unsurprisingly, Theorem 2 suggests that the performance of the BAVGM algorithm is affected by the resampling rate r; typically, one uses $r \in (0, 1)$. Roughly, when m becomes large we increase r, since the bias of the independent solutions may increase and we enjoy averaging affects from the BAVGM algorithm. When m is small, the BAVGM algorithm appears to provide limited benefits. The big-$\mathcal{O}$ notation hides some problem dependent constants for simplicity in the bound. We leave as an intriguing open question whether computing multiple bootstrap samples at each machine can yield improved performance for the BAVGM procedure.

3.3 Time complexity

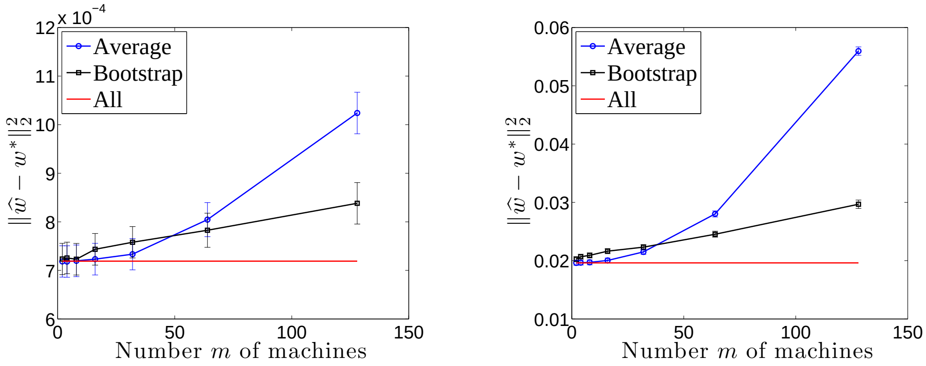
In practice, the exact empirical minimizers assumed in Theorems 1 and 2 may be unavailable. In this section, we sketch an argument that shows that both the AVGM algorithm and the BAVGM algorithm can use approximate empirical minimizers to achieve the same (optimal) asymptotic bounds. Indeed, suppose that we employ approximate empirical minimizers in AVGM and BAVGM instead of the exact ones.[2] Let the vector θ' denotes the approximation to the vector θ (at each point of the algorithm). With this notation, we have by the triangle inequality and Jensen's inequality that

$$\mathbb{E}[\|\overline{\theta}'_1 - \theta^*\|_2^2] \le 2\mathbb{E}[\left\|\overline{\theta}_1 - \theta^*\right\|_2^2] + 2\mathbb{E}[\|\overline{\theta}'_1 - \overline{\theta}_1\|_2^2] \le 2\mathbb{E}[\left\|\overline{\theta}_1 - \theta^*\right\|_2^2] + 2\mathbb{E}[\|\theta'_1 - \theta_1\|_2^2]. \tag{7}$$

The bound (7) shows that solving the empirical minimization problem to accuracy sufficient to have $\mathbb{E}[\|\theta'_1 - \theta_1\|_2^2] = \mathcal{O}((mn)^{-2})$ guarantees the same convergence rates provided by Theorem 1.

Now we show that in time $\mathcal{O}(n \log(mn))$—assuming that processing one sample requires one unit of time—it is possible to achieve empirical accuracy $\mathcal{O}((nm)^{-2})$. When this holds, the speedup

[2] We provide the arguments only for the AVGM algorithm to save space; the arguments for the BAVGM algorithm are completely similar, though they also include θ_2.



(a) (b)

Figure 1: Experiments plotting the error in the estimate of θ^* given by the AVGM algorithm and BAVGM algorithm for total number of samples $N = 10^5$ versus number of dataset splits (parallel machines) m. Each plot indicates a different dimension d of problem. (a) $d = 20$, (b) $d = 100$.

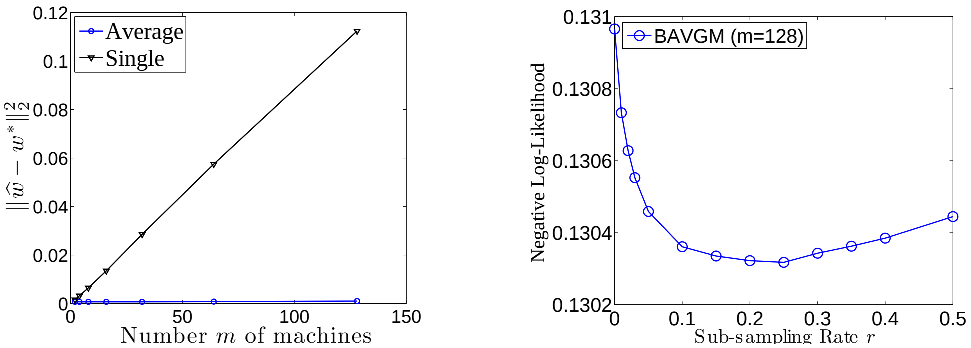
of the AVGM and similar algorithms over the naive approach of processing all $N = mn$ samples on one processor is at least of order $m/\log(N)$. Let us argue that for such time complexity the necessary empirical convergence is achievable. As we show in our proof of Theorem 1, with high probability the empirical risk F_1 is strongly convex in a ball $B_\rho(\theta_1)$ of constant radius $\rho > 0$ around θ_1 with high probability. (A similar conclusion holds for F_2.) A combination of stochastic gradient descent [14] and standard convex programming approaches [3] completes the argument. Indeed, performing stochastic gradient descent for $\mathcal{O}(\log^2(mn)/\rho^2)$ iterations on the empirical objective F_1 yields that with probability at least $1 - m^{-2}n^{-2}$, the resulting parameter falls within $B_\rho(\theta_1)$ [14, Proposition 2.1]. The local strong convexity guarantees that $\mathcal{O}(\log(mn))$ iterations of standard gradient descent [3, Chapter 9]—each requiring $\mathcal{O}(n)$ units of time—beginning from this parameter is sufficient to achieve $\mathbb{E}[\|\theta_1' - \theta_1\|_2^2] = \mathcal{O}((mn)^{-2})$, since gradient descent enjoys a locally linear convergence rate. The procedure outlined requires at most $\mathcal{O}(n\log(mn))$ units of time.

We also remark that under a slightly more global variant of Assumptions A–C, we can show that stochastic gradient descent achieves convergence rates of $\mathcal{O}((mn)^{-2} + n^{-3/2})$, which is order optimal. See the full version of this paper [5, Section 3.4] for this result.

4 Experiments with synthetic data

In this section, we report the results of simulation studies comparing the AVGM and BAVGM methods, as well as a trivial method using only a fraction of the data available on a single machine. For our simulated experiments, we solve linear regression problems of varying dimensionality. For each experiment, we use a fixed total number $N = 10^5$ of samples, but we vary the number of parallel splits m of the data (and consequently, the local dataset sizes $n = N/m$) and the dimensionality d of the problem solved. For each simulation, we choose a constant vector $u \in \mathbb{R}^d$. The data samples consist of pairs (x, y), where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$ is the target value. To sample each x vector, we choose five entries of x distributed as $\mathcal{N}(0, 1)$; the remainder of x is zero. The vector y is sampled as $y = \langle u, x\rangle + \sum_{j=1}^d (x_j/2)^3$, so the noise in the linear estimate $\langle u, x\rangle$ is correlated with x. For our linear regression problem, we use the loss $f(\theta; (x, y)) := \frac{1}{2}(\langle\theta, x\rangle - y)^2$. We attempt to find the vector θ^* minimizing $F(\theta) = \mathbb{E}[f(\theta; (X, Y))]$ using the standard batch solution, using AVGM, using BAVGM, and simply solving the linear regression problem resulting from a single split of the data (of size N/m). We use $m \in \{2, 4, 8, 16, 32, 64, 128\}$ datasets, recalling that the distributed datasets are of size $n = N/m$. We perform experiments with each of the dimensionalities $d = 20, 50, 100, 200, 400$. (We plot $d = 20$ and $d = 100$; other results are qualitatively similar.)

Let $\widehat{\theta}$ denote the vector output by any of our procedures after inference (so in the BAVGM case, for example, this is the vector $\widehat{\theta} = \overline{\theta}_{\text{BAVGM}} = (\overline{\theta}_1 - r\overline{\theta}_2)/(1 - r)$). We obtain the true optimal vector θ^* by solving the linear regression problem with sufficiently large number of samples. In Figure 1, we plot the error $\|\widehat{\theta} - \theta^*\|_2^2$ of the inferred parameter vector $\widehat{\theta}$ for the true parameters θ^* versus the number of splits, or number of parallel machines, m we use. We also plot standard errors (across



(a) (b)

Figure 2: (a) Sythetic data: comparison of AVGM estimator to linear regression estimator based on N/m data points. (b) Advertising data: the log-loss on held-out data for the BAVGM method applied with $m = 128$ parallel splits of the data, plotted versus the sub-sampling rate r.

fifty experiments) for each curve. In each plot, the flat bottom line is the error of the batch method using all the N samples.

From the plots in Figure 1, we can make a few claims. First, the AVGM and BAVGM algorithms indeed enjoy excellent performance, as our theory predicts. Even as the dimensionality d grows, we see that splitting the data into as many as $m = 64$ independent pieces and averaging the solution vectors θ_i estimated from each subsample i yields a vector $\widehat{\theta}$ whose estimate of θ^* is no worse than twice the solution using all N samples. We also see that the AVGM curve appears to increase roughly quadratically with m. This agrees with our theoretical predictions in Theorem 1. Indeed, setting $n = N/m$, we see that Theorem 1 implies $\mathbb{E}[\|\overline{\theta} - \theta^*\|_2^2] = \mathcal{O}(\frac{1}{mn} + \frac{1}{n^2}) = \mathcal{O}(\frac{1}{N} + \frac{m^2}{N^2})$, which matches Figure 1. In addition, we see that the BAVGM algorithm enjoys somewhat more stable performance, with increasing benefit as the number of machines m increases. We chose $r \propto \sqrt{d/n}$ for the BAVGM algorithm, as that choice appeared to give reasonable performance. (The optimal choice of r remains an open question.)

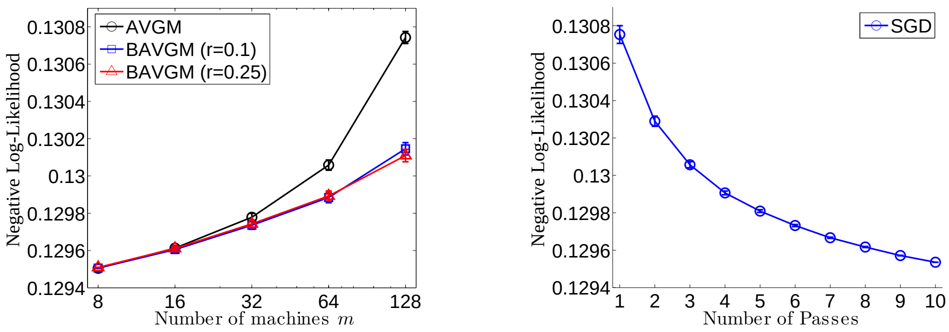
As a check that our results are not simply consequences of the fact that the problems are easy to solve, even using a fraction $1/m$ of the data in a single machine, in Figure 2(a) we plot the estimation error $\|\widehat{\theta} - \theta^*\|_2^2$ of an estimate of θ^* based on just a fraction $1/m$ of the data versus the number of machines/data splits m. Clearly, the average mixture approach dominates. (Figure 2(a) uses $d = 20$; larger dimensions are similar but more pronounced).

5 Experiments with advertising data

Predicting whether a user of a search engine will click on an advertisement presented to him or her is of central importance to the business of several internet companies, and in this section, we present experiments studying the performance of the AVGM and BAVGM methods for this task. We use a large dataset from the Tencent search engine, `soso.com` [20], which contains 641,707 distinct advertisement items with N = 235,582,879 data samples. Each sample consists of a so-called *impression*, which is a list containing a user-issued search, the advertisement presented to the user and a label $y \in \{+1, -1\}$ indicating whether the user clicked on the advertisement. The ads in our dataset were presented to 23,669,283 distinct users.

Tencent dataset provides a standard encoding to transform an impression into a useable set of regressors x. We list the features present in the data in Table 1 of the full version of this paper [22]. Each text-based feature is given a "bag-of-words" encoding [11]. Real-valued features are binned into a fixed number of intervals. When a feature falls into a particular bin, the corresponding entry of is assigned a 1, and otherwise assigned 0. This combination of encodings yields a binary-valued covariate vector $x \in \{0, 1\}^d$ with d = 741,725 dimensions.

Our goal is to predict the probability of a user clicking a given advertisement as a function of the covariates x. In order to do so, we use a logistic regression model to estimate the probability of a click response $P(y = 1 \mid x; \theta) := \frac{1}{1+\exp(-\langle\theta,x\rangle)}$, where $\theta \in \mathbb{R}^d$ is the unknown regression vector.



(a) (b)

Figure 3: The negative log-likelihood of the output of the AVGM, BAVGM, and a stochastic gradient descent method on the held-out dataset for the click-through prediction task. (a) Performance of the AVGM and BAVGM methods versus the number of splits m of the data. (b) Performance of the SGD baseline as a function of number of passes through the entire dataset.

We use the negative logarithm of P as the loss, incorporating a ridge regularization penalty. This combination yields the optimization objective

$$f(\theta;(x,y)) = \log\left(1+\exp(-y\left\langle\theta,x\right\rangle)\right) + \frac{\lambda}{2}\left\|\theta\right\|_2^2 .$$

In all our experiments, we use regularization parameter $\lambda = 10^{-6}$, a choice obtained by cross validation.

For this problem, we cannot evaluate the mean-squared error $\|\widehat{\theta}-\theta^*\|_2^2$, as we do not know the true optimal parameter θ^*. Consequently, we evaluate the performance of an estimate $\widehat{\theta}$ using log-loss on a held-out dataset. Specifically, we perform a five-fold validation experiment, where we shuffle the data and partition it into five equal-sized subsets. For each of our five experiments, we hold out one partition to use as the test set, using the remaining data as the training set used for inference. When studying the AVGM or BAVGM method, we compute the local estimate θ_i via a trust-region Newton-based method [15].

The dataset is too large to fit in main memory on most computers: in total, four splits of the data require 55 gigabytes. Consequently, it is difficult to provide an oracle training comparison using the full N samples. Instead, for each experiment, we perform 10 passes of stochastic gradient descent through the dataset to get a rough baseline of the performance attained by the empirical minimizer for the entire dataset. Figure 3(b) shows the hold-out set log-loss after each of the sequential passes through the training data finishes.

In Figure 3(a), we show the average hold-out set log-loss (with standard errors) of the estimator $\overline{\theta}_1$ provided by the AVGM method and the BAVGM method versus number of splits of the data m. The plot shows that for small m, both AVGM and BAVGM enjoy good performance, comparable to or better than (our proxy for) the oracle solution using all N samples. As the number of machines m grows, the de-biasing provided by the subsampled bootstrap method yield substantial improvements over the standard AVGM method. In addition, even with $m = 128$ splits of the dataset, the BAVGM method gives better hold-out set performance than performing two passes of stochastic gradient on the entire dataset of m samples. This is striking, as doing even one pass through the data with stochastic gradient descent is known to give minimax optimal convergence rates [16, 1].

It is instructive and important to understand the sensitivity of the BAVGM method to the resampling parameter r. We explore this question in in Figure 2(b) using $m = 128$ splits. We choose $m = 128$ because more data splits provide more variable performance in r. For the `soso.com` ad prediction data set, the choice $r = .25$ achieves the best performance, but Figure 2(b) suggests that misspecifying the ratio is not terribly detrimental. Indeed, while the performance of BAVGM degrades to that of the AVGM method, there is a wide range of r giving improved performance, and there does not appear to be a phase transition to poor performance.

Acknowledgments This work is based on research supported in part by the Office of Naval Research under MURI grant N00014-11-1-0688. JCD was also supported by an NDSEG fellowship and a Facebook PhD fellowship.

References

[1] A. Agarwal, P. L. Bartlett, P. Ravikumar, and M. J. Wainwright. Information-theoretic lower bounds on the oracle complexity of convex optimization. *IEEE Transactions on Information Theory*, 58(5):3235–3249, May 2012.

[2] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems 25*, 2011.

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

[4] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202, 2012.

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

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

[7] P. Hall. *The Bootstrap and Edgeworth Expansion*. Springer, 1992.

[8] E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006.

[9] B. Johansson, M. Rabi, and M. Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3):1157–1170, 2009.

[10] G. Mann, R. McDonald, M. Mohri, N. Silberman, and D. Walker. Efficient Large-Scale Distributed Training of Conditional Maximum Entropy Models. In *Advances in Neural Information Processing Systems 22*, pages 1231–1239, 2009.

[11] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[12] R. McDonald, K. Hall, and G. Mann. Distributed training strategies for the structured perceptron. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2010.

[13] A. Nedić and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54:48–61, 2009.

[14] 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.

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

[16] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[17] S. S. Ram, A. Nedić, and V. V. Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of Optimization Theory and Applications*, 147(3):516–545, 2010.

[18] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: a lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems 25*, 2011.

[19] H. Robbins. Asymptotically subminimax solutions of compound statistical decision problems. In *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, pages 131–148, 1951.

[20] G. Sun. KDD cup track 2 `soso.com` ads prediction challenge, 2012. Accessed August 1, 2012.

[21] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[22] Y. Zhang, J. C. Duchi, and M. J. Wainwright. Communication-efficient algorithms for statistical optimization. *arXiv:1209.4129 [stat.ML]*, 2012.

[23] M. A. Zinkevich, A. Smola, M. Weimer, and L. Li. Parallelized Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems 24*, 2010.