Distributed Learning of Non-Convex Linear Models with One Round of Communication

Mike Izbicki[1] and Christian R. Shelton[2]

[1] Claremont McKenna College, Claremont, CA, USA
mike@izbicki.me

[2] UC Riverside, Riverside, CA, USA
cshelton@cs.ucr.edu

Abstract. We present the *optimal weighted average* (OWA) distributed learning algorithm for linear models. OWA achieves statistically optimal learning rates, uses only one round of communication, works on non-convex problems, and supports a fast cross validation procedure. The OWA algorithm first trains local models on each of the compute nodes; then a master machine merges the models using a second round of optimization. This second optimization uses only a small fraction of the data, and so has negligible computational cost. Compared with similar distributed estimators that merge locally trained models, OWA either has stronger statistical guarantees, is applicable to more models, or has a more computationally efficient merging procedure.

Keywords: distributed machine learning · linear models

1 Introduction

Many datasets are too large to fit in the memory of a single machine. To analyze them, we must partition the data onto many machines and use distributed algorithms. Existing distributed learning algorithms fall into one of two categories:

Interactive algorithms require many rounds of communication between machines. Representative examples include [4, 14, 8, 16, 27, 23]. These algorithms resemble standard iterative algorithms where each iteration is followed by a communication step. The appeal of interactive algorithms is that they enjoy the same statistical performance as standard sequential algorithms. That is, given m machines each with n data points of dimension d, interactive algorithms have error that decays as $O(\sqrt{d/mn})$ for linear models. But, interactive algorithms have three main disadvantages. First, these algorithms are slow when communication latency is the bottleneck. An extreme example occurs in the *federated learning* environment proposed by McMahan et al. [18], which uses cell phones as the computational nodes. Recent work on interactive algorithms focuses on reducing this communication as much as possible [8, 27, 23]. Second, these algorithms require special implementations. They are not easy for non-experts to implement or use, and in particular they do not work with off-the-shelf statistics libraries

provided by (for example) Python, R, and Matlab. Third, because of the many rounds of communication, any sensitive information in the data is likely to leak between machines.

Non-interactive algorithms require only a single round of communication. Each machine independently solves the learning problem on a small subset of data, then a master machine merges the solutions together. These algorithms solve all the problems of interactive ones: they are fast when communication is the main bottleneck; they are easy to implement with off-the-shelf statistics packages; and they are robust to privacy considerations. The downside is worse statistical performance. The popular naive averaging estimator has worst case performance $O(\sqrt{d/n})$ completely independent of the number of machines m. A growing body of work improves the analysis of the averaging estimator under special conditions [17, 25, 26, 22, 24], and develops more robust non-interactive estimators [28, 15, 12, 2, 6, 9]. All of these estimators either work on only a limited class of models or have computationally intractable merge procedures.

In this paper, we propose a novel non-interactive estimator called the *optimal weighted average* (OWA). OWA's merge procedure uses a second round of optimization over the data. (All previous merge procedures do not depend on the data.) This data dependent merge procedure has four advantages: (i) OWA achieves the optimal error of $O(\sqrt{d/mn})$ in a general setting and with a simple analysis. In particular, we do not require a convex loss function. (ii) This second optimization uses a small number of data points projected onto a small dimensional space. It therefore has negligible computational and communication overhead. (iii) OWA is easily implemented on most distributed architectures with standard packages. Our implementation uses only a few dozen lines of Python and scikit-learn [21]. (iv) OWA is robust to the regularization strength used in the first round of optimization. In practice, this means that OWA does not require communication between nodes even in the model selection step of learning.

We also show a simple extension to the OWA algorithm that uses two rounds of communication to compute a cross validation estimate of the model's performance. The standard version of cross validation is too slow for large scale data, and therefore not widely used in the distributed setting. This procedure is the first fast cross validation method designed for the distributed setting, and is an additional advantage OWA has over interactive distributed learning algorithms.

Section 2 formally describes our problem setting, and Section 3 describes the OWA algorithm and its fast cross validation procedure. We take special care to show how OWA can be implemented with off-the-shelf optimizers. Section 4 provides a simple proof that OWA achieves the optimal $O(\sqrt{d/mn})$ error. Our main condition is that the single machine parameter vectors have a "sufficiently Gaussian" distribution. We show that this is a mild condition known to hold in many situations of interest. Section 5 compares OWA to existing distributed algorithms. We highlight how the analysis of existing algorithms requires more limiting assumptions than OWA's. Section 6 shows empirically that OWA performs well on synthetic and real world advertising data. We demonstrate that

OWA is robust to the strength of regularization, which is one of the reasons it performs well in practice.

2 Problem Setting

Let $\mathcal{Y} \subseteq \mathbb{R}$ be the space of response variables, $\mathcal{X} \subseteq \mathbb{R}^d$ be the space of covariates, and $\mathcal{W} \subseteq \mathbb{R}^d$ be the parameter space. We assume a linear model where the loss of data point $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ given the parameter $\mathbf{w} \in \mathcal{W}$ is denoted by $\ell(y, \mathbf{x}^\mathsf{T}\mathbf{w})$. We define the true loss of parameter vector $\mathbf{w}$ to be $\mathcal{L}^*(\mathbf{w}) = \mathbb{E}\ell(y; \mathbf{x}^\mathsf{T}\mathbf{w})$, and the optimal parameter vector $\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{W}} \mathcal{L}^*(\mathbf{w})$. We do not require that the model be correctly specified, nor do we require that ℓ be convex with respect to $\mathbf{w}$. Let $Z \subset \mathcal{X} \times \mathcal{Y}$ be a dataset of mn i.i.d. observations. Finally, let $r : \mathcal{W} \to \mathbb{R}$ be a regularization function (typically the L1 or L2 norm) and $\lambda \in \mathbb{R}$ be the regularization strength. Then the regularized empirical risk minimizer (ERM) is

$$\hat{\mathbf{w}}^{erm} = \arg\min_{\mathbf{w}\in\mathcal{W}} \sum_{(\mathbf{x},y)\in Z} \ell(y, \mathbf{x}^\mathsf{T}\mathbf{w}) + \lambda r(\mathbf{w}). \tag{1}$$

Assume that the dataset Z has been partitioned onto m machines so that each machine i has dataset Z_i of size n, and all the Z_i are disjoint. Then each machine calculates the local ERM

$$\hat{\mathbf{w}}_i^{erm} = \arg\min_{\mathbf{w}\in\mathcal{W}} \sum_{(\mathbf{x},y)\in Z_i} \ell(y, \mathbf{x}^\mathsf{T}\mathbf{w}) + \lambda r(\mathbf{w}). \tag{2}$$

Notice that computing $\hat{\mathbf{w}}_i^{erm}$ requires no communication with other machines. Our goal is to merge the $\hat{\mathbf{w}}_i^{erm}$s into a single improved estimate.

To motivate our OWA merge procedure, we briefly describe a baseline procedure called *naive averaging*:

$$\hat{\mathbf{w}}^{ave} = \frac{1}{m}\sum_{i=1}^{m} \hat{\mathbf{w}}_i^{erm}. \tag{3}$$

Naive averaging is simple to compute but has only limited theoretical guarantees. Recall that the quality of an estimator $\hat{\mathbf{w}}$ can be measured by the estimation error $\|\hat{\mathbf{w}} - \mathbf{w}^*\|$, and we can use the triangle inequality to decompose this error as

$$\|\hat{\mathbf{w}} - \mathbf{w}^*\| \le \|\hat{\mathbf{w}} - \mathbb{E}\hat{\mathbf{w}}\| + \|\mathbb{E}\hat{\mathbf{w}} - \mathbf{w}^*\|. \tag{4}$$

We refer to $\|\hat{\mathbf{w}} - \mathbb{E}\hat{\mathbf{w}}\|$ as the variance of the estimator and $\|\mathbb{E}\hat{\mathbf{w}} - \mathbf{w}^*\|$ as the bias. McDonald et al. [17] show that the $\hat{\mathbf{w}}^{ave}$ estimator has lower variance than the estimator $\hat{\mathbf{w}}_i^{erm}$ trained on a single machine, but the same bias. Zhang et al. [25] extend this analysis to show that if $\hat{\mathbf{w}}_i^{erm}$ is a "nearly unbiased estimator," then naive averaging is optimal. But Rosenblatt and Nadler [22] show that in high dimensional regimes, all models are heavily biased, and so naive averaging is suboptimal. All three results require ℓ to be convex in addition to other technical assumptions. Our goal is to design a merging procedure that has good error bounds in a more general setting.

3 The OWA Estimator

The *optimal weighted average* (OWA) estimator uses a second round of optimization to calculate the optimal linear combination of the $\hat{\mathbf{w}}_i^{erm}$s. This second optimization reduces the bias at the optimal rate. Furthermore, this second optimization occurs over a small fraction of the dataset, so its computational and communication cost is negligible.

3.1 Warmup: The Full OWA

To motivate the OWA estimator, we first present a less efficient estimator that uses the full dataset for the second round of optimization. Define the matrix $\hat{W} : \mathbb{R}^{d \times m}$ to have its ith column equal to $\hat{\mathbf{w}}_i^{erm}$. Now consider the estimator

$$\hat{\mathbf{w}}^{owa,full} = \hat{W}\hat{\mathbf{v}}^{owa,full}, \tag{5}$$

where

$$\hat{\mathbf{v}}^{owa,full} = \operatorname*{arg\,min}_{\mathbf{v} \in \mathbb{R}^m} \sum_{(\mathbf{x},y) \in Z} \ell\left(y, \mathbf{x}^\mathsf{T}\hat{W}\mathbf{v}\right) + \lambda r(\hat{W}\mathbf{v}). \tag{6}$$

Notice that $\hat{\mathbf{w}}^{owa,full}$ is just the empirical risk minimizer when the parameter space $\mathcal{W}$ is restricted to the subspace $\hat{\mathcal{W}}^{owa} = \text{span}\{\hat{\mathbf{w}}_i^{erm}\}_{i=1}^m$. In other words, the $\hat{\mathbf{v}}^{owa,full}$ vector contains the optimal weights to apply to each $\hat{\mathbf{w}}_i^{erm}$ when averaging. Figure 1 shows graphically that no other estimator in $\hat{\mathcal{W}}^{owa}$ can have lower regularized empirical loss than $\hat{\mathbf{w}}^{owa,full}$.

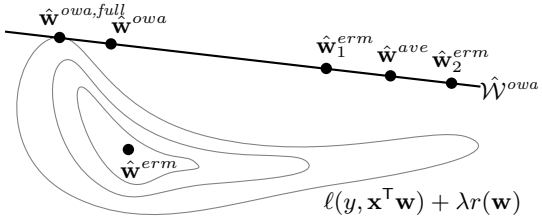

Fig. 1. $\hat{\mathbf{w}}^{owa,full}$ is the estimator with best loss in $\hat{\mathcal{W}}^{owa}$, and $\hat{\mathbf{w}}^{owa}$ is close with high probability.

3.2 The OWA Estimator

The OWA estimator uses fewer data points in the second round of optimization. Recall that in a linear model, the amount of data needed is proportional to the problem's dimension. Since the dimension of the second round is a fraction m/d

Algorithm 1 Calculating $\hat{\mathbf{w}}^{owa}$ only

Preconditions:
  each machine i already has dataset Z_i
  the master machine additionally has dataset Z^{owa}
Each machine i independently:
  calculates $\hat{\mathbf{w}}_i^{erm}$ using Equation (2)
  transmits $\hat{\mathbf{w}}_i^{erm}$ to the master
The master calculates $\hat{\mathbf{w}}^{owa}$ using Equation (7)
  (optionally) master uses approximation Equation (9)

smaller than the first round, only an m/d fraction of data is needed for the same accuracy. To simplify OWA's analysis in Section 4, we will assume here that this data is independent of the data used in the first round. This assumption, however, is an artifact of Section 4's simple analysis, and all our experiments in Section 6 reuse the same data for both optimizations.

Formally, let Z^{owa} be a set of m^2n/d additional data points sampled i.i.d. from the original data distribution. Thus the total amount of data the OWA estimator requires is $mn + m^2n/d$. Whenever $m/d \leq 1$, this expression simplifies to $O(mn)$, which is the same order of magnitude of data in the original problem. The OWA estimator is then defined as

$$\hat{\mathbf{w}}^{owa} = \hat{W}\hat{\mathbf{v}}^{owa}, \tag{7}$$

where

$$\hat{\mathbf{v}}^{owa} = \operatorname*{arg\,min}_{\mathbf{v}\in\mathbb{R}^m} \sum_{(\mathbf{x},y)\in Z^{owa}} \ell\left(y, \mathbf{x}^\mathsf{T}\hat{W}\mathbf{v}\right) + \lambda r(\hat{W}\mathbf{v}). \tag{8}$$

Algorithm 1 shows the procedure for calculating $\hat{\mathbf{w}}^{owa}$ in a distributed setting. Notice that we assume that a predesignated master machine already has access to the full Z^{owa} dataset.[3] Because this data is pre-assigned to the master machine, each machine i only needs to transmit the local parameter vector $\hat{\mathbf{w}}_i^{erm}$ to the master. Thus, the total number of bits communicated is $O(dm)$, which is the same as the naive averaging estimator. OWA's merge procedure is more complicated than the naive averaging merge procedure, but still very fast. Notice that the projected data points $\mathbf{x}^\mathsf{T}\hat{W}$ have dimensionality $m \ll d$, and there are only m^2n/d of them. Because the optimization uses a smaller dimension and fewer data points, it takes a negligible amount of time. In Section 6, we show an experiment where the first round of optimizations takes about a day, and the second optimization takes about a minute.

[3] Other non-interactive estimators have made similar assumptions [e.g. 28]. If this assumption is too limiting, however, Appendix A shows how to transfer these data points to the master machine after optimizing the local models. The idea is to first project the data onto the subspace $\hat{\mathcal{W}}^{owa}$ before transfer, reducing the dimensionality of the data. The communication complexity of this alternate procedure is $O(dm^2)$.

3.3 Implementing OWA with Existing Optimizers

In theory, standard optimization algorithms can be used to directly solve the second round of optimization in Equation (8). In practice, however, standard tools such as scikit-learn [21] do not support the regularization term $r(\hat{W}\mathbf{v})$, where the parameter vector is projected onto an alternative coordinate system before regularization. To make OWA easy to implement, we show in this section how to approximately solve (8) using these optimizers.

We suggest approximating the regularization term by L2 regularization directly on the $\mathbf{v}$ vector:

$$\lambda r(\hat{W}\mathbf{v}) \approx \lambda_2 \|\mathbf{v}\|^2, \tag{9}$$

where λ_2 is a new hyperparameter. We provide two justifications for this approximation:

1. When we want the parameter vector $\mathbf{w}$ to be sparse (and so the regularizer r is the L1 norm), we have no reason to believe that the $\mathbf{v}$ vector should be sparse. The desired sparsity is induced by the regularization when solving for $\hat{\mathbf{w}}_i^{erm}$s on the local machines, and it is maintained in any linear combination of the $\hat{\mathbf{w}}_i^{erm}$s.
2. As the size of the dataset increases, the importance of the regularizer decreases. In this second optimization, the dimensionality of the problem is small and the theory requires few data points, guaranteeing the optimization runs fast. If we can increase the number of data points by several orders of magnitude (say from m^2n/d to $100m^2n/d$), the optimization will remain fast in practice and the influence of the regularization term becomes negligible.

The new λ_2 regularization parameter should be set by cross validation. This is a fast procedure, however, because the second optimization has so little data. Furthermore, this cross validation can be computed locally on the master machine without any communication. We again emphasize that Section 6 contains experiments where the first round of optimization took about a day, and the second round (including the selection of λ_2) took only about a minute.

3.4 Fast Cross Validation for OWA

We now introduce a novel fast cross validation algorithm for estimating the predictive performance of OWA. The standard method for k-fold cross validation takes linear time in the number of folds k. For large scale problems, this is too computationally expensive, and so cross validation is typically not used in this regime. Our fast cross validation procedure can estimate the predictive performance of OWA in constant time (relative to k). This makes our procedure suitable for large scale problems. Our method has two restrictions. First, we require the number of folds k must be equal to the number of machines m. Second, we require each machine already have access to the full Z^{owa} dataset.

Our procedure uses two rounds of computation and is shown in Algorithm 2. The first round trains the local estimators $\hat{\mathbf{w}}_i^{erm}$ as in Algorithm 1, but then

Algorithm 2 Calculating $\hat{\mathbf{w}}^{owa}$ with fast cross validation

Preconditions:
  each machine i already has dataset Z_i
  each machine (not just the master) also has dataset Z^{owa}
Each machine i independently:
  calculates $\hat{\mathbf{w}}_i^{erm}$ using Equation (2)
  broadcasts $\hat{\mathbf{w}}_i^{erm}$ to all other machines
Each machine i independently:
  calculates $\hat{\mathbf{w}}_{-i}^{owa}$ using Equation (10)
  (optionally) $\hat{\mathbf{w}}_{-i}^{owa}$ calculated with approx. Eq. (9)
  computes $\widehat{\mathrm{err}}_i$ using Equation (12)
  transmits $\widehat{\mathrm{err}}_i$ to the master
The master
  computes $\hat{\mathbf{w}}^{owa}$ using Equation (7)
  computes $\frac{1}{m}\sum_{i=1}^{m}\widehat{\mathrm{err}}_i$

broadcasts these parameter vectors to all machines (rather than just the master). In the second round, each machine i calculates $\hat{\mathbf{w}}_{-i}^{owa}$, which is a version of the OWA estimator trained on the data from all the machines except machine i. More formally, we define the matrix $\hat{W}_{-i} : \mathbb{R}^{d\times(m-1)}$ to be the matrix $\hat{W}$ with ith column removed. That is, $\hat{W}_{-i}$ is the concatenation of the $\hat{\mathbf{w}}_j^{erm}$ vectors for all $j \neq i$. Then let $Z_{-i}^{owa} = \{Z_j\}_{j\neq i}$ be the data set used in the second round of optimization without the data points from machine i. Finally, define the estimator

$$\hat{\mathbf{w}}_{-i}^{owa} = \hat{W}_{-i}\hat{\mathbf{v}}_{-i}^{owa}, \tag{10}$$

where

$$\hat{\mathbf{v}}_{-i}^{owa} = \operatorname*{arg\,min}_{\mathbf{v}\in\mathbb{R}^{m-1}} \sum_{(\mathbf{x},y)\in Z_{-i}^{owa}} \ell\left(y, \mathbf{x}^{\mathsf{T}}\hat{W}_{-i}\mathbf{v}\right) + \lambda r(\hat{W}_{-i}\mathbf{v}). \tag{11}$$

Notice that $\hat{\mathbf{w}}_{-i}^{owa}$ does not depend on the local data set Z_i. So

$$\widehat{\mathrm{err}}_i = \frac{1}{n}\sum_{(\mathbf{x},y)\in Z_i} \ell(y, \mathbf{x}^{\mathsf{T}}\hat{\mathbf{w}}_{-i}^{owa}) \tag{12}$$

is an unbiased estimate of the true error $\mathcal{L}^*(\hat{\mathbf{w}}_{-i}^{owa})$. The algorithm then transmits the $\widehat{\mathrm{err}}_i$ values to the master machine, which computes $\hat{\mathbf{w}}^{owa}$ as normal and computes the average of the error estimates. In total, $O(dm^2)$ bits are transmitted in the first round, and $O(dm)$ bits in the second round. When compared with Algorithm 1, the fast cross validation method requires a factor of m times more communication, and approximately twice as much computation.

This fast cross validation procedure critically used the fact that the OWA estimator is non-interactive. Similar procedures can be developed for other non-interactive distributed learning algorithms, but this technique cannot be used to develop fast cross validation methods for interactive algorithms. OWA's fast cross validation procedure is closely related to the out-of-bag method [5], monoid fast cross validation [7], and incremental fast cross validation [10], but none of these previous methods was developed specifically for the distributed setting.

4 Analysis

A major advantage of OWA's analysis is that it requires only simple and general conditions. Essentially, we will prove that whenever ERM is an optimal estimator, then OWA is also optimal. In Section 5 below, we will see that previous methods require more complicated and less general conditions. In this section, we first describe our main condition in detail. Then we outline the argument that OWA's estimation error $\|\hat{\mathbf{w}}^{owa} - \mathbf{w}^*\|$ and generalization error $\mathcal{L}^*(\hat{\mathbf{w}}^{owa}) - \mathcal{L}^*(\mathbf{w}^*)$ both decay as $O(\sqrt{d/mn})$. Full proofs of all theorems are provided in Appendix B.

4.1 The Sub-Gaussian Tail (SGT) Condition

Recall that each estimator is a random vector that is a function of the data. Informally, our main condition is that these vectors follow an approximately Gaussian distribution. This is a mild condition that many statistical models are known to satisfy. For example, the estimated parameters for all generalized linear models (such as logistic regression and ordinary least squares regression) are known to be approximately Gaussian. We now formally define our criterion and describe in detail how to establish that it holds.

Definition 1 *We say that a statistical model satisfies the* sub-Gaussian tail (SGT) condition *if the emperical risk minimizer* $\hat{\mathbf{w}}$ *trained on* n *i.i.d. data points of dimension* d *has the sub-Gaussian estimation error*

$$\Pr\left[\|\hat{\mathbf{w}} - \mathbf{w}^*\| \leq O(\sqrt{dt/n})\right] \geq 1 - \exp(-t). \tag{13}$$

Remark 1. Notice that if $\hat{\mathbf{w}}$ has a Gaussian distribution it will satisfy the SGT condition, even if $\hat{\mathbf{w}}$ has arbitrary non-zero mean. (This is a standard property of sub-Gaussian distributions.) Thus, the SGT condition makes no assumptions about the model's bias.

A large body of statistical literature establishes the SGT condition for many models. Chapter 7 of Lehmann [13] provides an elementary introduction to results in the asymptotic regime as $n \to \infty$. Lehman requires only that the loss ℓ be three times differentiable, that the data points be i.i.d., and that $\mathbf{w}^*$ be identifiable. For example, models using the non-convex sigmoid loss satisfy these conditions, and thus can be used with the OWA estimator. Lehmann [13] also contains references to stronger asymptotic results that relax these already mild conditions.

Other work establishes the SGT condition in the non-asymptotic regime $n < \infty$. Panov et al. [20] provides a particularly strong example. Their only condition is that the empirical loss admit a local approximation via the so-called *bracketing device*, which can be thought of as a generalization of the Taylor expansion. The full explanation of this condition is rather technical, but we highlight that this result does not require a convex loss or even that the data be i.i.d.

The proofs of theorems establishing the SGT condition are typically long and technical. In our view, a limitation of previous non-interactive estimators is

that their analysis proves limited forms of the SGT condition from scratch. This makes their proofs long and technical as well. It also limits the applicability of their results, because they do not prove the more general versions of the SGT condition cited above. Our work improves on this practice by "factoring out" these technical details. By relying on this established body of literature to prove the SGT condition for us, we get simpler proofs that apply more generally. In particular, we essentially conclude that whenever the ERM estimator successfully learns on a single machine (i.e. the SGT condition holds), then the OWA estimator successfully learns in a distributed environment. No other distributed estimator (interactive or non-interactive) can make such a strong claim.

4.2 The Main Idea: $\hat{\mathcal{W}}^{owa}$ Contains Good Solutions

The most important idea of OWA's analysis is to show that when the local $\hat{\mathbf{w}}_i^{erm}$ estimators satisfy the SGT condition, then $\hat{\mathcal{W}}^{owa}$ is a good subspace to optimize over. In particular, if we let $\pi_{\hat{\mathcal{W}}^{owa}}\mathbf{w}^*$ denote the projection of $\mathbf{w}^*$ onto $\hat{\mathcal{W}}^{owa}$, then we have that $\pi_{\hat{\mathcal{W}}^{owa}}\mathbf{w}^* \approx \mathbf{w}^*$. This idea is formalized in the following lemma.

Lemma 2. *Assume the model satisfies the SGT condition. Let $t > 0$. Then with probability at least $1 - \exp(-t)$,*

$$\|\pi_{\hat{\mathcal{W}}^{owa}}\mathbf{w}^* - \mathbf{w}^*\| \leq O(\sqrt{dt/mn}). \tag{14}$$

The proof of Lemma 2 is a direct consequence of the SGT condition.

4.3 Bounding the Generalization Error

In order to connect the result of Lemma 1 to OWA's generalization error, we need to introduce a smoothness condition on the true loss function $\mathcal{L}^*$. Lipschitz continuity is a widely used technique in both convex and non-convex analysis.

Definition 2 *We say that $\mathcal{L}^*$ is* β-Lipschitz *continuous if for all $\mathbf{w}_1$ and $\mathbf{w}_2$,*

$$|\mathcal{L}^*(\mathbf{w}_1) - \mathcal{L}^*(\mathbf{w}_2)| \leq \beta\|\mathbf{w}_1 - \mathbf{w}_2\|. \tag{15}$$

We now state our first main result, which guarantees that OWA will generalize well.

Theorem 3. *Assume the model satisfies the SGT condition, and that $\mathcal{L}^*$ is β-Lipschitz continuous. Let $t > 0$. Then with probability at least $1 - \exp(-t)$,*

$$\mathcal{L}^*(\hat{\mathbf{w}}^{owa}) - \mathcal{L}^*(\mathbf{w}^*) \leq O(\beta\sqrt{dt/mn}). \tag{16}$$

4.4 Bounding the Estimation Error

To bound the estimation error, we introduce a quadratic restriction on the growth of the true loss $\mathcal{L}^*$.

Definition 3 *We say the true loss $\mathcal{L}^*$ satisfies the* lower quadratic growth (lower QG) condition *if for all points $\mathbf{w} \in \mathcal{W}$,*

$$\alpha_{lo}\|\mathbf{w} - \mathbf{w}^*\|^2 \leq \mathcal{L}^*(\mathbf{w}) - \mathcal{L}^*(\mathbf{w}^*). \tag{17}$$

We say that $\mathcal{L}^*$ satisfies the upper quadratic growth (upper QG) condition *if it satisfies*

$$\mathcal{L}^*(\mathbf{w}) - \mathcal{L}^*(\mathbf{w}^*) \leq \alpha_{hi}\|\mathbf{w} - \mathbf{w}^*\|^2. \tag{18}$$

The lower QG condition has previously been used to study the convergence of non-convex optimization [e.g. 3, 1]. This condition is a generalization of strong convexity that needs to hold only at the optimum $\mathbf{w}^*$ rather than all points in the domain. In particular, functions satisfying the lower QG condition may have many local minima with different objective values. Karimi et al. [11] compares the lower QG condition to six related generalizations of convexity, and shows that the QG condition is the weakest of these conditions in the sense that it is implied by all other conditions.

The intuitive meaning of the lower and upper QG conditions is that a quadratic function can be used to lower and upper bound $\mathcal{L}^*$. As the domain $\mathcal{W}$ shrinks to include only the optimal point $\mathbf{w}^*$, these lower and upper bounds converge to the Taylor expansion of $\mathcal{L}^*$. In this limit, the constant α_{lo} is the minimum eigenvalue of the Hessian at $\mathbf{w}^*$, and α_{hi} is the maximum eigenvalue. The ratio $\alpha_{\text{hi}}/\alpha_{\text{lo}}$ can then be thought of as a generalized condition number.

Our main result is:

Theorem 4. *Assume the SGT condition and that that $\mathcal{L}^*$ satisfies the lower and upper QG conditions. Let $t > 0$. Then with probability at least $1 - \exp(-t)$,*

$$\|\hat{\mathbf{w}}^{owa} - \mathbf{w}^*\| \leq O\left(\sqrt{(\alpha_{hi}/\alpha_{lo})(dt/mn)}\right). \tag{19}$$

Note that up to the constant factor $\sqrt{\alpha_{\text{hi}}/\alpha_{\text{lo}}}$, OWA's estimation error matches that of the oracle ERM.

5 Other Non-Interactive Estimators

Compared with similar non-interactive distributed estimators, OWA either has stronger statistical guarantees, is applicable to more models, or has a more computationally efficient merging procedure.

Lee et al. [12] and Battey et al. [2] independently develop closed form formulas for debiasing L1 regularized least squares regressions. They combine these debiased estimators with the averaging estimator to create a non-interactive estimator that reduces both bias and variance at the optimal rate. OWA's advantage over these methods is that it is that it can be applied to a much larger class of problems.

Jordan et al. [9] develop a more general approach that uses a single approximate Newton step in the merge procedure. As long as the initial starting point (they suggest using $\hat{\mathbf{w}}^{ave}$) is within $O(\sqrt{1/n})$ of the true parameter vector, then

this approach converges at the optimal rate. When implementing Jordan et al.'s approach, we found it suffered from two practical difficulties. First, Newton steps can diverge if the starting point is not close enough. We found in our experiments that $\hat{\mathbf{w}}^{ave}$ was not always close enough. Second, Newton steps require inverting a Hessian matrix. In Section 6, we consider a problem with dimension $d \approx 7 \times 10^5$; the corresponding Hessian is too large to practically invert. For these reasons, we do not compare against Jordan et al. [9] in our experiments.

Zhang et al. [25] provide a debiasing technique that works for any estimator. Let $s \in (0, 1)$, and Z_i^s be a bootstrap sample of Z_i of size sn. Then the bootstrap average estimator is

$$\hat{\mathbf{w}}^{boot} = \frac{\hat{\mathbf{w}}^{ave} - s\hat{\mathbf{w}}^{ave,s}}{1-s}, \tag{20}$$

where

$$\hat{\mathbf{w}}^{ave,s} = \frac{1}{m}\sum_{i=1}^{m} \arg\min_{\mathbf{w}} \sum_{(\mathbf{x},y)\in Z_i^s} \ell(y, \mathbf{x}^\mathsf{T}\mathbf{w}) + \lambda r(\mathbf{w}).$$

The intuition behind this estimator is to use the bootstrap sample to directly estimate and correct for the bias. When the loss function is convex, $\hat{\mathbf{w}}^{boot}$ enjoys a mean squared error (MSE) that decays as $O((mn)^{-1} + n^{-3})$. Theorem 2 directly implies that the MSE of $\hat{\mathbf{w}}^{owa}$ decays as $O((mn)^{-1})$ under more general conditions. There are two additional limitations to $\hat{\mathbf{w}}^{boot}$. First, the optimal value of s is not obvious and setting the parameter requires cross validation on the entire data set. Our proposed $\hat{\mathbf{w}}^{owa}$ estimator has a similar parameter λ_2 that needs tuning, but this tuning happens on a small fraction of the data and always with the L2 regularizer. So properly tuning λ_2 is more efficient than s. Second, performing a bootstrap on an unbiased estimator increases the variance. This means that $\hat{\mathbf{w}}^{boot}$ could perform worse than $\hat{\mathbf{w}}^{ave}$ on unbiased estimators. Our $\hat{\mathbf{w}}^{owa}$ estimator, in contrast, will perform at least as well as $\hat{\mathbf{w}}^{ave}$ with high probability even for highly biased estimators (see Figure 1). The next section shows that $\hat{\mathbf{w}}^{owa}$ has better empirical performance than $\hat{\mathbf{w}}^{boot}$.

Liu and Ihler [15] propose a more Bayesian approach. Instead of averaging the model's parameters, they directly "average the models" with the following KL-average estimator:

$$\hat{\mathbf{w}}^{kl} = \arg\min_{\mathbf{w}\in\mathcal{W}} \sum_{i=1}^{m} \mathrm{KL}\left(p(\cdot; \hat{\mathbf{w}}_i^{erm}) \,\middle\|\, p(\cdot; \mathbf{w})\right). \tag{21}$$

Liu and Ihler show theoretically that this is the best merge function in the class of functions that do not depend on the data. Since OWA's merge depends on the data, however, this bound does not apply. The main disadvantage of KL-averaging is computational. The minimization in (21) is performed via a bootstrap sample from the local models, which is computationally expensive. Let k be the size of the bootstrap sample. Then Liu and Ihler's method has MSE that shrinks as $O((mn)^{-1} + k^{-1})$. This implies that the bootstrap procedure requires as many samples as the original problem to get a MSE that shrinks at the same rate as the averaging estimator. Han and Liu [6] provide a method to reduce the

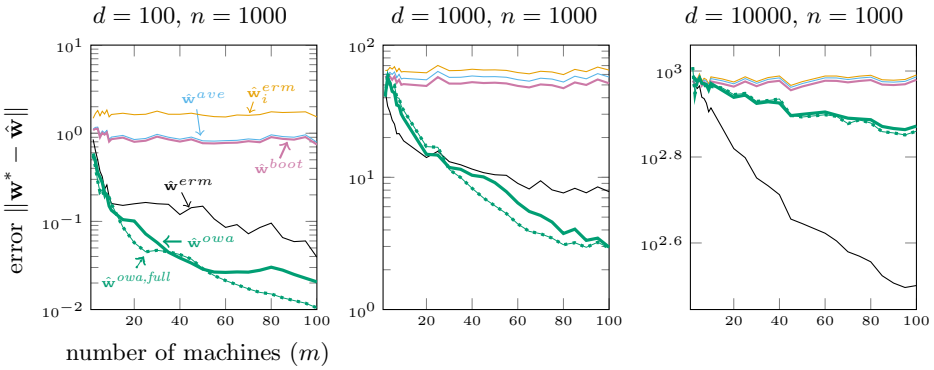

Fig. 2. The left figure shows scalability in the low dimension regime, the middle figure in a medium dimension regime, and the right figure in a high dimension regime. $\hat{\mathbf{w}}^{owa}$ scales well with the number of machines in all cases. Surprisingly, $\hat{\mathbf{w}}^{owa}$ outperforms the oracle estimator trained on all of the data $\hat{\mathbf{w}}^{erm}$ in some situations.

MSE to $O((mn)^{-1} + (n^2k)^{-1})$ using control variates, but the procedure remains prohibitively expensive. Their experiments show the procedure scaling only to datasets of size $mn \approx 10^4$, whereas our experiments involve a dataset of size $mn \approx 10^8$.

6 Experiments

We evaluate OWA on synthetic and real-world logistic regression tasks. In each experiment, we compare $\hat{\mathbf{w}}^{owa}$ with four baseline estimators: the naive estimator using the data from only a single machine $\hat{\mathbf{w}}_i^{erm}$; the averaging estimator $\hat{\mathbf{w}}^{ave}$; the bootstrap estimator $\hat{\mathbf{w}}^{boot}$; and the oracle estimator of all data trained on a single machine $\hat{\mathbf{w}}^{erm}$. The $\hat{\mathbf{w}}^{boot}$ estimator has a parameter s that needs to be tuned. In all experiments we evaluate $\hat{\mathbf{w}}^{boot}$ with $s \in \{0.005, 0.01, 0.02, 0.04, 0.1, 0.2\}$, which is a set recommended in the original paper [25], and then report only the value of s with highest true likelihood. Thus we are reporting an overly optimistic estimate of the performance of $\hat{\mathbf{w}}^{boot}$, and as we shall see $\hat{\mathbf{w}}^{owa}$ still tends to perform better. OWA is always trained using the regularization approximation of Section 3.3, and Z^{owa} is always resampled from the original dataset.

In all experiments, we use the scikit-learn machine learning library [21] to perform the optimizations. We made no special efforts to tune parameters of the optimization routines. For example, all optimizations are performed with the default target accuracy of 1×10^{-3}. Additionally, when performing the hyperparameter optimization for λ_2 in (9), we use the default hyperparameter selection procedure.

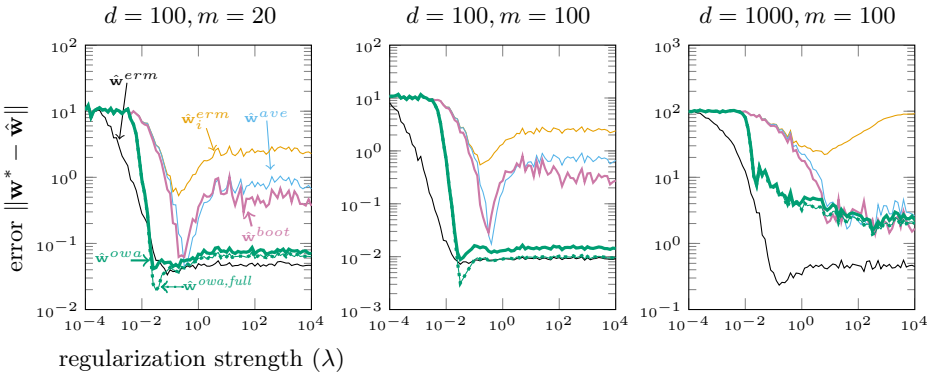

Fig. 3. OWA is robust to the regularization strength used to solve $\hat{\mathbf{w}}_i^{erm}$. Our theory states that as $m \to d$, we have that $\hat{\mathcal{W}}^{owa} \to \mathcal{W}$, and so $\hat{\mathbf{w}}^{owa} \to \hat{\mathbf{w}}^{erm}$. This is confirmed in the middle experiment. In the left experiment, $m < d$, but $\hat{\mathbf{w}}^{owa}$ still behaves similarly to $\hat{\mathbf{w}}^{erm}$. In the right experiment, $\hat{\mathbf{w}}^{owa}$ has similar performance as $\hat{\mathbf{w}}^{ave}$ and $\hat{\mathbf{w}}^{boot}$ but over a wider range of λ values.

6.1 Synthetic Data

We generate the data according to a sparse logistic regression model. Each component of $\mathbf{w}^*$ is sampled i.i.d. from a spike and slab distribution. With probability 0.9, it is 0; with probability 0.1, it is sampled from a standard normal distribution. The data points are then sampled as

$$\mathbf{x}_i \sim \mathcal{N}(0, I) \tag{22}$$

$$y_i \sim \text{Bernoulli}\left(1/\left(1 + \exp(-\mathbf{x}_i^\mathsf{T}\mathbf{w}^*)\right)\right). \tag{23}$$

The primary advantage of synthetic data is that we know the model's true parameter vector. So for each estimator $\hat{\mathbf{w}}$ that we evaluate, we can directly calculate the error $\|\hat{\mathbf{w}} - \mathbf{w}^*\|$. We run two experiments on the synthetic data. In both experiments, we use the L1 regularizer to induce sparsity in our estimates of $\mathbf{w}^*$. Results are qualitatively similar when using a Laplace, Gaussian, or uniform prior on $\mathbf{w}^*$, and with L2 regularization.

Our first experiment shows how the estimators scale as the number of machines m increases. We fix $n = 1000$ data points per machine, so the size of the dataset mn grows as we add more machines. This simulates the typical "big data" regime where data is abundant, but processing resources are scarce. For each value of m, we generate 50 datasets and report the average of the results. The results are shown in Figure 2. As the analysis predicted, the performance of $\hat{\mathbf{w}}^{owa}$ scales much better than $\hat{\mathbf{w}}^{ave}$ and $\hat{\mathbf{w}}^{boot}$. Surprisingly, in the low dimensional regimes, $\hat{\mathbf{w}}^{owa}$ outperforms the single machine oracle $\hat{\mathbf{w}}^{erm}$.

Our second experiment shows the importance of proper λ selection. We evaluate the performance of the estimators with λ varying from 10^{-4} to 10^4 on a grid of 80 points. Figure 3 shows the results. The $\hat{\mathbf{w}}^{owa}$ estimator is more robust

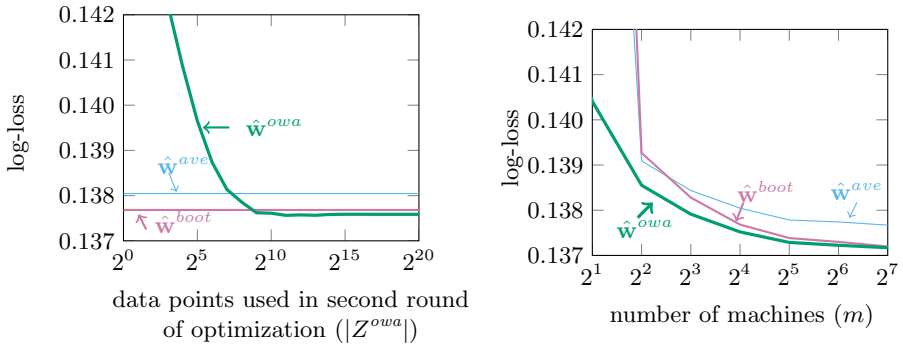

Fig. 4. (*left*) Relatively few data points are needed in the second round of optimization for $\hat{\mathbf{w}}^{owa}$ to converge. On this dataset, only 2.7×10^{-6} percent of the data is needed. (*right*) Performance of the parallel estimators on advertising data as the number of machines m increases.

to the choice of λ than the other distributed estimators. We suspect that slight misspecification of λ in the first round of optimization is compensated for in the second round of optimization.

6.2 Real World Advertising Data

We evaluate the estimators on real world data from the KDD 2012 Cup [19]. The goal is to predict whether a user will click on an ad from the Tencent internet search engine. This dataset was previously used to evaluate the performance of $\hat{\mathbf{w}}^{boot}$ [25]. This dataset is too large to fit on a single machine, so we must use distributed estimators, and we do not provide results of the oracle estimator $\hat{\mathbf{w}}^{erm}$ in our figures. There are 235,582,879 distinct data points, each of dimension 741,725. The data points are sparse, so we use the L1 norm to encourage sparsity in our final solution. The regularization strength was set using cross validation in the same manner as for the synthetic data. For each test, we split the data into 80 percent training data and 20 percent test data. The training data is further subdivided into 128 partitions, one for each of the machines used. It took about 1 day to train the local model on each machine in our cluster.

Our first experiment measures the importance of the number of data points used in the second optimization (i.e. $|Z^{owa}|$). We fix $m = 128$, and allow $|Z^{owa}|$ to vary from 2^0 to 2^{20}. When $|Z^{owa}| = 2^{20}$, almost the entire dataset is used in the second optimization. We repeated the experiment 50 times, each time using a different randomly selected set Z^{owa} for the second optimization. Figure 4 (*left*) shows the results. Our $\hat{\mathbf{w}}^{owa}$ estimator has lower loss than $\hat{\mathbf{w}}^{ave}$ using only $|Z^{owa}| = 2^{15}$ data points (approximately 4×10^{-8} percent of the full training set) and $\hat{\mathbf{w}}^{owa}$ has converged to its final loss value with only $|Z^{owa}| = 2^{17}$ data points (approximately 2.7×10^{-6} percent of the full training set). This justifies our claim that only a small number of data points are needed for the second round of optimization. The computation is also very fast due to the lower dimensionality

and L2 regularization in the second round of optimization. When $|Z^{owa}| = 2^{17}$, computing the merged model took about a minute (including the cross validation time to select λ_2). This time is negligible compared to the approximately 1 day it took to train the models on the individual machines.

Our last experiment shows the performance as we scale the number of machines m. The results are shown in Figure 4 (*right*). Here, $\hat{\mathbf{w}}^{owa}$ performs especially well with low m. For large m, $\hat{\mathbf{w}}^{owa}$ continues to slightly outperform $\hat{\mathbf{w}}^{boot}$ without the need for an expensive model selection procedure to determine the s parameter.

7 Conclusion

We introduced OWA, a non-interactive distributed estimator for linear models. OWA is easy to implement and has optimal statistical guarantees that hold under general conditions. We showed experimentally that OWA outperforms other non-interactive estimators, and in particular that OWA exhibits a weaker dependence on the regularization strength.

Acknowledgments

Shelton was supported by the National Science Foundation (IIS 1510741).

References

1. Mihai Anitescu. Degenerate nonlinear programming with a quadratic growth condition. *SIAM Journal on Optimization*, 10(4):1116–1135, 2000.
2. Heather Battey, Jianqing Fan, Han Liu, Junwei Lu, and Ziwei Zhu. Distributed estimation and inference with statistical guarantees. *arXiv preprint arXiv:1509.05457*, 2015.
3. Joseph Frédéric Bonnans and Alexander Ioffe. Second-order sufficiency and quadratic growth for nonisolated minima. *Mathematics of Operations Research*, 20 (4):801–817, 1995.
4. Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.
5. Leo Breiman. Out-of-bag estimation, 1996. Technical report.
6. Jun Han and Qiang Liu. Bootstrap model aggregation for distributed statistical learning. *NeurIPS*, 2016.
7. Mike Izbicki. Algebraic classifiers: a generic approach to fast cross-validation, online training, and parallel training. *ICML*, 2013.
8. Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *NeurIPS*, pages 3068–3076, 2014.
9. Michael I. Jordan, Jason D. Lee, and Yun Yang. Communication-efficient distributed statistical inference. *arXiv preprint arXiv:1605.07689*, 2016.

10. Pooria Joulani, András György, and Csaba Szepesvári. Fast cross-validation for incremental learning. In *IJCAI*, pages 3597–3604, 2015.
11. Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.
12. Jason D Lee, Qiang Liu, Yuekai Sun, and Jonathan E Taylor. Communication-efficient sparse regression. *JMLR*, 18(5):1–30, 2017.
13. Erich Leo Lehmann. *Elements of large-sample theory*. Springer Science & Business Media, 1999.
14. Mu Li, David G Andersen, and Jun Woo Park. Scaling distributed machine learning with the parameter server. In *OSDI*, 2014.
15. Qiang Liu and Alexander T Ihler. Distributed estimation, information loss and exponential families. In *NeurIPS*, pages 1098–1106, 2014.
16. Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I Jordan, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. *International Conference of Machine Learning*, 2015.
17. Ryan McDonald, Mehryar Mohri, Nathan Silberman, Dan Walker, and Gideon S Mann. Efficient large-scale distributed training of conditional maximum entropy models. In *NeurIPS*, pages 1231–1239, 2009.
18. H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *AISTATS*, 2017.
19. Yanzhi Niu, Yi Wang, Gordon Sun, Aden Yue, Brian Dalessandro, Claudia Perlich, and Ben Hamner. The tencent dataset and KDD-cup'12. 2012.
20. Maxim Panov, Vladimir Spokoiny, et al. Finite sample Bernstein–von Mises theorem for semiparametric problems. *Bayesian Analysis*, 10(3):665–710, 2015.
21. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011.
22. Jonathan D Rosenblatt and Boaz Nadler. On the optimality of averaging in distributed statistical learning. *Information and Inference*, 5(4):379–404, 2016.
23. Virginia Smith, Simone Forte, Ma Chenxin, Martin Takáč, Michael I Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *JMLR*, 18:230, 2018.
24. Shusen Wang. A sharper generalization bound for divide-and-conquer ridge regression. *AAAI*, 2019.
25. Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. In *NeurIPS*, pages 1502–1510, 2012.
26. Yuchen Zhang, John C Duchi, and Martin J Wainwright. Divide and conquer kernel ridge regression. In *COLT*, 2013.
27. Shen-Yi Zhao, Ru Xiang, Ying-Hao Shi, Peng Gao, and Wu-Jun Li. Scope: Scalable composite optimization for learning on spark. *AAAI*, 2017.
28. Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *NeurIPS*, pages 2595–2603, 2010.