Journal of Machine Learning Research 13 (2012) 165-202 Submitted 11/10, Revised 10/11; Published 1/12

Optimal Distributed Online Prediction Using Mini-Batches

Ofer Dekel OFERD@MICROSOFT.COM
Ran Gilad-Bachrach RANG@MICROSOFT.COM
Microsoft Research
1 Microsoft Way
Redmond, WA 98052, USA

Ohad Shamir OHADSH@MICROSOFT.COM
Microsoft Research
1 Memorial Drive
Cambridge, MA 02142, USA

Lin Xiao LIN.XIAO@MICROSOFT.COM
Microsoft Research
1 Microsoft Way
Redmond, WA 98052, USA

Editor: Tong Zhang

Abstract

Online prediction methods are typically presented as serial algorithms running on a single processor. However, in the age of web-scale prediction problems, it is increasingly common to encounter situations where a single processor cannot keep up with the high rate at which inputs arrive. In this work, we present the *distributed mini-batch* algorithm, a method of converting many serial gradient-based online prediction algorithms into distributed algorithms. We prove a regret bound for this method that is asymptotically optimal for smooth convex loss functions and stochastic inputs. Moreover, our analysis explicitly takes into account communication latencies between nodes in the distributed environment. We show how our method can be used to solve the closely-related distributed stochastic optimization problem, achieving an asymptotically linear speed-up over multiple processors. Finally, we demonstrate the merits of our approach on a web-scale online prediction problem.

Keywords: distributed computing, online learning, stochastic optimization, regret bounds, convex optimization

1. Introduction

Many natural prediction problems can be cast as stochastic online prediction problems. These are often discussed in the serial setting, where the computation takes place on a single processor. However, when the inputs arrive at a high rate and have to be processed in real time, there may be no choice but to distribute the computation across multiple cores or multiple cluster nodes. For example, modern search engines process thousands of queries a second, and indeed they are implemented as distributed algorithms that run in massive data-centers. In this paper, we focus on such *large-scale* and *high-rate* online prediction problems, where parallel and distributed computing is critical to providing a real-time service.

First, we begin by defining the stochastic online prediction problem. Suppose that we observe a stream of inputs $z_1, z_2, \ldots$, where each z_i is sampled independently from a fixed unknown distribution over a sample space $\mathcal{Z}$. Before observing each z_i, we predict a point w_i from a set W. After making the prediction w_i, we observe z_i and suffer the loss $f(w_i, z_i)$, where f is a predefined loss function. Then we use z_i to improve our prediction mechanism for the future (e.g., using a stochastic gradient method). The goal is to accumulate the smallest possible loss as we process the sequence of inputs. More specifically, we measure the quality of our predictions using the notion of *regret*, defined as

$$R(m) = \sum_{i=1}^{m} \left(f(w_i, z_i) - f(w^\star, z_i) \right) ,$$

where $w^\star = \arg\min_{w \in W} \mathbb{E}_z[f(w,z)]$. Regret measures the difference between the cumulative loss of our predictions and the cumulative loss of the fixed predictor $w^\star$, which is optimal with respect to the underlying distribution. Since regret relies on the stochastic inputs z_i, it is a random variable. For simplicity, we focus on bounding the expected regret $\mathbb{E}[R(m)]$, and later use these results to obtain high-probability bounds on the actual regret. In this paper, we restrict our discussion to convex prediction problems, where the loss function $f(w,z)$ is convex in w for every $z \in \mathcal{Z}$, and W is a closed convex subset of $\mathbb{R}^n$.

Before continuing, we note that the stochastic online *prediction* problem is closely related, but not identical, to the stochastic *optimization* problem (see, e.g., Wets, 1989; Birge and Louveaux, 1997; Nemirovski et al., 2009). The main difference between the two is in their goals: in stochastic optimization, the goal is to generate a sequence $w_1, w_2, \ldots$ that quickly converges to the minimizer of the function $F(\cdot) = \mathbb{E}_z[f(\cdot, z)]$. The motivating application is usually a static (batch) problem, and not an online process that occurs over time. Large-scale static optimization problems can always be solved using a serial approach, at the cost of a longer running time. In online prediction, the goal is to generate a sequence of predictions that accumulates a small loss along the way, as measured by regret. The relevant motivating application here is providing a real-time service to users, so our algorithm must keep up with the inputs as they arrive, and we cannot choose to slow down. In this sense, distributed computing is critical for large-scale online prediction problems. Despite these important differences, our techniques and results can be readily adapted to the stochastic online optimization setting.

We model our distributed computing system as a set of k *nodes*, each of which is an independent processor, and a *network* that enables the nodes to communicate with each other. Each node receives an incoming stream of examples from an outside source, such as a load balancer/splitter. As in the real world, we assume that the network has a limited bandwidth, so the nodes cannot simply share all of their information, and that messages sent over the network incur a non-negligible latency. However, we assume that network operations are *non-blocking*, meaning that each node can continue processing incoming traffic while network operations complete in the background.

How well can we perform in such a distributed environment? At one extreme, an ideal (but unrealistic) solution to our problem is to run a serial algorithm on a single "super" processor that is k times faster than a standard node. This solution is optimal, simply because any distributed algorithm can be simulated on a fast-enough single processor. It is well-known that the optimal regret bound that can be achieved by a gradient-based serial algorithm on an arbitrary convex loss is $O(\sqrt{m})$ (e.g., Nemirovski and Yudin, 1983; Cesa-Bianchi and Lugosi, 2006; Abernethy et al., 2009). At the other extreme, a trivial solution to our problem is to have each node operate in isolation of the other $k-1$ nodes, running an independent copy of a serial algorithm, without any communication over the network. We call this the *no-communication* solution. The main disadvantage of this solution is that the performance guarantee, as measured by regret, scales poorly with the network size k. More specifically, assuming that each node processes m/k inputs, the expected regret per node is $O(\sqrt{m/k})$. Therefore, the total regret across all k nodes is $O(\sqrt{km})$ - namely, a factor of $\sqrt{k}$ worse than the ideal solution. The first sanity-check that any distributed online prediction algorithm must pass is that it outperforms the naïve no-communication solution.

In this paper, we present the *distributed mini-batch* (DMB) algorithm, a method of converting any serial gradient-based online prediction algorithm into a parallel or distributed algorithm. This method has two important properties:

- It can use any gradient-based update rule for serial online prediction as a black box, and convert it into a parallel or distributed online prediction algorithm.

- If the loss function $f(w,z)$ is smooth in w (see the precise definition in Equation (5)), then our method attains an asymptotically optimal regret bound of $O(\sqrt{m})$. Moreover, the coefficient of the dominant term $\sqrt{m}$ is the same as in the serial bound, and *independent* of k and of the network topology.

The idea of using mini-batches in stochastic and online learning is not new, and has been previously explored in both the serial and parallel settings (see, e.g., Shalev-Shwartz et al., 2007; Gimpel et al., 2010). However, to the best of our knowledge, our work is the first to use this idea to obtain such strong results in a parallel and distributed learning setting (see Section 7 for a comparison to related work).

Our results build on the fact that the optimal regret bound for serial stochastic gradient-based prediction algorithms can be refined if the loss function is smooth. In particular, it can be shown that the hidden coefficient in the $O(\sqrt{m})$ notation is proportional to the standard deviation of the stochastic gradients evaluated at each predictor w_i (Juditsky et al., 2011; Lan, 2009; Xiao, 2010). We make the key observation that this coefficient can be effectively reduced by averaging a mini-batch of stochastic gradients computed at the same predictor, and this can be done in parallel with simple network communication. However, the non-negligible communication latencies prevent a straightforward parallel implementation from obtaining the optimal serial regret bound.[1] In order to close the gap, we show that by letting the mini-batch size grow slowly with m, we can attain the optimal $O(\sqrt{m})$ regret bound, where the dominant term of order $\sqrt{m}$ is *independent* of the number of nodes k and of the latencies introduced by the network.

The paper is organized as follows. In Section 2, we present a template for stochastic gradient-based serial prediction algorithms, and state refined variance-based regret bounds for smooth loss functions. In Section 3, we analyze the effect of using mini-batches in the serial setting, and show that it does not significantly affect the regret bounds. In Section 4, we present the DMB algorithm, and show that it achieves an asymptotically optimal serial regret bound for smooth loss functions. In Section 5, we show that the DMB algorithm attains the optimal rate of convergence for stochastic optimization, with an asymptotically linear speed-up. In Section 6, we complement our theoretical results with an experimental study on a realistic web-scale online prediction problem. While substantiating the effectiveness of our approach, our empirical results also demonstrate some interesting

1. For example, if the network communication operates over a minimum-depth spanning tree and the diameter of the network scales as $\log(k)$, then we can show that a straightforward implementation of the idea of parallel variance reduction leads to an $O\big(\sqrt{m\log(k)}\big)$ regret bound. See Section 4 for details.

Algorithm 1: Template for a serial first-order stochastic online prediction algorithm.

```
for $j = 1, 2, \dots$ do
    predict $w_j$
    receive input $z_j$ sampled i.i.d. from unknown distribution
    suffer loss $f(w_j, z_j)$
    define $g_j = \nabla_w f(w_j, z_j)$
    compute $(w_{j+1}, a_{j+1}) = \phi(a_j, g_j, \alpha_j)$
end
```

properties of mini-batching that are not reflected in our theory. We conclude with a comparison of our methods to previous work in Section 7, and a discussion of potential extensions and future research in Section 8. The main topics presented in this paper are summarized in Dekel et al. (2011). Dekel et al. (2011) also present robust variants of our approach, which are resilient to failures and node heterogeneity in an asynchronous distributed environment.

2. Variance Bounds for Serial Algorithms

Before discussing distributed algorithms, we must fully understand the serial algorithms on which they are based. We focus on gradient-based optimization algorithms that follow the template outlined in Algorithm 1. In this template, each prediction is made by an unspecified *update rule*:

$$(w_{j+1}, a_{j+1}) = \phi(a_j, g_j, \alpha_j). \tag{1}$$

The update rule ϕ takes three arguments: an auxiliary state vector a_j that summarizes all of the necessary information about the past, a gradient g_j of the loss function $f(\cdot, z_j)$ evaluated at w_j, and an iteration-dependent parameter α_j such as a stepsize. The update rule outputs the next predictor $w_{j+1} \in W$ and a new auxiliary state vector a_{j+1}. Plugging in different update rules results in different online prediction algorithms. For simplicity, we assume for now that the update rules are deterministic functions of their inputs.

As concrete examples, we present two well-known update rules that fit the above template. The first is the *projected gradient descent* update rule,

$$w_{j+1} = \pi_W\left(w_j - \frac{1}{\alpha_j} g_j\right), \tag{2}$$

where π_W denotes the Euclidean projection onto the set W. Here $1/\alpha_j$ is a decaying learning rate, with α_j typically set to be $\Theta(\sqrt{j})$. This fits the template in Algorithm 1 by defining a_j to simply be w_j, and defining ϕ to correspond to the update rule specified in Equation (2). We note that the projected gradient method is a special case of the more general class of *mirror descent* algorithms (e.g., Nemirovski et al., 2009; Lan, 2009), which all fit in the template of Equation (1).

Another family of update rules that fit in our setting is the *dual averaging* method (Nesterov, 2009; Xiao, 2010). A dual averaging update rule takes the form

$$w_{j+1} = \operatorname*{arg\,min}_{w \in W} \left\{ \left\langle \sum_{i=1}^{j} g_i, w \right\rangle + \alpha_j h(w) \right\}, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ denotes the vector inner product, $h : W \to \mathbb{R}$ is a strongly convex auxiliary function, and α_j is a monotonically increasing sequence of positive numbers, usually set to be $\Theta(\sqrt{j})$. The dual averaging update rule fits the template in Algorithm 1 by defining a_j to be $\sum_{i=1}^{j} g_i$. In the special case where $h(w) = (1/2)\|w\|_2^2$, the minimization problem in Equation (3) has the closed-form solution

$$w_{j+1} = \pi_W \left(-\frac{1}{\alpha_j} \sum_{i=1}^{j} g_j \right) . \tag{4}$$

For stochastic online prediction problems with convex loss functions, both of these update rules have expected regret bound of $O(\sqrt{m})$. In general, the coefficient of the dominant $\sqrt{m}$ term is proportional to an upper bound on the expected norm of the stochastic gradient (e.g., Zinkevich, 2003). Next we present refined bounds for smooth convex loss functions, which enable us to develop optimal distributed algorithms.

2.1 Optimal Regret Bounds for Smooth Loss Functions

As stated in the introduction, we assume that the loss function $f(w,z)$ is convex in w for each $z \in \mathcal{Z}$ and that W is a closed convex set. We use $\|\cdot\|$ to denote the Euclidean norm in $\mathbb{R}^n$. For convenience, we use the notation $F(w) = \mathbb{E}_z[f(w,z)]$ and assume $w^\star = \arg\min_{w\in W} F(w)$ always exists. Our main results require a couple of additional assumptions:

- *Smoothness* - we assume that f is L-smooth in its first argument, which means that for any $z \in \mathcal{Z}$, the function $f(\cdot, z)$ has L-Lipschitz continuous gradients. Formally,

$$\forall z \in \mathcal{Z}, \quad \forall w, w' \in W, \qquad \|\nabla_w f(w,z) - \nabla_w f(w',z)\| \le L\|w - w'\| \, . \tag{5}$$

- *Bounded Gradient Variance* - we assume that $\nabla_w f(w,z)$ has a σ^2-bounded variance for any fixed w, when z is sampled from the underlying distribution. In other words, we assume that there exists a constant $\sigma \ge 0$ such that

$$\forall w \in W, \qquad \mathbb{E}_z\left[\big\| \nabla_w f(w,z) - \nabla F(w) \big\|^2 \right] \le \sigma^2 \, .$$

Using these assumptions, regret bounds that explicitly depend on the gradient variance can be established (Juditsky et al., 2011; Lan, 2009; Xiao, 2010). In particular, for the projected stochastic gradient method defined in Equation (2), we have the following result:

Theorem 1 *Let $f(w,z)$ be an L-smooth convex loss function in w for each $z \in \mathcal{Z}$ and assume that the stochastic gradient $\nabla_w f(w,z)$ has σ^2-bounded variance for all $w \in W$. In addition, assume that W is convex and bounded, and let $D = \sqrt{\max_{u,v\in W} \|u - v\|^2/2}$. Then using $\alpha_j = L + (\sigma/D)\sqrt{j}$ in Equation (2) gives*

$$\mathbb{E}[R(m)] \;\le\; \big(F(w_1) - F(w^\star)\big) + D^2 L + 2D\sigma\sqrt{m}.$$

In the above theorem, the assumption that W is a bounded set does not play a critical role. Even if the learning problem has no constraints on w, we could always confine the search to a bounded set (say, a Euclidean ball of some radius) and Theorem 1 guarantees an $O(\sqrt{m})$ regret compared to the optimum within that set.

Similarly, for the dual averaging method defined in Equation (3), we have:

Theorem 2 *Let $f(w,z)$ be an L-smooth convex loss function in w for each $z \in \mathcal{Z}$, assume that the stochastic gradient $\nabla_w f(w,z)$ has σ^2-bounded variance for all $w \in W$, and let $D = \sqrt{h(w^\star) - \min_{w\in W} h(w)}$. Then, by setting $w_1 = \arg\min_{w\in W} h(w)$ and $\alpha_j = L + (\sigma/D)\sqrt{j}$ in the dual averaging method we have*

$$\mathbb{E}[R(m)] \;\leq\; \big(F(w_1) - F(w^\star)\big) + D^2 L + 2D\sigma\sqrt{m}.$$

For both of the above theorems, if $\nabla F(w^\star) = 0$ (which is certainly the case if $W = \mathbb{R}^n$), then the expected regret bounds can be simplified to

$$\mathbb{E}[R(m)] \;\leq\; 2D^2 L + 2D\sigma\sqrt{m}\;. \tag{6}$$

Proofs for these two theorems, as well as the above simplification, are given in Appendix A. Although we focus on expected regret bounds here, our results can equally be stated as high-probability bounds on the actual regret (see Appendix B for details).

In both Theorem 1 and Theorem 2, the parameters α_j are functions of σ. It may be difficult to obtain precise estimates of the gradient variance in many concrete applications. However, note that any upper bound on the variance suffices for the theoretical results to hold, and identifying such a bound is often easier than precisely estimating the actual variance. A loose bound on the variance will increase the constants in our regret bounds, but will not change its qualitative $O(\sqrt{m})$ rate.

Euclidean gradient descent and dual averaging are not the only update rules that can be plugged into Algorithm 1. The analysis in Appendix A (and Appendix B) actually applies to a much larger class of update rules, which includes the family of mirror descent updates (Nemirovski et al., 2009; Lan, 2009) and the family of (non-Euclidean) dual averaging updates (Nesterov, 2009; Xiao, 2010). For each of these update rules, we get an expected regret bound that closely resembles the bound in Equation (6).

Similar results can also be established for loss functions of the form $f(w,z) + \Psi(w)$, where $\Psi(w)$ is a simple convex regularization term that is not necessarily smooth. For example, setting $\Psi(w) = \lambda\|w\|_1$ with $\lambda > 0$ promotes sparsity in the predictor w. To extend the dual averaging method, we can use the following update rule in Xiao (2010):

$$w_{j+1} = \arg\min_{w\in W}\left\{\left\langle \frac{1}{j}\sum_{i=1}^{j} g_i,\, w\right\rangle + \Psi(w) + \frac{\alpha_j}{j}h(w)\right\}.$$

Similar extensions to the mirror descent method can be found in, for example, Duchi and Singer (2009). Using these composite forms of the algorithms, the same regret bounds as in Theorem 1 and Theorem 2 can be achieved even if $\Psi(w)$ is nonsmooth. The analysis is almost identical to Appendix A by using the general framework of Tseng (2008).

Asymptotically, the bounds we presented in this section are only controlled by the variance σ^2 and the number of iterations m. Therefore, we can think of any of the bounds mentioned above as an abstract function $\psi(\sigma^2, m)$, which we assume to be monotonically increasing in its arguments.

2.2 Analyzing the No-Communication Parallel Solution

Using the abstract notation $\psi(\sigma^2, m)$ for the expected regret bound simplifies our presentation significantly. As an example, we can easily give an analysis of the no-communication parallel solution described in the introduction.

Algorithm 2: Template for a serial mini-batch algorithm.

```
for $j = 1, 2, \dots$ do
    initialize $\bar{g}_j := 0$
    for $s = 1, \dots, b$ do
        define $i := (j-1)b + s$
        predict $w_j$
        receive input $z_i$ sampled i.i.d. from unknown distribution
        suffer loss $f(w_j, z_i)$
        $g_i := \nabla_w f(w_j, z_i)$
        $\bar{g}_j := \bar{g}_j + (1/b) g_i$
    end
    set $(w_{j+1}, a_{j+1}) = \phi(a_j, \bar{g}_j, \alpha_j)$
end
```

In the naïve no-communication solution, each of the k nodes in the parallel system applies the same serial update rule to its own substream of the high-rate inputs, and no communication takes place between them. If the total number of examples processed by the k nodes is m, then each node processes at most $\lceil m/k \rceil$ inputs. The examples received by each node are i.i.d. from the original distribution, with the same variance bound σ^2 for the stochastic gradients. Therefore, each node suffers an expected regret of at most $\psi(\sigma^2, \lceil m/k \rceil)$ on its portion of the input stream, and the total regret bound is obtain by simply summing over the k nodes, that is,

$$\mathbb{E}[R(m)] \; \le \; k\psi\left(\sigma^2, \left\lceil \frac{m}{k} \right\rceil\right).$$

If $\psi(\sigma^2, m) = 2D^2L + 2D\sigma\sqrt{m}$, as in Equation (6), then the expected total regret is

$$\mathbb{E}[R(m)] \; \le \; 2kD^2L + 2D\sigma k\sqrt{\left\lceil \frac{m}{k} \right\rceil}.$$

Comparing this bound to $2D^2L + 2D\sigma\sqrt{m}$ in the ideal serial solution, we see that it is approximately $\sqrt{k}$ times worse in its leading term. This is the price one pays for the lack of communication in the distributed system. In Section 4, we show how this $\sqrt{k}$ factor can be avoided by our DMB approach.

3. Serial Online Prediction using Mini-Batches

The expected regret bounds presented in the previous section depend on the variance of the stochastic gradients. The explicit dependency on the variance naturally suggests the idea of using averaged gradients over mini-batches to reduce the variance. Before we present the distributed mini-batch algorithm in the next section, we first analyze a *serial* mini-batch algorithm.

In the setting described in Algorithm 1, the update rule is applied after each input is received. We deviate from this setting and apply the update only periodically. Letting b be a user-defined *batch size* (a positive integer), and considering every b consecutive inputs as a *batch*. We define the *serial mini-batch algorithm* as follows: Our prediction remains constant for the duration of each batch, and is updated only when a batch ends. While processing the b inputs in batch j, the algorithm calculates and accumulates gradients and defines the average gradient

$$\bar{g}_j = \frac{1}{b}\sum_{s=1}^{b} \nabla_w f(w_j, z_{(j-1)b+s}) \ .$$

Hence, each batch of b inputs generates a single average gradient. Once a batch ends, the serial mini-batch algorithm feeds $\bar{g}_j$ to the update rule ϕ as the j^{th} gradient and obtains the new prediction for the next batch and the new state. See Algorithm 2 for a formal definition of the serial mini-batch algorithm. The appeal of the serial mini-batch setting is that the update rule is used less frequently, which may have computational benefits.

Theorem 3 *Let $f(w,z)$ be an L-smooth convex loss function in w for each $z \in \mathcal{Z}$ and assume that the stochastic gradient $\nabla_w f(w,z_i)$ has σ^2-bounded variance for all w. If the update rule ϕ has the serial regret bound $\psi(\sigma^2, m)$, then the expected regret of Algorithm 2 over m inputs is at most*

$$b\psi\left(\frac{\sigma^2}{b}, \left\lceil\frac{m}{b}\right\rceil\right) .$$

If $\psi(\sigma^2,m) = 2D^2L + 2D\sigma\sqrt{m}$, then the expected regret is bounded by

$$2bD^2L + 2D\sigma\sqrt{m+b}.$$

Proof Assume without loss of generality that b divides m, and that the serial mini-batch algorithm processes exactly m/b complete batches.[2] Let $\mathcal{Z}^b$ denote the set of all sequences of b elements from $\mathcal{Z}$, and assume that a sequence is sampled from $\mathcal{Z}^b$ by sampling each element i.i.d. from $\mathcal{Z}$. Let $\bar{f} : W \times \mathcal{Z}^b \mapsto \mathbb{R}$ be defined as

$$\bar{f}(w,(z_1,\ldots,z_b)) \ = \ \frac{1}{b}\sum_{s=1}^{b} f(w,z_s) \ .$$

In other words, $\bar{f}$ averages the loss function f across b inputs from $\mathcal{Z}$, while keeping the prediction constant. It is straightforward to show that $\mathbb{E}_{\bar{z}\in\mathcal{Z}^b}\bar{f}(w,\bar{z}) = \mathbb{E}_{z\in\mathcal{Z}} f(w,z) = F(w)$.

Using the linearity of the gradient operator, we have

$$\nabla_w \bar{f}(w,(z_1,\ldots,z_b)) = \frac{1}{b}\sum_{s=1}^{b} \nabla_w f(w,z_s) \ .$$

Let $\bar{z}_j$ denote the sequence $(z_{(j-1)b+1},\ldots,z_{jb})$, namely, the sequence of b inputs in batch j. The vector $\bar{g}_j$ in Algorithm 2 is precisely the gradient of $\bar{f}(\cdot,\bar{z}_j)$ evaluated at w_j. Therefore the serial mini-batch algorithm is equivalent to using the update rule ϕ with the loss function $\bar{f}$.

Next we check the properties of $\bar{f}(w,\bar{z})$ against the two assumptions in Section 2.1. First, if f is L-smooth then $\bar{f}$ is L-smooth as well due to the triangle inequality. Then we analyze the variance of the stochastic gradient. Using the properties of the Euclidean norm, we can write

$$\begin{aligned}\left\|\nabla_w \bar{f}(w,\bar{z}) - \nabla F(w)\right\|^2 &= \left\|\frac{1}{b}\sum_{s=1}^{b}(\nabla_w f(w,z_s) - \nabla F(w))\right\|^2 \\ &= \frac{1}{b^2}\sum_{s=1}^{b}\sum_{s'=1}^{b}\Big\langle \nabla_w f(w,z_s) - \nabla F(w), \nabla_w f(w,z_{s'}) - \nabla F(w)\Big\rangle.\end{aligned}$$

2. We can make this assumption since if b does not divide m then we can pad the input sequence with additional inputs until $m/b = \lceil m/b \rceil$, and the expected regret can only increase.

Notice that z_s and $z_{s'}$ are independent whenever $s \neq s'$, and in such cases,

$$\begin{aligned}&\mathbb{E}\Big\langle \nabla_w f(w,z_s) - \nabla F(w), \nabla_w f(w,z_{s'}) - \nabla F(w)\Big\rangle \\ =\ & \Big\langle \mathbb{E}\big[\nabla_w f(w,z_s) - \nabla F(w)\big],\, \mathbb{E}\big[\nabla_w f(w,z_{s'}) - \nabla F(w)\big]\Big\rangle \ = \ 0.\end{aligned}$$

Therefore, we have for every $w \in W$,

$$\mathbb{E}\left\|\nabla_w \bar{f}(w,\bar{z}) - \nabla F(w)\right\|^2 \ = \ \frac{1}{b^2}\sum_{s=1}^{b}\mathbb{E}\left\|\left(\nabla_w f(w,z_s) - \nabla F(w)\right)\right\|^2 \ \leq \ \frac{\sigma^2}{b}. \tag{7}$$

So we conclude that $\nabla_w \bar{f}(w,\bar{z}_j)$ has a (σ^2/b)-bounded variance for each j and each $w \in W$. If the update rule ϕ has a regret bound $\psi(\sigma^2,m)$ for the loss function f over m inputs, then its regret for $\bar{f}$ over m/b batches is bounded as

$$\mathbb{E}\left[\sum_{j=1}^{m/b}\left(\bar{f}(w_j,\bar{z}_j) - \bar{f}(w^\star,\bar{z}_j)\right)\right] \ \leq \ \psi\left(\frac{\sigma^2}{b},\frac{m}{b}\right).$$

By replacing $\bar{f}$ above with its definition, and multiplying both sides of the above inequality by b, we have

$$\mathbb{E}\left[\sum_{j=1}^{m/b}\sum_{i=(j-1)b+1}^{jb}\left(f(w_j,z_i) - f(w^\star,z_i)\right)\right] \ \leq \ b\psi\left(\frac{\sigma^2}{b},\frac{m}{b}\right).$$

If $\psi(\sigma^2,m) = 2D^2L + 2D\sigma\sqrt{m}$, then simply plugging in the general bound $b\psi(\sigma^2/b, \lceil m/b\rceil)$ and using $\lceil m/b\rceil \leq m/b + 1$ gives the desired result. However, we note that the optimal algorithmic parameters, as specified in Theorem 1 and Theorem 2, must be changed to $\alpha_j = L + (\sigma/\sqrt{b}D)\sqrt{j}$ to reflect the reduced variance σ^2/b in the mini-batch setting. ■

The bound in Theorem 3 is asymptotically equivalent to the $2D^2L + 2D\sigma\sqrt{m}$ regret bound for the basic serial algorithms presented in Section 2. In other words, performing the mini-batch update in the serial setting does not significantly hurt the performance of the update rule. On the other hand, it is also not surprising that using mini-batches in the serial setting does not improve the regret bound. After all, it is still a serial algorithm, and the bounds we presented in Section 2.1 are optimal. Nevertheless, our experiments demonstrate that in real-world scenarios, mini-batching can in fact have a very substantial positive effect on the transient performance of the online prediction algorithm, even in the serial setting (see Section 6 for details). Such positive effects are not captured by our asymptotic, worst-case analysis.

4. Distributed Mini-Batch for Stochastic Online Prediction

In this section, we show that in a distributed setting, the mini-batch idea can be exploited to obtain nearly optimal regret bounds. To make our setting as realistic as possible, we assume that any communication over the network incurs a latency. More specifically, we view the network as an undirected graph $\mathcal{G}$ over the set of nodes, where each edge represents a bi-directional network link. If nodes u and v are not connected by a link, then any communication between them must be relayed through other nodes. The latency incurred between u and v is therefore proportional to the graph distance between them, and the longest possible latency is thus proportional to the diameter of $\mathcal{G}$.

In addition to latency, we assume that the network has limited bandwidth. However, we would like to avoid the tedious discussion of data representation, compression schemes, error correcting, packet sizes, etc. Therefore, we do not explicitly quantify the bandwidth of the network. Instead, we require that the communication load at each node remains constant, and does not grow with the number of nodes k or with the rate at which the incoming functions arrive.

Although we are free to use any communication model that respects the constraints of our network, we assume only the availability of a distributed vector-sum operation. This is a standard[3] synchronized network operation. Each vector-sum operation begins with each node holding a vector v_j, and ends with each node holding the sum $\sum_{j=1}^{k} v_j$. This operation transmits messages along a rooted minimum-depth spanning-tree of $\mathcal{G}$, which we denote by $\mathcal{T}$: first the leaves of $\mathcal{T}$ send their vectors to their parents; each parent sums the vectors received from his children and adds his own vector; the parent then sends the result to his own parent, and so forth; ultimately the sum of all vectors reaches the tree root; finally, the root broadcasts the overall sum down the tree to all of the nodes.

An elegant property of the vector-sum operation is that it uses each up-link and each down-link in $\mathcal{T}$ exactly once. This allows us to start vector-sum operations back-to-back. These vector-sum operations will run concurrently without creating network congestion on any edge of $\mathcal{T}$. Furthermore, we assume that the network operations are *non-blocking*, meaning that each node can continue processing incoming inputs while the vector-sum operation takes place in the background. This is a key property that allows us to efficiently deal with network latency. To formalize how latency affects the performance of our algorithm, let μ denote the number of inputs that are processed by the entire system during the period of time it takes to complete a vector-sum operation across the entire network. Usually μ scales linearly with the diameter of the network, or (for appropriate network architectures) logarithmically in the number of nodes k.

4.1 The DMB Algorithm

We are now ready to present a general technique for applying a deterministic update rule ϕ in a distributed environment. This technique resembles the serial mini-batch technique described earlier, and is therefore called the *distributed mini-batch* algorithm, or DMB for short.

Algorithm 3 describes a template of the DMB algorithm that runs in parallel on each node in the network, and Figure 1 illustrates the overall algorithm work-flow. Again, let b be a batch size, which we will specify later on, and for simplicity assume that k divides b and μ. The DMB algorithm processes the input stream in batches $j = 1,2,\ldots$, where each batch contains $b+\mu$ consecutive inputs. During each batch j, all of the nodes use a common predictor w_j. While observing the first b inputs in a batch, the nodes calculate and accumulate the stochastic gradients of the loss function f at w_j. Once the nodes have accumulated b gradients altogether, they start a distributed vector-sum operation to calculate the sum of these b gradients. While the vector-sum operation completes in the background, μ additional inputs arrive (roughly μ/k per node) and the system keeps processing them using the same predictor w_j. The gradients of these additional μ inputs are discarded (to this end, they do not need to be computed). Although this may seem wasteful, we show that this waste can be made negligible by choosing b appropriately.

3. For example, all-reduce with the sum operation is a standard operation in MPI.

Algorithm 3: Distributed mini-batch (DMB) algorithm (running on each node).

```
for $j = 1, 2, \ldots$ do
    initialize $\hat{g}_j := 0$
    for $s = 1, \ldots, b/k$ do
        predict $w_j$
        receive input $z$ sampled i.i.d. from unknown distribution
        suffer loss $f(w_j, z)$
        compute $g := \nabla_w f(w_j, z)$
        $\hat{g}_j := \hat{g}_j + g$
    end
    call the distributed vector-sum to compute the sum of $\hat{g}_j$ across all nodes
    receive $\mu/k$ additional inputs and continue predicting using $w_j$
    finish vector-sum and compute average gradient $\bar{g}_j$ by dividing the sum by $b$
    set $(w_{j+1}, a_{j+1}) = \phi(a_j, \bar{g}_j, \alpha_j)$
end
```

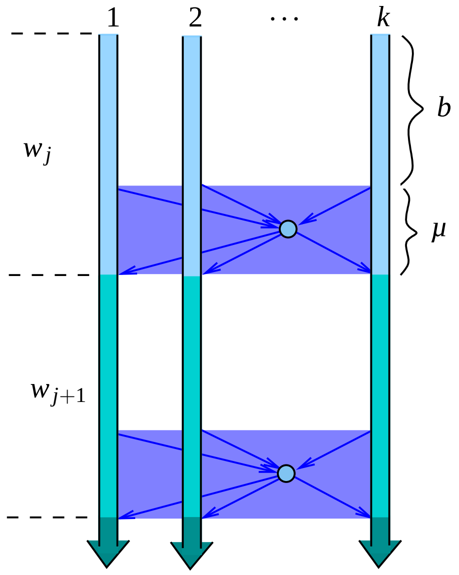

Figure 1: Work flow of the DMB algorithm. Within each batch $j = 1, 2, \ldots$, each node accumulates the stochastic gradients of the first b/k inputs. Then a vector-sum operation across the network is used to compute the average across all nodes. While the vector-sum operation completes in the background, a total of μ inputs are processed by the processors using the same predictor w_j, but their gradients are not collected. Once all of the nodes have the overall average $\bar{g}_j$, each node updates the predictor using the same deterministic serial algorithm.

Once the vector-sum operation completes, each node holds the sum of the b gradients collected during batch j. Each node divides this sum by b and obtains the average gradient, which we denote by $\bar{g}_j$. Each node feeds this average gradient to the update rule ϕ, which returns a new synchronized prediction w_{j+1}. In summary, during batch j each node processes $(b+\mu)/k$ inputs using the same predictor w_j, but only the first b/k gradients are used to compute the next predictor. Nevertheless, all $b+\mu$ inputs are counted in our regret calculation.

If the network operations are conducted over a spanning tree, then an obvious variants of the DMB algorithm is to let the root apply the update rule to get the next predictor, and then broadcast it to all other nodes. This saves repeated executions of the update rule at each node (but requires interruption or modification of the standard vector-sum operations in the network communication model). Moreover, this guarantees all the nodes having the same predictor even with update rules that depends on some random bits.

Theorem 4 *Let $f(w,z)$ be an L-smooth convex loss function in w for each $z \in \mathcal{Z}$ and assume that the stochastic gradient $\nabla_w f(w,z_i)$ has σ^2-bounded variance for all $w \in W$. If the update rule ϕ has the serial regret bound $\psi(\sigma^2,m)$, then the expected regret of Algorithm 3 over m samples is at most*

$$(b+\mu)\,\psi\left(\frac{\sigma^2}{b}, \left\lceil \frac{m}{b+\mu} \right\rceil\right) .$$

Specifically, if $\psi(\sigma^2,m) = 2D^2L + 2D\sigma\sqrt{m}$, then setting the batch size $b = m^{1/3}$ gives the expected regret bound

$$2D\sigma\sqrt{m} + 2Dm^{1/3}\,(LD + \sigma\sqrt{\mu}) + 2D\sigma m^{1/6} + 2D\sigma\mu m^{-1/6} + 2\mu D^2 L. \qquad (8)$$

In fact, if $b = m^\rho$ for any $\rho \in (0,1/2)$, the expected regret bound is $2D\sigma\sqrt{m} + o(\sqrt{m})$.

To appreciate the power of this result, we compare the specific bound in Equation (8) with the ideal serial solution and the naïve no-communication solution discussed in the introduction. It is clear that our bound is asymptotically equivalent to the ideal serial bound $\psi(\sigma^2,m)$—even the constants in the dominant terms are identical. Our bound scales nicely with the network latency and the cluster size k, because μ (which usually scales logarithmically with k) does not appear in the dominant $\sqrt{m}$ term. On the other hand, the naïve no-communication solution has regret bounded by $k\psi\left(\sigma^2, m/k\right) = 2kD^2L + 2D\sigma\sqrt{km}$ (see Section 2.2). If $1 \ll k \ll m$, this bound is worse than the bound in Theorem 4 by a factor of $\sqrt{k}$.

Finally, we note that choosing b as m^ρ for an appropriate ρ requires knowledge of m in advance. However, this requirement can be relaxed by applying a standard doubling trick (Cesa-Bianchi and Lugosi, 2006). This gives a single algorithm that does not take m as input, with asymptotically similar regret. If we use a fixed b regardless of m, the dominant term of the regret bound becomes $2D\sigma\sqrt{\log(k)m/b}$; see the following proof for details.

Proof Similar to the proof of Theorem 3, we assume without loss of generality that k divides $b+\mu$, we define the function $\bar{f} : W \times \mathcal{Z}^b \mapsto \mathbb{R}$ as

$$\bar{f}(w,(z_1,\ldots,z_b)) \;=\; \frac{1}{b}\sum_{s=1}^{b} f(w,z_s) \;\;,$$

and we use $\bar{z}_j$ to denote the *first b inputs* in batch j. By construction, the function $\bar{f}$ is L-smooth and its gradients have σ^2/b-bounded variance. The average gradient $\bar{g}_j$ computed by the DMB algorithm is the gradient of $\bar{f}(\cdot,\bar{z}_j)$ evaluated at the point w_j. Therefore,

$$\mathbb{E}\left[\sum_{j=1}^{m/(b+\mu)} \left(\bar{f}(w_j,\bar{z}_j) - \bar{f}(w^\star,\bar{z}_j)\right)\right] \;\le\; \psi\left(\frac{\sigma^2}{b}, \frac{m}{b+\mu}\right). \tag{9}$$

This inequality only involve the additional μ examples in counting the number of batches as $m/b+\mu$. In order to count them in the total regret, we notice that

$$\forall\, j, \quad \mathbb{E}\left[\bar{f}(w_j,\bar{z}_j)\,|\,w_j\right] = \mathbb{E}\left[\frac{1}{b+\mu}\sum_{i=(j-1)(b+\mu)+1}^{j(b+\mu)} f(w_j,z_i)\;\middle|\; w_j\right],$$

and a similar equality holds for $\bar{f}(w^\star,z_i)$. Substituting these equalities in the left-hand-side of Equation (9) and multiplying both sides by $b+\mu$ yields

$$\mathbb{E}\left[\sum_{j=1}^{m/(b+\mu)} \sum_{i=(j-1)(b+\mu)+1}^{j(b+\mu)} \left(f(w_j,z_i) - f(w^\star,z_i)\right)\right] \;\le\; (b+\mu)\,\psi\left(\frac{\sigma^2}{b}, \frac{m}{b+\mu}\right).$$

Again, if $(b+\mu)$ divides m, then the left-hand side above is exactly the expected regret of the DMB algorithm over m examples. Otherwise, the expected regret can only be smaller.

For the concrete case of $\psi(\sigma^2,m) = 2D^2L + 2D\sigma\sqrt{m}$, plugging in the new values for σ^2 and m results in a bound of the form

$$\begin{aligned}(b+\mu)\psi\left(\frac{\sigma^2}{b}, \left\lceil\frac{m}{b+\mu}\right\rceil\right) \;&\le\; (b+\mu)\psi\left(\frac{\sigma^2}{b}, \frac{m}{b+\mu}+1\right)\\ &\le\; 2(b+\mu)D^2L + 2D\sigma\sqrt{m + \frac{\mu}{b}m + \frac{(b+\mu)^2}{b}}.\end{aligned}$$

Using the inequality $\sqrt{x+y+z} \le \sqrt{x}+\sqrt{y}+\sqrt{z}$, which holds for any nonnegative numbers x, y and z, we bound the expression above by

$$2(b+\mu)D^2L + 2D\sigma\sqrt{m} + 2D\sigma\sqrt{\frac{\mu m}{b}} + 2D\sigma\frac{b+\mu}{\sqrt{b}}.$$

It is clear that with $b = Cm^\rho$ for any $\rho \in (0,1/2)$ and any constant $C > 0$, this bound can be written as $2D\sigma\sqrt{m} + o(\sqrt{m})$. Letting $b = m^{1/3}$ gives the smallest exponents in the $o(\sqrt{m})$ terms. ■

In the proofs of Theorem 3 and Theorem 4, decreasing the variance by a factor of b, as given in Equation (7), relies on properties of the Euclidean norm. For serial gradient-type algorithms that are specified with different norms (see the general framework in Appendix A), the variance does not typically decrease as much. For example, in the dual averaging method specified in Equation (3), if we use $h(w) = 1/(2(p-1))\|w\|_p^2$ for some $p \in (1,2]$, then the "variance" bounds for the stochastic gradients must be expressed in the dual norm, that is, $\mathbb{E}\,\|\nabla_w f(w,z) - \nabla F(w)\|_q^2 \le \sigma^2$, where $q = p/(p-1) \in [2,\infty)$. In this case, the variance bound for the averaged function becomes

$$\mathbb{E}\left\|\nabla_w \bar{f}(w,\bar{z}) - \nabla F(w)\right\|_q^2 \;\le\; C(n,q)\frac{\sigma^2}{b},$$

where $C(n,q) = \min\{q-1, O(\log(n))\}$ is a space-dependent constant.[4] Nevertheless, we can still obtain a linear reduction in b even for such non-Euclidean norms. The net effect is that the regret bound for the DMB algorithm becomes $2D\sqrt{C(n,q)}\sigma\sqrt{m} + o(\sqrt{m})$.

4.2 Improving Performance on Short Input Streams

Theorem 4 presents an optimal way of choosing the batch size b, which results in an asymptotically optimal regret bound. However, our asymptotic approach hides a potential shortcoming that occurs when m is small. Say that we know, ahead of time, that the sequence length is $m = 15,000$. Moreover, say that the latency is $\mu = 100$, and that $\sigma = 1$ and $L = 1$. In this case, Theorem 4 determines that the optimal batch size is $b \sim 25$. In other words, for every 25 inputs that participate in the update, 100 inputs are discarded. This waste becomes negligible as b grows with m and does not affect our asymptotic analysis. However, if m is known to be small, we can take steps to improve the situation.

Assume for simplicity that b divides μ. Now, instead of running a single distributed mini-batch algorithm, we run $c = 1 + \mu/b$ independent interlaced instances of the distributed mini-batch algorithm on each node. At any given moment, $c-1$ instances are asleep and one instance is active. Once the active instance collects b/k gradients on each node, it starts a vector-sum network operation, awakens the next instance, and puts itself to sleep. Note that each instance awakens after $(c-1)b = \mu$ inputs, which is just in time for its vector-sum operation to complete.

In the setting described above, c different vector-sum operations propagate concurrently through the network. The distributed vector sum operation is typically designed such that each network link is used at most once in each direction, so concurrent sum operations that begin at different times should not compete for network resources. The batch size should indeed be set such that the generated traffic does not exceed the network bandwidth limit, but the latency of each sum operation should not be affected by the fact that multiple sum operations take place at once.

Simply interlacing c independent copies of our algorithm does not resolve the aforementioned problem, since each prediction is still defined by $1/c$ of the observed inputs. Therefore, instead of using the predictions prescribed by the individual online predictors, we use their average. Namely, we take the most recent prediction generated by each instance, average these predictions, and use this average in place of the original prediction.

The advantages of this modification are not apparent from our theoretical analysis. Each instance of the algorithm handles m/c inputs and suffers a regret of at most

$$b\psi\left(\frac{\sigma^2}{b}, 1 + \frac{m}{bc}\right) \quad ,$$

and, using Jensen's inequality, the overall regret using the average prediction is upper bounded by

$$bc\psi\left(\frac{\sigma^2}{b}, 1 + \frac{m}{bc}\right) \quad .$$

The bound above is precisely the same as the bound in Theorem 4. Despite this fact, we conjecture that this method will indeed improve empirical results when the batch size b is small compared to the latency term μ.

4. For further details of algorithms using p-norm, see Xiao (2010, Section 7.2) and Shalev-Shwartz and Tewari (2011). For the derivation of $C(n,q)$ see for instance Lemma B.2 in Cotter et al. (2011).

5. Stochastic Optimization

As we discussed in the introduction, the *stochastic optimization* problem is closely related, but not identical, to the stochastic online prediction problem. In both cases, there is a loss function $f(w,z)$ to be minimized. The difference is in the way success is measured. In online prediction, success is measured by regret, which is the difference between the cumulative loss suffered by the prediction algorithm and the cumulative loss of the best fixed predictor. The goal of stochastic optimization is to find an approximate solution to the problem

$$\underset{w\in W}{\text{minimize}} \quad F(w) \triangleq \mathbb{E}_z[f(w,z)] \ ,$$

and success is measured by the difference between the expected loss of the final output of the optimization algorithm and the expected loss of the true minimizer $w^\star$. As before, we assume that the loss function $f(w,z)$ is convex in w for any $z \in \mathcal{Z}$, and that W is a closed convex set.

We consider the same *stochastic approximation* type of algorithms presented in Algorithm 1, and define the final output of the algorithm, after processing m i.i.d. samples, to be

$$\bar{w}_m = \frac{1}{m}\sum_{j=1}^{m} w_j \ .$$

In this case, the appropriate measure of success is the optimality gap

$$G(m) \ = \ F(\bar{w}_m) - F(w^\star) \ .$$

Notice that the optimality gap $G(m)$ is also a random variable, because $\bar{w}_m$ depends on the random samples $z_1,\ldots,z_m$. It can be shown (see, e.g., Xiao, 2010, Theorem 3) that for convex loss functions and i.i.d. inputs, we always have

$$\mathbb{E}[G(m)] \ \le \ \frac{1}{m}\mathbb{E}[R(m)] \ .$$

Therefore, a bound on the expected optimality gap can be readily obtained from a bound on the expected regret of the same algorithm. In particular, if f is an L-smooth convex loss function and $\nabla_w f(w,z)$ has σ^2-bounded variance, and our algorithm has a regret bound of $\psi(\sigma^2,m)$, then it also has an expected optimality gap of at most

$$\bar{\psi}(\sigma^2,m) = \frac{1}{m}\psi(\sigma^2,m) \ .$$

For the specific regret bound $\psi(\sigma^2,m) = 2D^2L + 2D\sigma\sqrt{m}$, which holds for the serial algorithms presented in Section 2, we have

$$\mathbb{E}[G(m)] \ \le \ \bar{\psi}(\sigma^2,m) \ = \ \frac{2D^2L}{m} + \frac{2D\sigma}{\sqrt{m}} \ .$$

5.1 Stochastic Optimization using Distributed Mini-Batches

Our template of a DMB algorithm for stochastic optimization (see Algorithm 4) is very similar to the one presented for the online prediction setting. The main difference is that we do not have to process inputs while waiting for the vector-sum network operation to complete. Again let b be the batch size, and the number of batches $r = \lfloor m/b \rfloor$. For simplicity of discussion, we assume that b divides m.

Algorithm 4: Template of DMB algorithm for stochastic optimization.

```
$r \leftarrow \lfloor \frac{m}{b} \rfloor$
**for** $j = 1, 2, \dots, r$ **do**
    reset $\hat{g}_j = 0$
    **for** $s = 1, \dots, b/k$ **do**
        receive input $z_s$ sampled i.i.d. from unknown distribution
        calculate $g_s = \nabla_w f(w_j, z_s)$
        calculate $\hat{g}_j \leftarrow \hat{g}_j + g_i$
    **end**
    start distributed vector sum to compute the sum of $\hat{g}_j$ across all nodes
    finish distributed vector sum and compute average gradient $\bar{g}_j$
    set $(w_{j+1}, a_{j+1}) = \phi(a_j, \bar{g}_j, j)$
**end**
**Output**: $\frac{1}{r} \sum_{j=1}^{r} w_j$
```

Theorem 5 *Let $f(w,z)$ be an L-smooth convex loss function in w for each $z \in \mathcal{Z}$ and assume that the stochastic gradient $\nabla_w f(w,z)$ has σ^2-bounded variance for all $w \in W$. If the update rule ϕ used in a serial setting has an expected optimality gap bounded by $\bar{\psi}(\sigma^2, m)$, then the expected optimality gap of Algorithm 4 after processing m samples is at most*

$$\bar{\psi}\left(\frac{\sigma^2}{b}, \frac{m}{b}\right) .$$

If $\bar{\psi}(\sigma^2, m) = \frac{2D^2L}{m} + \frac{2D\sigma}{\sqrt{m}}$, then the expected optimality gap is bounded by

$$\frac{2bD^2L}{m} + \frac{2D\sigma}{\sqrt{m}} .$$

The proof of the theorem follows along the lines of Theorem 3, and is omitted.

We comment that the accelerated stochastic gradient methods of Lan (2009), Hu et al. (2009) and Xiao (2010) can also fit in our template for the DMB algorithm, but with more sophisticated updating rules. These accelerated methods have an expected optimality bound of $\bar{\psi}(\sigma^2, m) = {}^{4D^2L}/_{m^2} + {}^{4D\sigma}/_{\sqrt{m}}$, which translates into the following bound for the DMB algorithm:

$$\bar{\psi}\left(\frac{\sigma^2}{b}, \frac{m}{b}\right) = \frac{4b^2D^2L}{m^2} + \frac{4D\sigma}{\sqrt{m}} .$$

Most recently, Ghadimi and Lan (2010) developed accelerated stochastic gradient methods for strongly convex functions that have the convergence rate $\bar{\psi}(\sigma^2, m) = O(1)\left({}^{L}/_{m^2} + {}^{\sigma^2}/_{\nu m}\right)$, where ν is the strong convexity parameter of the loss function. The corresponding DMB algorithm has a convergence rate

$$\bar{\psi}\left(\frac{\sigma^2}{b}, \frac{m}{b}\right) = O(1)\left(\frac{b^2L}{m^2} + \frac{\sigma^2}{\nu m}\right) .$$

Apparently, this also fits in the DMB algorithm nicely.

The significance of our result is that the dominating factor in the convergence rate is not affected by the batch size. Therefore, depending on the value of m, we can use large batch sizes without affecting the convergence rate in a significant way. Since we can run the workload associated with a single batch in parallel, this theorem shows that the mini-batch technique is capable of turning many serial optimization algorithms into parallel ones. To this end, it is important to analyze the speed-up of the parallel algorithms in terms of the running time (wall-clock time).

5.2 Parallel Speed-Up

Recall that k is the number of parallel computing nodes and m is the total number of i.i.d. samples to be processed. Let $b(m)$ be the batch size that depends on m. We define a *time-unit* to be the time it takes a single node to process one sample (including computing the gradient and updating the predictor). For convenience, let δ be the latency of the vector-sum operation in the network (measured in number of time-units).[5] Then the parallel speed-up of the DMB algorithm is

$$S(m) = \frac{m}{\frac{m}{b(m)}\left(\frac{b(m)}{k} + \delta\right)} = \frac{k}{1 + \frac{\delta}{b(m)}k} \ ,$$

where $m/b(m)$ is the number of batches, and $b(m)/k + \delta$ is the wall-clock time by k processors to finish one batch in the DMB algorithm. If $b(m)$ increases at a fast enough rate, then we have $S(m) \to k$ as $m \to \infty$. Therefore, we obtain an asymptotically linear speed-up, which is the ideal result that one would hope for in parallelizing the optimization process (see Gustafson, 1988).

In the context of stochastic optimization, it is more appropriate to measure the speed-up with respect to the same optimality gap, not the same amount of samples processed. Let ε be a given target for the expected optimality gap. Let $m_{\mathrm{srl}}(\varepsilon)$ be the number of samples that the serial algorithm needs to reach this target and let $m_{\mathrm{DMB}}(\varepsilon)$ be the number of samples needed by the DMB algorithm. Slightly overloading our notation, we define the parallel speed-up with respect to the expected optimality gap ε as

$$S(\varepsilon) = \frac{m_{\mathrm{srl}}(\varepsilon)}{\frac{m_{\mathrm{DMB}}(\varepsilon)}{b}\left(\frac{b}{k} + \delta\right)} \ . \tag{10}$$

In the above definition, we intentionally leave the dependence of b on m unspecified. Indeed, once we fix the function $b(m)$, we can substitute it into the equation $\bar{\psi}(\sigma^2/b, m/b) = \varepsilon$ to solve for the exact form of $m_{\mathrm{DMB}}(\varepsilon)$. As a result, b is also a function of ε.

Since both $m_{\mathrm{srl}}(\varepsilon)$ and $m_{\mathrm{DMB}}(\varepsilon)$ are upper bounds for the actual running times to reach ε-optimality, their ratio $S(\varepsilon)$ may not be a precise measure of the speed-up. However, it is difficult in practice to measure the actual running times of the algorithms in terms of reaching ε-optimality. So we only hope $S(\varepsilon)$ gives a conceptual guide in comparing the actual performance of the algorithms. The following result shows that if the batch size b is chosen to be of order m^{ρ} for any $\rho \in (0, 1/2)$, then we still have asymptotic linear speed-up.

Theorem 6 *Let $f(w,z)$ be an L-smooth convex loss function in w for each $z \in \mathcal{Z}$ and assume that the stochastic gradient $\nabla_w f(w,z)$ has σ^2-bounded variance for all $w \in W$. Suppose the update rule ϕ used in the serial setting has an expected optimality gap bounded by $\bar{\psi}(\sigma^2, m) = \frac{2D^2L}{m} + \frac{2D\sigma}{\sqrt{m}}$. If the*

5. The relationship between δ and μ defined in the online setting (see Section 4) is roughly $\mu = k\delta$.

batch size in the DMB algorithm is chosen as $b(m) = \Theta(m^\rho)$*, where* $\rho \in (0, 1/2)$*, then we have*

$$\lim_{\varepsilon \to 0} S(\varepsilon) = k.$$

Proof By solving the equation

$$\frac{2D^2L}{m} + \frac{2D\sigma}{\sqrt{m}} = \varepsilon\ ,$$

we see that the following number of samples is sufficient for the serial algorithm to reach ε-optimality:

$$m_{\text{srl}}(\varepsilon) \;=\; \frac{D^2\sigma^2}{\varepsilon^2}\left(1+\sqrt{1+\frac{2L\varepsilon}{\sigma^2}}\right)^2 .$$

For the DMB algorithm, we use the batch size $b(m) = (\theta\sigma/DL)m^\rho$, with some $\theta > 0$, to obtain the equation

$$\frac{2b(m)D^2L}{m} + \frac{2D\sigma}{\sqrt{m}} \;=\; \frac{2D\sigma}{m^{1/2}}\left(1+\frac{\theta}{m^{1/2-\rho}}\right) \;=\; \varepsilon. \tag{11}$$

We use $m_{\text{DMB}}(\varepsilon)$ to denote the solution of the above equation. Apparently $m_{\text{DMB}}(\varepsilon)$ is a monotone function of ε and $\lim_{\varepsilon\to 0} m_{\text{DMB}}(\varepsilon) = \infty$. For convenience (with some abuse of notation), let $b(\varepsilon)$ to denote $b(m_{\text{DMB}}(\varepsilon))$, which is also monotone in ε and satisfies $\lim_{\varepsilon\to 0} b(\varepsilon) = \infty$. Moreover, for any batch size $b > 1$, we have $m_{\text{DMB}}(\varepsilon) \geq m_{\text{srl}}(\varepsilon)$. Therefore, from Equation (10) we get

$$\limsup_{\varepsilon\to 0} S(\varepsilon) \leq \lim_{\varepsilon\to 0} \frac{k}{1+\frac{\delta}{b(\varepsilon)}k} = k.$$

Next we show $\liminf_{\varepsilon\to 0} S(\varepsilon) \geq k$. For any $\eta > 0$, let

$$m_\eta(\varepsilon) = \frac{4D^2\sigma^2(1+\eta)^2}{\varepsilon^2}.$$

which is monotone decreasing in ε, and can be seen as the solution to the equation

$$\frac{2D\sigma}{m^{1/2}}(1+\eta) \;=\; \varepsilon.$$

Comparing this equation with Equation (11), we see that, for any $\eta > 0$, there exists an ε' such that for all $0 < \varepsilon \leq \varepsilon'$, we have $m_{\text{DMB}}(\varepsilon) \leq m_\eta(\varepsilon)$. Therefore,

$$\liminf_{\varepsilon\to 0} S(\varepsilon) \;\geq\; \lim_{\varepsilon\to 0} \frac{m_{\text{srl}}(\varepsilon)}{m_\eta(\varepsilon)} \frac{k}{1+\frac{\delta}{b(\varepsilon)}k} \;=\; \lim_{\varepsilon\to 0} \frac{\left(1+\sqrt{1+\frac{2L\varepsilon}{\sigma^2}}\right)^2}{4(1+\eta)^2} \frac{k}{1+\frac{\delta}{b(\varepsilon)}k} \;=\; \frac{1}{(1+\eta)^2}k.$$

Since the above inequality holds for any $\eta > 0$, we can take $\eta \to 0$ and conclude that $\liminf_{\varepsilon\to 0} S(\varepsilon) \geq k$. This finishes the proof. ■

For accelerated stochastic gradient methods whose convergence rates have a similar dependence on the gradient variance (Lan, 2009; Hu et al., 2009; Xiao, 2010; Ghadimi and Lan, 2010), the batch size b has a even smaller effect on the convergence rate (see discussions after Theorem 5), which implies a better parallel speed-up.

6. Experiments

We conducted experiments with a large-scale online binary classification problem. First, we obtained a log of one billion queries issued to the Internet search engine Bing. Each entry in the log specifies a time stamp, a query text, and the id of the user who issued the query (using a temporary browser cookie). A query is said to be *highly monetizable* if, in the past, users who issued this query tended to then click on online advertisements. Given a predefined list of one million highly monetizable queries, we observe the queries in the log one-by-one and attempt to predict whether the next query will be highly monetizable or not. A clever search engine could use this prediction to optimize the way it presents search results to the user. A prediction algorithm for this task must keep up with the stream of queries received by the search engine, which calls for a distributed solution.

The predictions are made based on the recent query-history of the current user. For example, the predictor may learn that users who recently issued the queries “island weather” and “sunscreen reviews” (both not highly monetizable in our data) are likely to issue a subsequent query which is highly monetizable (say, a query like “Hawaii vacation”). In the next section, we formally define how each input, z_t, is constructed.

First, let n denote the number of distinct queries that appear in the log and assume that we have enumerated these queries, $q_1,\ldots,q_n$. Now define $x_t \in \{0,1\}^n$ as follows

$$x_{t,j} = \begin{cases} 1 & \text{if query } q_j \text{ was issued by the current user during the last two hours,} \\ 0 & \text{otherwise.} \end{cases}$$

Let y_t be a binary variable, defined as

$$y_t = \begin{cases} +1 & \text{if the current query is highly monetizable,} \\ -1 & \text{otherwise.} \end{cases}$$

In other words, y_t is the binary label that we are trying to predict. Before observing x_t or y_t, our algorithm chooses a vector $w_t \in \mathbb{R}^n$. Then x_t is observed and the resulting binary prediction is the sign of their inner product $\langle w_t, x_t\rangle$. Next, the correct label y_t is revealed and our binary prediction is incorrect if $y_t\langle w_t, x_t\rangle \le 0$. We can re-state this prediction problem in an equivalent way by defining $z_t = y_t x_t$, and saying that an incorrect prediction occurs when $\langle w_t, z_t\rangle \le 0$.

We adopt the logistic loss function as a smooth convex proxy to the error indicator function. Formally, define f as

$$f(w,z) \;=\; \log_2\big(1+\exp(-\langle w,z\rangle)\big) \;.$$

Additionally, we introduced the convex regularization constraint $\|w_t\| \le C$, where C is a predefined regularization parameter.

We ran the synchronous version of our distributed algorithm using the Euclidean dual averaging update rule (4) in a cluster simulation. The simulation allowed us to easily investigate the effects of modifying the number of nodes in the cluster and the latencies in the network.

We wanted to specify a realistic latency in our simulation, which faithfully mimics the behavior of a real network in a search engine datacenter. To this end, we assumed that the nodes are connected via a standard 1Gbs Ethernet network. Moreover, we assumed that the nodes are arranged in a precomputed logical binary-tree communication structure, and that all communication is done along the edges in this tree. We conservatively estimated the round-trip latency between proximal nodes in the tree to be 0.5ms. Therefore, the total time to complete each vector-sum network operation

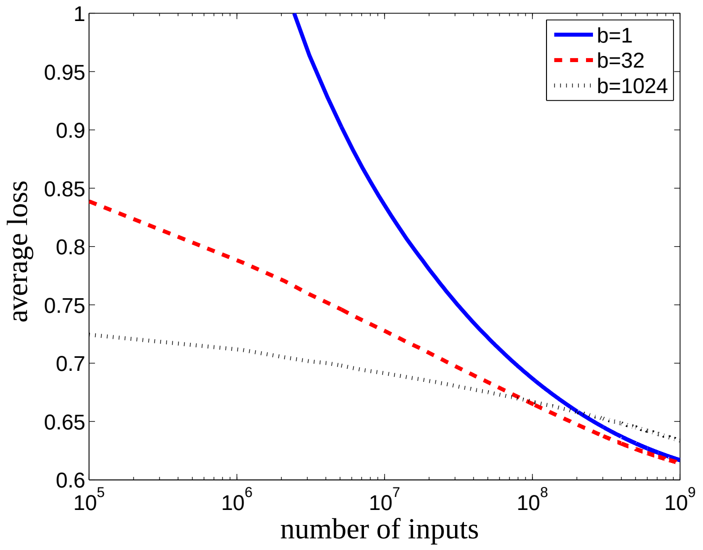

Figure 2: The effects of of the batch size when serial mini-batching on average loss. The mini-batches algorithm was applied with different batch sizes. The x-axis presents the number of instances observed, and the y-axis presents the average loss. Note that the case $b = 1$ is the standard serial dual-averaging algorithm.

is $\log_2(k)$ ms, where k is the number of nodes in the cluster. We assumed that our search engine receives 4 queries per ms (which adds up to ten billion queries a month). Overall, the number of queries discarded between mini-batches is $\mu = 4\log_2(k)$.

In all of our experiments, we use the algorithmic parameter $\alpha_j = L + \gamma\sqrt{j}$ (see Theorem 2). We set the smoothness parameter L to a constant, and the parameter γ to a constant divided by $\sqrt{b}$. This is because L depends only on the loss function f, which does not change in DMB, while γ is proportional to σ, the standard deviation of the gradient-averages. We chose the constants by manually exploring the parameter space on a separate held-out set of 500 million queries.

We report all of our results in terms of the average loss suffered by the online algorithm. This is simply defined as $(1/t)\sum_{i=1}^{t} f(w_i, z_i)$. We cannot plot regret, as we do not know the offline risk minimizer $w^\star$.

6.1 Serial Mini-Batching

As a warm-up, we investigated the effects of modifying the mini-batch size b in a standard serial Euclidean dual averaging algorithm. This is equivalent to running the distributed simulation with a cluster size of $k = 1$, with varying mini-batch size. We ran the experiment with $b = 1, 2, 4, \ldots, 1024$. Figure 2 shows the results for three representative mini-batch sizes. The experiments tell an interesting story, which is more refined than our theoretical upper bounds. While the asymptotic worst-case theory implies that batch-size should have no significant effect, we actually observe that mini-batching accelerates the learning process on the first 10^8 inputs. On the other hand, after 10^8 inputs, a large mini-batch size begins to hurt us and the smaller mini-batch sizes gain the lead. This behavior is not an artifact of our choice of the parameters γ and L, as we observed a similar behavior

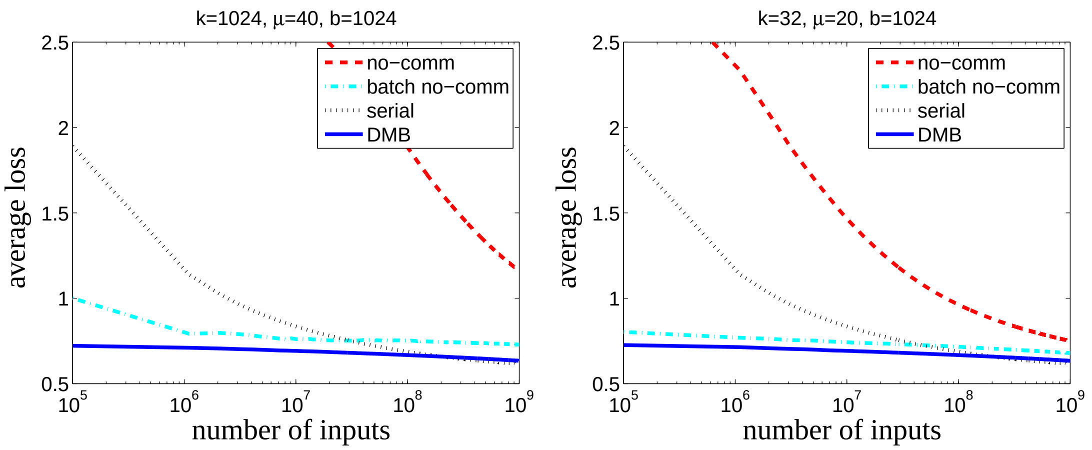

Figure 3: Comparing DBM with the serial algorithm and the no-communication distributed algorithm. Results for a large cluster of $k = 1024$ machines are presented on the left. Results for a small cluster of $k = 32$ machines are presented on the right.

for many different parameter setting, during the initial stage when we tuned the parameters on a held-out set.

Similar transient behaviors also exist for multi-step stochastic gradient methods (see, e.g., Polyak, 1987, Section 4.3.2), where the multi-step interpolation of the gradients also gives the smoothing effects as using averaged gradients. Typically such methods converge faster in the early iterations when the iterates are far from the optimal solution and the relative value of the stochastic noise is small, but become less effective asymptotically.

6.2 Evaluating DBM

Next, we compared the average loss of the DBM algorithm with the average loss of the serial algorithm and the no-communication algorithm (where each cluster node works independently). We tried two versions of the no-communication solution. The first version simply runs k independent copies of the serial prediction algorithm. The second version runs k independent copies of the serial mini-batch algorithm, with a mini-batch size of 128. We included the second version of the no-communication algorithm after observing that mini-batching has significant advantages even in the serial setting. We experimented with various cluster sizes and various mini-batch sizes. As mentioned above, we set the latency of the DBM algorithm to $\mu = 4\log_2(k)$. Taking a cue from our theoretical analysis, we set the batch size to $b = m^{1/3} \simeq 1024$. We repeated the experiment for various cluster sizes and the results were very consistent. Figure 3 presents the average loss of the three algorithms for clusters of sizes $k = 1024$ and $k = 32$. Clearly, the simple no-communication algorithm performs very poorly compared to the others. The no-communication algorithm that uses mini-batch updates on each node does surprisingly well, but is still outperformed quite significantly by the DMB solution.

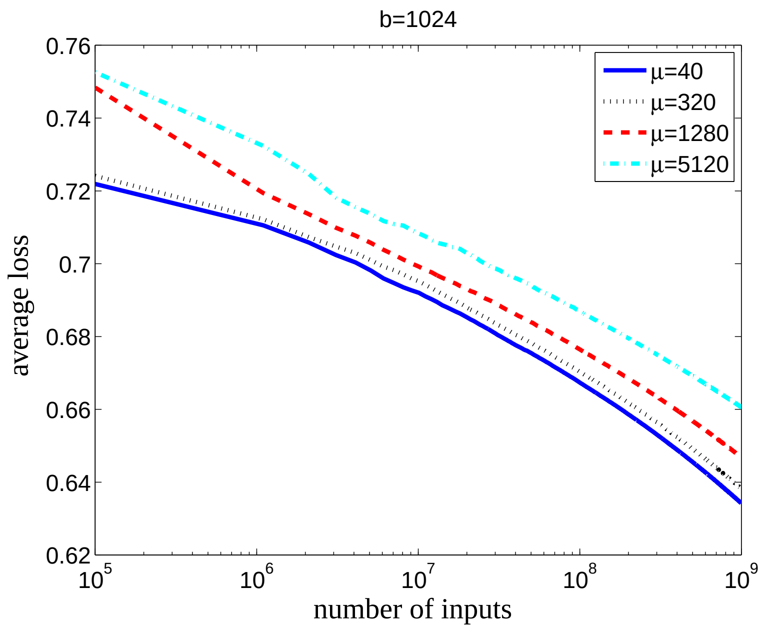

Figure 4: The effects of increased network latency. The loss of the DMB algorithm is reported with different latencies as measured by μ. In all cases, the batch size is fixed at $b = 1024$.

6.3 The Effects of Latency

Network latency results in the DMB discarding gradients, and slows down the algorithm's progress. The theoretical analysis shows that this waste is negligible in the asymptotic worst-case sense. However, latency will obviously have some negative effect on any finite prefix of the input stream. We examined what would happen if the single-link latency were much larger than our 0.5ms estimate (e.g., if the network is very congested or if the cluster nodes are scattered across multiple datacenters). Concretely, we set the cluster size to $k = 1024$ nodes, the batch size to $b = 1024$, and the single-link latency to $0.5, 1, 2, \ldots, 512$ ms. That is, 0.5ms mimics a realistic 1Gbs Ethernet link, while 512ms mimics a network whose latency between any two machines is 1024 times greater, namely, each vector-sum operation takes a full second to complete. Note that μ is still computed as before, namely, for latency $0.5 \cdot 2^p$, $\mu = 2^p 4 \log_2(k) = 2^p \cdot 40$. Figure 4 shows how the average loss curve reacts to four representative latencies. As expected, convergence rate degrades monotonically with latency. When latency is set to be 8 times greater than our realistic estimate for 1Gbs Ethernet, the effect is minor. When the latency is increased by a factor of 1024, the effect becomes more noticeable, but still quite small.

6.4 Optimal Mini-Batch Size

For our final experiment, we set out to find the optimal batch size for our problem on a given cluster size. Our theoretical analysis is too crude to provide a sufficient answer to this question. The theory basically says that setting $b = \Theta(m^\rho)$ is asymptotically optimal for any $\rho \in (0, 1/2)$, and

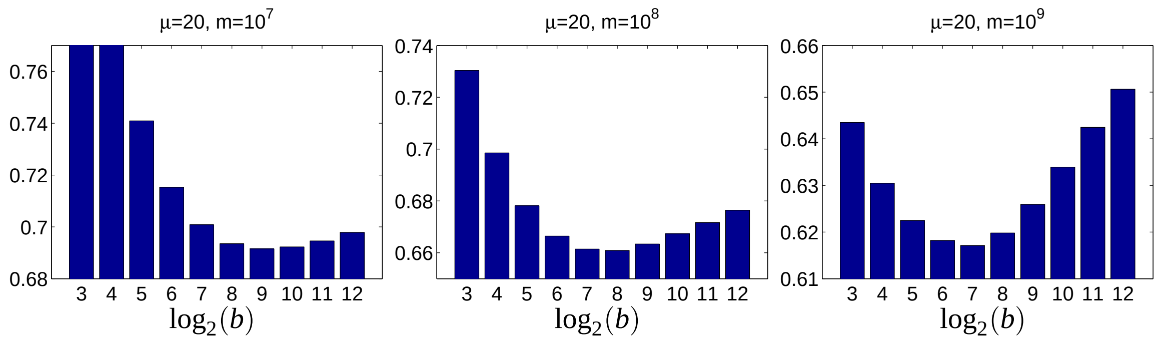

Figure 5: The effect of different mini-batch sizes (b) on the DBM algorithm. The DMB algorithm was applied with different batch sizes $b = 8, \ldots, 4096$. The loss is reported after 10^7 instances (left), 10^8 instances (middle) and 10^9 instances (right).

that $b = \Theta(m^{1/3})$ is a pretty good concrete choice. We have already seen that larger batch sizes accelerate the initial learning phase, even in a serial setting. We set the cluster size to $k = 32$ and set batch size to $8, 16, \ldots, 4096$. Note that $b = 32$ is the case where each node processes a single example before engaging in a vector-sum network operation. Figure 5 depicts the average loss after $10^7, 10^8$, and 10^9 inputs. As noted in the serial case, larger batch sizes ($b = 512$) are beneficial at first ($m = 10^7$), while smaller batch sizes ($b = 128$) are better in the end ($m = 10^9$).

6.5 Discussion

We presented an empirical evaluation of the serial mini-batch algorithm and its distributed version, the DMB algorithm, on a realistic web-scale online prediction problem. As expected, the DMB algorithm outperforms the näive no-communication algorithm. An interesting and somewhat unexpected observation is the fact that the use of large batches improves performance even in the serial setting. Moreover, the optimal batch size seems to generally decrease with time.

We also demonstrated the effect of network latency on the performance of the DMB algorithm. Even for relatively large values of μ, the degradation in performance was modest. This is an encouraging indicator of the efficiency and robustness of the DMB algorithm, even when implemented in a high-latency environment, such as a grid.

7. Related Work

In recent years there has been a growing interest in distributed online learning and distributed optimization.

Langford et al. (2009) address the distributed online learning problem, with a similar motivation to ours: trying to address the scalability problem of online learning algorithms which are inherently sequential. The main observation Langford et al. (2009) make is that in many cases, computing the gradient takes much longer than computing the update according to the online prediction algorithm. Therefore, they present a pipeline computational model. Each worker alternates between computing the gradient and computing the update rule. The different workers are synchronized such that no two workers perform an update simultaneously.

Similar to results presented in this paper, Langford et al. (2009) attempted to show that it is possible to achieve a cumulative regret of $O(\sqrt{m})$ with k parallel workers, compared to the $O(\sqrt{km})$ of the naïve solution. However their work suffers from a few limitations. First, their proofs only hold for unconstrained convex optimization where no projection is needed. Second, since they work in a model where one node at a time updates a shared predictor, while the other nodes compute gradients, the scalability of their proposed method is limited by the ratio between the time it takes to compute a gradient to the time it takes to run the update rule of the serial online learning algorithm.

In another related work, Duchi et al. (2010) present a distributed dual averaging method for optimization over networks. They assume the loss functions are Lipschitz continuous, but their gradients may not be. Their method does not need synchronization to average gradients computed at the same point. Instead, they employ a distributed consensus algorithm on all the gradients generated by different processors at different points. When applied to the stochastic online prediction setting, even for the most favorable class of communication graphs, with constant spectral gaps (e.g., expander graphs), their best regret bound is $O\left(\sqrt{km}\log(m)\right)$. This bound is no better than one would get by running k parallel machines without communication (see Section 2.2).

In another recent work, Zinkevich et al. (2010) study a method where each node in the network runs the classic stochastic gradient method, using random subsets of the overall data set, and only aggregate their solutions in the end (by averaging their final weight vectors). In terms of online regret, it is obviously the same as running k machines independently without communication. So a more suitable measure is the optimality gap (defined in Section 5) of the final averaged predictor. Even with respect to this measure, their expected optimality gap does not show advantage over running k machines independently. A similar approach was also considered by Nesterov and Vial (2008) and an experimental study of such a method was reported in Harrington et al. (2003).

A key difference between our DMB framework and many related work is that DMB does not consider distributed comuting as a constraint to overcome. Instead, our novel use of the variance-based regret bounds can exploit parallel/distributed computing to obtain the asymptotic optimal regret bound. Beyond the asymptotic optimality of our bounds, our work has other features that set it apart from previous work. As far as we know, we are the first to propose a general principled framework for distributing many gradient-based update rule, with a concrete regret analysis for the large family of mirror descent and dual averaging update rules. Additionally, our work is the first to explicitly include network latency in our regret analysis, and to theoretically guarantee that a large latency can be overcome by setting parameters appropriately.

8. Conclusions and Further Research

The increase in serial computing power of modern computers is out-paced by the growth rate of web-scale prediction problems and data sets. Therefore, it is necessary to adopt techniques that can harness the power of parallel and distributed computers.

In this work we studied the problems of distributed stochastic online prediction and distributed stochastic optimization. We presented a family of distributed online algorithms with asymptotically optimal regret and optimality gap guarantees. Our algorithms use the distributed computing infrastructure to reduce the variance of stochastic gradients, which essentially reduces the noise in the algorithm's updates. Our analysis shows that asymptotically, a distributed computing system can perform as well as a hypothetical fast serial computer. This result is far from trivial, and much of the prior art in the field did not show any provable gain by using distributed computers.

While the focus of this work is the theoretical analysis of a distributed online prediction algorithm, we also presented experiments on a large-scale real-world problem. Our experiments showed that indeed the DMB algorithm outperforms other simple solutions. They also suggested that improvements can be made by optimizing the batch size and adjusting the learning rate based on empirical measures.

Our formal analysis hinges on the fact that the regret bounds of many stochastic online update rules scale with the variance of the stochastic gradients when the loss function is smooth. It is unclear if smoothness is a necessary condition, or if it can be replaced with a weaker assumption. In principle, our results apply in a broader setting. For any serial update rule ϕ with a regret bound of $\psi(\sigma^2, m) = C\sigma\sqrt{m} + o\left(\sqrt{m}\right)$, the DMB algorithm and its variants have the optimal regret bound of $C\sigma\sqrt{m} + o\left(\sqrt{m}\right)$, provided that the bound $\psi(\sigma^2, m)$ applies equally to the function f and to the function

$$\bar{f}(w, (z_1, \ldots, z_b)) \; = \; \frac{1}{b} \sum_{s=1}^{b} f(w, z_s) \; .$$

Note that this result holds independently of the network size k and the network latency μ. Extending our results to non-smooth functions is an interesting open problem. A more ambitious challenge is to extend our results to the non-stochastic case, where inputs may be chosen by an adversary.

An important future direction is to develop distributed learning algorithms that perform robustly and efficiently on heterogeneous clusters and in asynchronous distributed environments. This direction has been further explored in Dekel et al. (2011). For example, one can use the following simple reformulation of the DMB algorithm in a master-workers setting: each worker process inputs at its own pace and periodically sends the accumulated gradients to the master; the master applies the update rule whenever the number of accumulated gradients reaches a certain threshold and broadcasts the new predictor back to the workers. In a dynamic environment, where the network can be partitioned and reconnected and where nodes can be added and removed, a new master (or masters) can be chosen as needed by a standard leader election algorithm. We refer the reader to Dekel et al. (2011) for more details.

A central property of our method is that all of the gradients in a batch must be taken at the same prediction point. In an asynchronous distributed computing environment (see, e.g., Tsitsiklis et al., 1986; Bertsekas and Tsitsiklis, 1989), this can be quite wasteful. In order to reduce the waste generated by the need for global synchronization, we may need to allow different nodes to accumulate gradients at different yet close points. Such a modification is likely to work since the smoothness assumption precisely states that gradients of nearby points are similar. There have been extensive studies on distributed optimization with inaccurate or delayed subgradient information, but mostly without the smoothness assumption (e.g., Nedić et al., 2001; Nedić and Ozdaglar, 2009). We believe that our main results under the smoothness assumption can be extended to asynchronous and distributed environments as well.

Acknowledgments

The authors are grateful to Karthik Sridharan for spotting the fact that for non-Euclidean spaces, our variance-reduction argument had to be modified with a space-dependent constant (see the discussion at the end of Section 4.1). We also thank the reviewers for their helpful comments.

Appendix A. Smooth Stochastic Online Prediction in the Serial Setting

In this appendix, we prove expected regret bounds for stochastic dual averaging and stochastic mirror descent applied to smooth loss functions. In the main body of the paper, we discussed only the Euclidean special case of these algorithms, while here we present the algorithms and regret bounds in their full generality. In particular, Theorem 1 is a special case of Theorem 9, and Theorem 2 is a special case of Theorem 7.

Recall that we observe a stochastic sequence of inputs $z_1, z_2, \ldots$, where each $z_i \in \mathcal{Z}$. Before observing each z_i we predict $w_i \in W$, and suffer a loss $f(w_i, z_i)$. We assume W is a closed convex subset of a finite dimensional vector space $\mathcal{V}$ with endowed norm $\|\cdot\|$. We assume that $f(w,z)$ is convex and differentiable in w, and we use $\nabla_w f(w,z)$ to denote the gradient of f with respect to its first argument. $\nabla_w f(w,z)$ is a vector in the dual space $\mathcal{V}^*$, with endowed norm $\|\cdot\|_*$.

We assume that $f(\cdot,z)$ is L-smooth for any realization of z. Namely, we assume that $f(\cdot,z)$ is differentiable and that

$$\forall z \in \mathcal{Z}, \quad \forall w, w' \in W, \qquad \|\nabla_w f(w,z) - \nabla_w f(w',z)\|_* \le L\|w - w'\| \ .$$

We define $F(w) = \mathbb{E}_z[f(w,z)]$ and note that $\nabla_w F(w) = \mathbb{E}_z[\nabla_w f(w,z)]$ (see Rockafellar and Wets, 1982). This implies that

$$\forall w, w' \in W, \qquad \|\nabla_w F(w) - \nabla_w F(w')\|_* \le L\|w - w'\| \ .$$

In addition, we assume that there exists a constant $\sigma \ge 0$ such that

$$\forall w \in W, \qquad \mathbb{E}_z[\|\nabla_w f(w,z) - \nabla_w E_z[f(w,z)]\|_*^2] \le \sigma^2 \ .$$

We assume that $w^\star = \arg\min_{w \in W} F(w)$ exists, and we abbreviate $F^\star = F(w^\star)$.

Under the above assumptions, we are concerned with bounding the expected regret $\mathbb{E}[R(m)]$, where regret is defined as

$$R(m) \ = \ \sum_{i=1}^{m} \left(f(w_i, z_i) - f(w^\star, z_i) \right) \ .$$

In order to present the algorithms in their full generality, we first recall the concepts of strongly convex function and Bregman divergence.

A function $h : W \to \mathbb{R} \cup \{+\infty\}$ is said to be *μ-strongly convex* with respect to $\|\cdot\|$ if

$$\forall \alpha \in [0,1], \quad \forall u, v \in W, \quad h(\alpha u + (1-\alpha)v) \le \alpha h(u) + (1-\alpha)h(v) - \frac{\mu}{2}\alpha(1-\alpha)\|u - v\|^2 \ .$$

If h is μ-strongly convex then for any $u \in \operatorname{dom} h$, and $v \in \operatorname{dom} h$ that is sub-differentiable, then

$$\forall s \in \partial h(v), \quad h(u) \ge h(v) + \langle s, u - v \rangle + \frac{\mu}{2}\|u - v\|^2 \ .$$

(See, e.g., Goebel and Rockafellar, 2008.) If a function h is strictly convex and differentiable (on an open set contained in $\operatorname{dom} h$), then we can defined the Bregman divergence generated by h as

$$d_h(u,v) = h(u) - h(v) - \langle \nabla h(v), u - v \rangle \ .$$

We often drop the subscript h in d_h when it is obvious from the context. Some key properties of the Bregman divergence are:

- $d(u,v) \geq 0$, and the equality holds if and only if $u = v$.
- In general $d(u,v) \neq d(v,u)$, and d may not satisfy the triangle inequality.
- The following *three-point identity* follows directly from the definition:

$$d(u,w) = d(u,v) + d(v,w) + \langle \nabla h(v) - \nabla h(w), u - v \rangle \ .$$

The following inequality is a direct consequence of the μ-strong convexity of h:

$$d(u,v) \geq \frac{\mu}{2}\|u-v\|^2 \ . \tag{12}$$

A.1 Stochastic Dual Averaging

The proof techniques for the stochastic dual averaging method are adapted from those for the accelerated algorithms presented in Tseng (2008) and Xiao (2010).

Let $h : W \to \mathbb{R}$ be a 1-strongly convex function. Without loss of generality, we can assume that $\min_{w \in W} h(w) = 0$. In the stochastic dual averaging method, we predict each w_i by

$$w_{i+1} = \arg\min_{w \in W} \left\{ \left\langle \sum_{j=1}^{i} g_j, w \right\rangle + (L + \beta_{i+1}) h(w) \right\} \ , \tag{13}$$

where g_j denotes the stochastic gradient $\nabla_w f(w_j, z_j)$, and $(\beta_i)_{i\geq 1}$ is a sequence of positive and nondecreasing parameters (i.e., $\beta_{i+1} \geq \beta_i$). As a special case of the above, we initialize w_1 to

$$w_1 = \arg\min_{w \in W} h(w) \ . \tag{14}$$

We are now ready to state a bound on the expected regret of the dual averaging method, in the smooth stochastic case.

Theorem 7 *The expected regret of the stochastic dual averaging method is bounded as*

$$\forall m, \quad \mathbb{E}[R(m)] \leq (F(w_1) - F(w^\star)) + (L + \beta_m) h(w^\star) + \frac{\sigma^2}{2} \sum_{i=1}^{m-1} \frac{1}{\beta_i}.$$

The optimal choice of β_i is exactly of order $\sqrt{i}$. More specifically, let $\beta_i = \gamma\sqrt{i}$, where γ is a positive parameter. Then Theorem 7 implies that

$$\mathbb{E}[R(m)] \leq (F(w_1) - F(w^\star)) + L h(w^\star) + \left(\gamma h(w^\star) + \frac{\sigma^2}{\gamma}\right)\sqrt{m}.$$

Choosing $\gamma = \sigma/\sqrt{h(w^\star)}$ gives

$$\mathbb{E}[R(m)] \leq (F(w_1) - F(w^\star)) + Lh(w^\star) + \left(2\sigma\sqrt{h(w^\star)}\right)\sqrt{m}.$$

If $\nabla F(w^\star) = 0$ (this is certainly the case if W is the whole space), then we have

$$F(w_1) - F(w^\star) \leq \frac{L}{2}\|w_1 - w^\star\|^2 \leq Lh(w^\star).$$

Then the expected regret bound can be simplified as

$$\mathbb{E}[R(m)] \leq 2Lh(w^\star) + \left(2\sigma\sqrt{h(w^\star)}\right)\sqrt{m}.$$

To prove Theorem 7 we require the following fundamental lemma, which can be found, for example, in Nesterov (2005), Tseng (2008) and Xiao (2010).

Lemma 8 *Let W be a closed convex set, φ be a convex function on W, and h be μ-strongly convex on W with respect to $\|\cdot\|$. If*

$$w^+ = \arg\min_{w\in W}\left\{\varphi(w) + h(w)\right\},$$

then

$$\forall w \in W, \qquad \varphi(w) + h(w) \geq \varphi(w^+) + h(w^+) + \frac{\mu}{2}\|w - w^+\|^2.$$

With Lemma 8, we are now ready to prove Theorem 7.

Proof First, we define the linear functions

$$\ell_i(w) = F(w_i) + \langle \nabla F(w_i), w - w_i\rangle, \qquad \forall i \geq 1,$$

and (using the notation $g_i = \nabla f(w_i, z_i)$)

$$\hat{\ell}_i(w) = F(w_i) + \langle g_i, w - w_i\rangle = \ell_i(w) + \langle q_i, w - w_i\rangle,$$

where

$$q_i = g_i - \nabla F(w_i).$$

Therefore, the stochastic dual averaging method specified in Equation (13) is equivalent to

$$w_i = \arg\min_{w\in W}\left\{\sum_{j=1}^{i-1}\hat{\ell}_j(w) + (L+\beta_i)h(w)\right\}.$$

Using the smoothness assumption, we have (e.g., Nesterov 2004, Lemma 1.2.3)

$$\begin{aligned}
F(w_{i+1}) &\leq \ell_i(w_{i+1}) + \frac{L}{2}\|w_{i+1} - w_i\|^2 \\
&= \hat{\ell}_i(w_{i+1}) + \frac{L+\beta_i}{2}\|w_{i+1} - w_i\|^2 - \langle q_i, w_{i+1} - w_i\rangle - \frac{\beta_i}{2}\|w_{i+1} - w_i\|^2 \\
&\leq \hat{\ell}_i(w_{i+1}) + \frac{L+\beta_i}{2}\|w_{i+1} - w_i\|^2 + \|q_i\|_*\|w_{i+1} - w_i\| - \frac{\beta_i}{2}\|w_{i+1} - w_i\|^2 \\
&= \hat{\ell}_i(w_{i+1}) + \frac{L+\beta_i}{2}\|w_{i+1} - w_i\|^2 - \left(\frac{1}{\sqrt{2\beta_i}}\|q_i\|_* - \sqrt{\frac{\beta_i}{2}}\|w_{i+1} - w_i\|\right)^2 + \frac{\|q_i\|_*^2}{2\beta_i} \\
&\leq \hat{\ell}_i(w_{i+1}) + \frac{L+\beta_i}{2}\|w_{i+1} - w_i\|^2 + \frac{\|q_i\|_*^2}{2\beta_i}. \qquad (15)
\end{aligned}$$

Next we use Lemma 8 with $\varphi(w) = \sum_{j=1}^{i-1} \hat{\ell}_j(w)$ and $\mu = (L + \beta_i)$,

$$\sum_{j=1}^{i-1} \hat{\ell}_j(w_{i+1}) + (L+\beta_i)h(w_{i+1}) \geq \sum_{j=1}^{i-1} \hat{\ell}_j(w_i) + (L+\beta_i)h(w_i) + \frac{L+\beta_i}{2}\|w_{i+1} - w_i\|^2,$$

Combining the above inequality with Equation (15), we have

$$\begin{aligned} F(w_{i+1}) &\leq \hat{\ell}_i(w_{i+1}) + \sum_{j=1}^{i-1} \hat{\ell}_j(w_{i+1}) + (L+\beta_i)h(w_{i+1}) - \sum_{j=1}^{i-1} \hat{\ell}_j(w_i) - (L+\beta_i)h(w_i) + \frac{\|q_i\|_*^2}{2\beta_i} \\ &\leq \sum_{j=1}^{i} \hat{\ell}_j(w_{i+1}) + (L+\beta_{i+1})h(w_{i+1}) - \sum_{j=1}^{i-1} \hat{\ell}_j(w_i) - (L+\beta_i)h(w_i) + \frac{\|q_i\|_*^2}{2\beta_i}, \end{aligned}$$

where in the last inequality, we used the assumptions $\beta_{i+1} > \beta_i > 0$ and $h(w_{i+1}) \geq 0$. Summing the above inequality from $i = 1$ to $i = m-1$, we have

$$\begin{aligned} \sum_{i=2}^{m} F(w_i) &\leq \sum_{i=1}^{m-1} \hat{\ell}_i(w_m) + (L+\beta_m)h(w_m) + \sum_{i=1}^{m-1} \frac{\|q_i\|_*^2}{2\beta_i} \\ &\leq \sum_{i=1}^{m-1} \hat{\ell}_i(w^\star) + (L+\beta_m)h(w^\star) + \sum_{i=1}^{m-1} \frac{\|q_i\|_*^2}{2\beta_i} \\ &\leq \sum_{i=1}^{m-1} \ell_i(w^\star) + (L+\beta_m)h(w^\star) + \sum_{i=1}^{m-1} \frac{\|q_i\|_*^2}{2\beta_i} + \sum_{i=1}^{m-1} \langle q_i, w^\star - w_i \rangle \\ &\leq (m-1)F(w^\star) + (L+\beta_i)h(w^\star) + \sum_{i=1}^{m-1} \frac{\|q_i\|_*^2}{2\beta_i} + \sum_{i=1}^{m-1} \langle q_i, w^\star - w_i \rangle. \end{aligned}$$

Therefore,

$$\sum_{i=2}^{m} \big(F(w_i) - F(w^\star)\big) \leq (L+\beta_m)h(w^\star) + \sum_{i=1}^{m-1} \frac{\|q_i\|_*^2}{2\beta_i} + \sum_{i=1}^{m-1} \langle q_i, w^\star - w_i \rangle. \tag{16}$$

Notice that each w_i is a deterministic function of $z_1, \ldots, z_{i-1}$, so

$$\mathbb{E}_{z_i}\big(\langle q_i, w^\star - w_i \rangle \,|\, z_1, \ldots, z_{i-1}\big) = 0$$

by recalling the definition $q_i = \nabla f(w_i, z_i) - \nabla F(w_i)$. Taking expectation of both sides of Equation (16) with respect to $z_1, \ldots, z_m$, and adding the term $F(w_1) - F(w^\star)$, we have

$$\mathbb{E} \sum_{i=1}^{m} \big(F(w_i) - F(w^\star)\big) \leq F(w_1) - F(w^\star) + (L+\beta_m)h(w^\star) + \sum_{i=1}^{m-1} \frac{\sigma^2}{2\beta_i}.$$

Theorem 7 is proved by further noticing

$$\mathbb{E}\, f(w_i, z_i) = \mathbb{E}\, F(w_i), \qquad \mathbb{E}\, f(w^\star, z_i) = F(w^\star), \qquad \forall i \geq 1,$$

which are due to the fact that w_i is a deterministic function of $z_0, \ldots, z_{i-1}$. ■

A.2 Stochastic Mirror Descent

Variance-based convergence rates for the stochastic Mirror Descent methods are due to Juditsky et al. (2011), and were extended to an accelerated stochastic Mirror Descent method by Lan (2009). For completeness, we adapt their proofs to the context of regret for online prediction problems.

Again let $h : W \to \mathbb{R}$ be a differentiable 1-strongly convex function with $\min_{w \in W} h(w) = 0$. Also let d be the Bregman divergence generated by h. In the stochastic mirror descent method, we use the same initialization as in the dual averaging method (see Equation (14)) and then we set

$$w_{i+1} = \arg\min_{w \in W} \Big\{ \langle g_i, w \rangle + (L + \beta_i) d(w, w_i) \Big\}, \qquad i \geq 1.$$

As in the dual averaging method, we assume that the sequence $(\beta_i)_{i \geq 1}$ to be positive and nondecreasing.

Theorem 9 *Assume that the convex set W is closed and bounded. In addition assume $d(u,v)$ is bounded on W and let*

$$D^2 = \max_{u,v \in W} d(u,v).$$

Then the expected regret of the stochastic mirror descent method is bounded as

$$\mathbb{E}[R(m)] \leq (F(w_1) - F(w^\star)) + (L + \beta_m) D^2 + \frac{\sigma^2}{2} \sum_{i=1}^{m-1} \frac{1}{\beta_i}.$$

Similar to the dual averaging case, using the sequence of parameters $\beta_i = (\sigma/D)\sqrt{i}$ gives the expected regret bound

$$\mathbb{E}[R(m)] \leq (F(w_1) - F(w^\star)) + LD^2 + (2\sigma D)\sqrt{m}.$$

Again, if $\nabla F(w^\star) = 0$, we have $F(w_1) - F(w^\star) \leq (L/2)\|w_1 - w^\star\|^2 \leq Lh(w^\star) \leq LD^2$, thus the simplified bound

$$\mathbb{E}[R(m)] \leq 2LD^2 + (2\sigma D)\sqrt{m}.$$

We note that here we have stronger assumptions than in the dual averaging case. These assumptions are certainly satisfied by using the standard Euclidean distance $d(u,v) = (1/2)\|u - v\|_2^2$ on a compact convex set W. However, it excludes the case of using the KL-divergence $d(u,v) = \sum_{i=1}^n u_i \log(u_i/v_i)$ on the simplex, because the KL-divergence is unbounded on the simplex. Nevertheless, it is possible to remove such restrictions by considering other variants of the stochastic mirror descent method. For example, if we use a constant β_i that depends on the prior knowledge of the number of total steps to be performed, then we can weaken the assumption and replace D in the above bounds by $\sqrt{h(w^\star)}$. More precisely, we have

Theorem 10 *Suppose we know the total number of steps m to be performed by the stochastic mirror descent method ahead of time. Then by using the initialization in Equation (14) and the constant parameter*

$$\beta_i = \frac{\sigma}{\sqrt{2h(w^\star)}}\sqrt{m},$$

we have the expected regret bound

$$\mathbb{E}[R(m)] \leq (F(w_1) - F(w^\star)) + Lh(w^\star) + \sigma\sqrt{2h(w^\star)}\sqrt{m}.$$

Theorem 10 is essentially the same as a result in Lan (2009), who also developed an accelerated versions of the stochastic mirror descent method. To prove Theorem 9 and Theorem 10 we need the following standard Lemma, which can be found in Chen and Teboulle (1993), Lan et al. (2011) and Tseng (2008).

Lemma 11 *Let W be a closed convex set, φ be a convex function on W, and h be a differentiable, strongly convex function on W. Let d be the Bregman divergence generated by h. Given $u \in W$, if*

$$w^+ = \operatorname*{arg\,min}_{w \in W} \left\{ \varphi(w) + d(w,u) \right\},$$

then

$$\varphi(w) + d(w,u) \geq \varphi(w^+) + d(w^+,u) + d(w,w^+).$$

We are ready to prove Theorem 9 and Theorem 10.

Proof We start with the inequality in Equation (15). Using Equation (12) with $\mu = 1$ gives

$$F(w_{i+1}) \leq \hat{\ell}_i(w_{i+1}) + (L+\beta_i) d(w_{i+1},w_i) + \frac{\|q_i\|_*^2}{2\beta_i}. \tag{17}$$

Now using Lemma 11 with $\varphi(w) = \hat{\ell}_i(w)$ yields

$$\hat{\ell}_i(w_{i+1}) + (L+\beta_i)d(w_{i+1},w_i) \leq \hat{\ell}_i(w^\star) + (L+\beta_i)d(w^\star,w_i) - (L+\beta_i)d(w^\star,w_{i+1}).$$

Combining with Equation (17) gives

$$\begin{aligned}
F(w_{i+1}) \;&\leq\; \hat{\ell}_i(w^\star) + (L+\beta_i)d(w^\star,w_i) - (L+\beta_i)d(w^\star,w_{i+1}) + \frac{\|q_i\|_*^2}{2\beta_i} \\
&= \; \ell_i(w^\star) + (L+\beta_i)d(w^\star,w_i) - (L+\beta_{i+1})d(w^\star,w_{i+1}) + (\beta_{i+1}-\beta_i)d(w^\star,w_{i+1}) \\
&\qquad + \frac{\|q_i\|_*^2}{2\beta_i} + \langle q_i, w^\star - w_i \rangle \\
&\leq \; F(w^\star) + (L+\beta_i)d(w^\star,w_i) - (L+\beta_{i+1})d(w^\star,w_{i+1}) + (\beta_{i+1}-\beta_i)D^2 \\
&\qquad + \frac{\|q_i\|_*^2}{2\beta_i} + \langle q_i, w^\star - w_i \rangle,
\end{aligned}$$

where in the last inequality, we used the definition of D^2 and the assumption that $\beta_{i+1} \geq \beta_i$. Summing the above inequality from $i = 1$ to $i = m-1$, we have

$$\begin{aligned}
\sum_{i=2}^{m} F(w_i) \quad &\leq \quad (m-1)F(w^\star) + (L+\beta_1)d(w^\star,w_1) - (L+\beta_m)d(w^\star,w_m) + (\beta_m - \beta_1)D^2 \\
&\qquad\qquad + \sum_{i=1}^{m-1} \frac{\|q_i\|_*^2}{2\beta_i} + \sum_{i=1}^{m-1} \langle q_i, w^\star - w_i \rangle.
\end{aligned}$$

Notice that $d(w^\star,w_i) \geq 0$ and $d(w^\star,w_1) \leq D^2$, so we have

$$\sum_{i=2}^{m} F(w_i) \leq (m-1)F(w^\star) + (L+\beta_m)D^2 + \sum_{i=1}^{m-1} \frac{\|q_i\|_*^2}{2\beta_i} + \sum_{i=1}^{m-1} \langle q_i, w^\star - w_i \rangle.$$

The rest of the proof for Theorem 9 is similar to that for the dual averaging method (see arguments following Equation (16)).

Finally we prove Theorem 10. From the proof of Theorem 9 above, we see that if $\beta_i = \beta_m$ is a constant for all $i = 1, \ldots, m$, then we have

$$\mathbb{E} \sum_{i=2}^{m} (F(w_i) - F(w^\star)) \leq (L + \beta_m) d(w^\star, w_1) + \frac{\sigma^2}{2} \sum_{i=1}^{m-1} \frac{1}{\beta_i}.$$

Notice that for the above result, we do not need to assume boundedness of W, nor boundedness of the Bregman divergence $d(u,v)$. Since we use $w_1 = \arg\min_{w \in W} h(w)$ and assume $h(w_1) = 0$ (without loss of generality), it follows $d(w^\star, w_1) \leq h(w^\star)$. Plugging in $\beta_m = (\sigma/\sqrt{2h(w^\star)})\sqrt{m}$ gives the desired result. ■

Appendix B. High-Probability Bounds

For simplicity, the theorems stated throughout the paper involved bounds on the expected regret, $\mathbb{E}[R(m)]$. A stronger type of result is a high-probability bound, where $R(m)$ itself is bounded with arbitrarily high probability $1 - \delta$, and the bound having only logarithmic dependence on δ. Here, we demonstrate how our theorems can be extended to such high-probability bounds.

First, we need to justify that the expected regret bounds for the online prediction rules discussed in Appendix A have high-probability versions. For simplicity, we will focus on a high-probability version of the regret bound for dual averaging (Theorem 7), but exactly the same technique will work for stochastic mirror descent (Theorem 9 and Theorem 10). With these results in hand, we will show how our main theorem for distributed learning using the DMB algorithm (Theorem 4) can be extended to a high-probability version. Identical techniques will work for the other theorems presented in the paper.

Before we begin, we will need to make a few additional mild assumptions. First, we assume that there are positive constants B, G such that $|f(w,z)| \leq B$ and $\|\nabla_w f(w,z)\| \leq G$ for all $w \in W$ and $z \in \mathcal{Z}$. Second, we assume that there is a positive constant $\hat{\sigma}$ such that $\mathrm{Var}_z(f(w,z) - f(w^\star, z)) \leq \hat{\sigma}^2$ for all $w \in W$ (note that $\hat{\sigma}^2 \leq 4B^2$ always holds). Third, that W has a bounded diameter D, namely $\|w - w'\| \leq D$ for all $w, w' \in W$.

Under these assumptions, we can show the following high-probability version of Theorem 7.

Theorem 12 *For any m and any $\delta \in (0,1]$, the regret of the stochastic dual averaging method is bounded with probability at least $1 - \delta$ over the sampling of $z_1, \ldots, z_m$ by*

$$\begin{aligned} R(m) \leq\ & (F(w_1) - F(w^\star)) + (L + \beta_m) h(w^\star) + \frac{\sigma^2}{2} \sum_{i=1}^{m-1} \frac{1}{\beta_i} \\ & + 2\log(2/\delta) \left(DG + \frac{2G^2}{\beta_1} \right) \sqrt{1 + 36 \frac{G^2 \sigma^2 \sum_{i=1}^{m} \frac{1}{\beta_i^2} + D^2 \sigma^2 m}{\log(2/\delta)}} \\ & + 4\log(2/\delta) B \sqrt{1 + \frac{18 m \hat{\sigma}^2}{\log(2/\delta)}}. \end{aligned}$$

Proof The proof of the theorem is identical to the one of Theorem 7, up to Equation (16):

$$\sum_{i=2}^{m}\big(F(w_i)-F(w^\star)\big) \le (L+\beta_m)h(w^\star)+\sum_{i=1}^{m-1}\frac{\|q_i\|^2}{2\beta_i}+\sum_{i=1}^{m-1}\langle q_i, w^\star-w_i\rangle. \tag{18}$$

In the proof of Theorem 7, we proceeded by taking expectations of both sides with respect to the sequence $z_1,\dots,z_m$. Here, we will do things a bit differently.

The main technical tool we use is a well-known Bernstein-type inequality for martingales (e.g., Cesa-Bianchi and Lugosi, 2006, Lemma A.8), an immediate corollary of which can be stated as follows: suppose $x_1,\dots,x_m$ is a martingale difference sequence with respect to the sequence $z_1,\dots,z_m$, such that $|x_i|\le b$, and let

$$v=\sum_{i=1}^{m}\mathrm{Var}(x_i|z_1,\dots,z_{i-1}).$$

Then for any $\delta\in(0,1)$, it holds with probability at least $1-\delta$ that

$$\sum_{i=1}^{m}x_i \le b\log(1/\delta)\sqrt{1+\frac{18v}{\log(1/\delta)}}. \tag{19}$$

Recall the definition $q_i=\nabla f(w_i,z_i)-\nabla F(w_i)$, and let $\sigma_i^2=\mathbb{E}[\|q_i\|^2]$. Note that $\sigma_i^2\le\sigma^2$. We will first use this result for the sequence

$$x_i=\frac{\|q_i\|^2-\sigma_i^2}{2\beta_i}+\langle q_i,w^\star-w_i\rangle.$$

It is easily seen that $\mathbb{E}_{z_i}[x_i|z_1,\dots,z_{i-1}]=0$, so it is indeed a martingale difference sequence w.r.t. $z_1,\dots,z_m$. Moreover, $|\langle q_i,w^\star-w_i\rangle|\le D\|q_i\|\le 2DG$, $\|q_i\|^2\le 4G^2$. In terms of the variances, let Var_{z_i} and $\mathbb{E}_{z_i}$ be shorthand for the variance (resp. expectation) over z_i conditioned over $z_1,\dots,z_{i-1}$. Then

$$\begin{aligned}
\mathrm{Var}_{z_i}(x_i) &\le 2\mathrm{Var}_{z_i}\left(\frac{\|q_i\|^2-\sigma_i^2}{2\beta_i}\right)+2\mathrm{Var}_{z_i}\left(\langle q_i,w^\star-w_i\rangle\right)\\
&\le \frac{1}{2}\mathbb{E}_{z_i}\left(\frac{\|q_i\|^4}{\beta_i^2}\right)+2\mathbb{E}_{z_i}[(\langle q_i,w^\star-w_i\rangle)^2]\\
&\le 2G^2\mathbb{E}_{z_i}\left(\frac{\|q_i\|^2}{\beta_i^2}\right)+2\|w^\star-w_i\|^2\mathbb{E}_{z_i}[\|q_i\|^2]\\
&\le 2G^2\frac{\sigma_i^2}{\beta_i^2}+2D^2\sigma_i^2 \;\le\; 2G^2\frac{\sigma^2}{\beta_i^2}+2D^2\sigma^2.
\end{aligned}$$

Combining these observations with Equation (19), we get that with probability at least $1-\delta$,

$$\sum_{i=1}^{m-1}\frac{\|q_i\|^2-\sigma^2}{\beta_i}+\langle q_i,w^\star-w_i\rangle \le \left(2DG+\frac{4G^2}{\beta_1}\right)\log(1/\delta)\sqrt{1+36\frac{G^2\sigma^2\sum_{i=1}^{m}\frac{1}{\beta_i^2}+D^2\sigma^2 m}{\log(1/\delta)}}. \tag{20}$$

A similar type of bound can be derived for the sequence $x_i=(f(w_i,z_i)-f(w^\star,z_i))-(F(w_i)-F(w^\star))$. It is easily verified to be a martingale difference sequence w.r.t. $z_1,\dots,z_m$, since

$$\mathbb{E}\left[(f(w_i,z_i)-f(w^\star,z_i))-(F(w_i)-F(w^\star))\,|z_1,\dots,z_{i-1}\right]=0.$$

Also,

$$|(f(w_i,z_i)-f(w^\star,z_i))-(F(w_i)-F(w^\star))| \le 4B,$$

and

$$\begin{aligned}\mathrm{Var}_{z_i}\left(\left(f(w_i,z_i)-f(w^\star,z_i)\right)-\left(F(w_i)-F(w^\star)\right)\right) &= \mathrm{Var}_{z_i}\left(f(w_i,z_i)-f(w^\star,z_i)\right)\\ &\le \hat{\sigma}^2 .\end{aligned}$$

So again using Equation (19), we have that with probability at least $1-\delta$ that

$$\sum_{i=1}^{m}(f(w_i,z_i)-f(w^\star,z_i))-(F(w_i)-F(w^\star)) \le 4B\log(1/\delta)\sqrt{1+\frac{18m\hat{\sigma}^2}{\log(1/\delta)}} . \tag{21}$$

Finally, adding $F(w_1)-F(w^\star)$ to both sides of Equation (18), and combining Equation (20) and Equation (21) with a union bound, the result follows. ■

Comparing the theorem to Theorem 7, and assuming that $\beta_i = \Theta(\sqrt{i})$, we see that the bound has additional $O(\sqrt{m})$ terms. However, the bound retains the important property of having the dominant terms multiplied by the variances $\sigma^2, \hat{\sigma}^2$. Both variances become smaller in the mini-batch setting, where the update rules are applied over averages of b such functions and their gradients. As we did earlier in the paper, let us think of this bound as an abstract function $\psi(\sigma^2,\hat{\sigma}^2,\delta,m)$. Notice that now, the regret bound also depends on the function variance $\hat{\sigma}^2$, and the confidence parameter δ.

Theorem 13 *Let f is an L-smooth convex loss function. Assume that the stochastic gradient $\nabla_w f(w,z_i)$ is bounded by a constant and has σ^2-bounded variance for all i and all w, and that $f(w,z_i)$ is bounded by a constant and has $\hat{\sigma}^2$-bounded variance for all i and for all w. If the update rule ϕ has a serial high-probability regret bound $\psi(\sigma^2,\hat{\sigma}^2,\delta,m)$. then with probability at least $1-\delta$, the total regret of Algorithm 3 over m examples is at most*

$$(b+\mu)\psi\left(\frac{\sigma^2}{b},\frac{\hat{\sigma}^2}{b},\delta,1+\frac{m}{b+\mu}\right)+O\left(\hat{\sigma}\sqrt{\left(1+\frac{\mu}{b}\right)\log(1/\delta)m}\right) .$$

Comparing the obtained bound to the one in Theorem 4, we note that we pay an additional $O(\sqrt{m})$ factor.

Proof The proof closely resembles the one of Theorem 4. We let $\bar{z}_j$ denote the first b inputs on batch j, and define $\bar{f}$ as the average loss on these inputs. Note that for any w, the variance of $\bar{f}(w,\bar{z}_j)$ is at most $\hat{\sigma}^2/b$, and the variance of $\nabla_w \bar{f}(w,z)$ is at most σ^2/b. Therefore, with probability at least $1-\delta$, it holds that

$$\sum_{j=1}^{\bar{m}}\left(\bar{f}(w_j,\bar{z}_j)-\bar{f}(w^\star,\bar{z}_j)\right) \le \psi\left(\frac{\sigma^2}{b},\frac{\hat{\sigma}^2}{b},\delta,\bar{m}\right) . \tag{22}$$

where $\bar{m}$ is the number of inputs given to the update rule ϕ. Let Z_j denote the set of all examples received between the commencement of batch j and the commencement of batch $j+1$, including the vector-sum phase in between ($b+\mu$ examples overall). In the proof of Theorem 4, we had that

$$\mathbb{E}\left[\left(\bar{f}(w_j,\bar{z}_j)-\bar{f}(w^\star,\bar{z}_j)\right)|w_j\right] = \mathbb{E}\left[\frac{1}{b+\mu}\sum_{z\in Z_j}(f(w_j,z_i)-f(w^\star,z_i)) \,\Big|\, w_j\right] ,$$

and thus the *expected value* of the left-hand side of Equation (22) equals the total regret, divided by $b+\mu$. Here, we need to work a bit harder. To do so, note that the sequence of random variables

$$\left(\frac{1}{b}\sum_{z\in\bar{Z}_j}\big(f(w_j,z)-f(w^\star,z)\big)\right)-\left(\frac{1}{b+\mu}\sum_{z\in Z_j}\big(f(w_j,z)-f(w^\star,z)\big)\right),$$

indexed by j, is a martingale difference sequence with respect to $Z_1,Z_2,\ldots$. Moreover, conditioned on $Z_1,\ldots,Z_{j-1}$, the variance of each such random variable is at most $4\hat{\sigma}^2/b$. To see why, note that the first sum has conditional variance $\hat{\sigma}^2/b$, since the summands are independent and each has variance $\hat{\sigma}^2$. Similarly, the second sum has conditional variance $\hat{\sigma}^2/(b+\mu)\le\hat{\sigma}^2/b$. Applying the Bernstein-type inequality for martingales discussed in the proof of Theorem 12, we get that with probability at least $1-\delta$,

$$\sum_{j=1}^{\bar{m}}\frac{1}{b+\mu}\sum_{z\in Z_j}\big(f(w_j,z)-f(w^\star,z)\big)\le\sum_{j=1}^{\bar{m}}\frac{1}{b}\sum_{z\in\bar{Z}_j}\big(f(w_j,z)-f(w^\star,z)\big)+O\left(\hat{\sigma}\sqrt{\frac{\bar{m}\log(1/\delta)}{b}}\right),$$

where the O-notation hides only a (linear) dependence on the absolute bound over $|f(w,z)|$ for all w,z, that we assume to hold.

Combining this and Equation (22) with a union bound, we get that with probability at least $1-\delta$,

$$\sum_{j=1}^{\bar{m}}\sum_{z\in Z_j}\big(f(w_j,z)-f(w^\star,z)\big)\le(b+\mu)\psi\left(\frac{\sigma^2}{b},\frac{\hat{\sigma}^2}{b},\delta,\frac{m}{b+\mu}\right)+O\left((b+\mu)\hat{\sigma}\sqrt{\frac{\bar{m}\log(1/\delta)}{b}}\right).$$

If $b+\mu$ divides m, then $\bar{m}=m/(b+\mu)$, and we get a bound of the form

$$(b+\mu)\psi\left(\frac{\sigma^2}{b},\frac{\hat{\sigma}^2}{b},\delta,\frac{m}{b+\mu}\right)+O\left(\hat{\sigma}\sqrt{\left(1+\frac{\mu}{b}\right)\log(1/\delta)m}\right).$$

Otherwise, we repeat the ideas of Theorem 3 to get the regret bound. ■

References

J. Abernethy, A. Agarwal, A. Rakhlin, and P. L. Bartlett. A stochastic view of optimal regret through minimax duality. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT)*, 2009.

D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, New York, 1997.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

G. Chen and M. Teboulle. Convergence analysis of a proximal-like minimization algorithm using Bregman functions. *SIAM Journal on Optimization*, 3(3):538–543, August 1993.

A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *Advances in Neural Information Processing Systems 24*, 2011.

O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 713–720, 2011.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2873–2898, 2009.

J. Duchi, A. Agarwal, and M. Wainwright. Distributed dual averaging in networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 550–558, 2010.

S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization. Technical report, Department of Industrial and System Engineering, University of Florida, Gainesville, FL, 2010.

K. Gimpel, D. Das, and N. A. Smith. Distributed asynchronous online learning for natural language processing. In *Proceedings of the Fourth Conference on Computational Natural Language Learning*, pages 213–222, 2010.

R. Goebel and R. T. Rockafellar. Local strong convexity and local Lipschitz continuity of the gradient of convex functions. *Journal of Convex Analysis*, 15(2):263–270, 2008.

J. L. Gustafson. Reevaluating Amdahl's Law. *Communications of the ACM*, 31(5):532–533, 1988.

E. Harrington, R. Herbrich, J. Kivinen, J. Platt, and R. C. Williamson. Online Bayes point machines. In *Proceedings of the Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 241–252, 2003.

C. Hu, J. T. Kwok, and W. Pan. Accelerated gradient methods for stochastic optimization and online learning. In Y. Bengio, D. Schuurmans, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 781–789, 2009.

A. Juditsky, A. Nemirovski, and C. Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1:1–42, 2011.

G. Lan. An optimal method for stochastic composite optimization. Technical report, Georgia Institute of Technology, 2009.

G. Lan, Z. Lu, and R. D. C. Monteiro. Primal-dual first-order methods with $O(1/\varepsilon)$ iteration-complexity for cone programming. *Mathematical Programming*, 126:1–29, 2011.

J. Langford, A. J. Smola, and M. Zinkevich. Slow learners are fast. In *Advances in Neural Information Processing Systems (NIPS) 22*, pages 2331–2339, 2009.

A. Nedić and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

A. Nedić, D. P. Bertsekas, and V. S. Borkar. Distributed asynchronous incremental subgradient methods. In D. Butnariu, Y. Censor, and S. Reich, editors, *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, Studies in Computational Mathematics, pages 381–407. Elsevier, 2001.

A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Series in Discrete Mathematics. Wiley-Interscience, 1983.

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.

Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Boston, 2004.

Y. Nesterov. Smooth minimization of nonsmooth functions. *Mathematical Programming*, 103: 127–152, 2005.

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, August 2009.

Y. Nesterov and J.-Ph. Vial. Confidence level solutions for stochastic programming. *Automatica*, 44(6):1559–1568, 2008.

B. T. Polyak. *Introduction to Optimization*. Translations Series in Mathematics and Engineering. Optimization Software, Inc., New York, 1987.

R. T. Rockafellar and R. J-B Wets. On the interchange of subdifferentiation and conditional expectation for convex functionals. *Stochastics: An International Journal of Probability and Stochastic Processes*, 7(3):173–182, 1982.

S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1-regularized loss minimization. *Journal of Machine Learning Research*, 12:1865–1892, 2011.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 807–814, 2007.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. Submitted to *SIAM Journal on Optimization*, 2008.

J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

R. J-B Wets. Stochastic programming. In G. Nemhauser and A. Rinnnooy Kan, editors, *Handbook for Operations Research and Management Sciences*, volume 1, pages 573–629. Elsevier Science Publishers, Amsterdam, The Netherlands, 1989.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 928–936, Washington DC, 2003.

M. Zinkevich, M. Weimer, A. Smola, and L. Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2595–2603, 2010.