Distributed Stochastic Optimization via Adaptive SGD

Ashok Cutkosky
Stanford University, USA*
cutkosky@google.com

Róbert Busa-Fekete
Yahoo! Research, New York, USA
busafekete@oath.com

Abstract

Stochastic convex optimization algorithms are the most popular way to train machine learning models on large-scale data. Scaling up the training process of these models is crucial, but the most popular algorithm, Stochastic Gradient Descent (SGD), is a serial method that is surprisingly hard to parallelize. In this paper, we propose an efficient distributed stochastic optimization method by combining adaptivity with variance reduction techniques. Our analysis yields a linear speedup in the number of machines, constant memory footprint, and only a logarithmic number of communication rounds. Critically, our approach is a black-box reduction that parallelizes any serial online learning algorithm, streamlining prior analysis and allowing us to leverage the significant progress that has been made in designing adaptive algorithms. In particular, we achieve optimal convergence rates without any prior knowledge of smoothness parameters, yielding a more robust algorithm that reduces the need for hyperparameter tuning. We implement our algorithm in the Spark distributed framework and exhibit dramatic performance gains on large-scale logistic regression problems.

1 Setup

We consider a fundamental problem in machine learning, stochastic convex optimization:

$$\min_{w \in W} F(w) := \mathop{\mathbb{E}}_{f \sim \mathcal{D}}[f(w)] \tag{1}$$

Here, W is a convex subset of $\mathbb{R}^d$ and $\mathcal{D}$ is a distribution over L-smooth convex functions $W \to \mathbb{R}$. We do not have direct access to F, and the distribution $\mathcal{D}$ is unknown, but we do have the ability to generate i.i.d. samples $f \sim \mathcal{D}$ through some kind of stream or oracle. In practice, each function $f \sim \mathcal{D}$ corresponds to a new datapoint in some learning problem. Algorithms for this problem are widely applicable: for example, in logistic regression the goal is to optimize $F(w) = \mathbb{E}[f(w)] = \mathbb{E}[\log(1 + \exp(-yw^T x))]$ when the (x, y) pairs are the (feature vector, label) pairs coming from a fixed data distribution. Given a budget of N oracle calls (e.g. a dataset of size N), we wish to find a $\hat{w}$ such that $F(\hat{w}) - F(w_\star)$ (called the *suboptimality*) is as small as possible as fast as possible using as little memory as possible, where $w_\star \in \operatorname{argmin} F$.

The most popular algorithm for solving (1) is Stochastic Gradient Descent (SGD), which achieves statistically optimal $O(1/\sqrt{N})$ suboptimality in $O(N)$ time and constant memory. However, in modern large-scale machine learning problems the number of data points N is often gigantic, and so even the linear time-complexity of SGD becomes onerous. We need a *parallel* algorithm that runs in only $O(N/m)$ time using m machines. We address this problem in this paper, evaluating solutions on three metrics: time complexity, space complexity, and communication complexity. Time complexity is the total time taken to process the data points. Space complexity is the amount of space required per

---

*now at Google

machine. Note that in our streaming model, an algorithm that keeps only the most recently seen data point in memory is considered to run in constant memory. Communication complexity is measured in terms of the number of "rounds" of communication in which all the machines synchronize. In measuring these quantities we often suppress all constants other than those depending on N and m and all logarithmic factors.

In this paper we achieve the ideal parallelization complexity (up to a logarithmic factor) of $\tilde{O}(N/m)$ time, $O(1)$ space and $\tilde{O}(1)$ rounds of communication, so long as $m < \sqrt{N}$. Further, in contrast to much prior work, our algorithm is a *reduction* that enables us to generically parallelize any serial online learning algorithm that obtains a sufficiently adaptive convergence guarantee (e.g. [8, 14, 6]) in a black-box way. This significantly simplifies our analysis by decoupling the learning rates or other internal variables of the serial algorithm from the parallelization procedure. This technique allows our algorithm to adapt to an unknown smoothness parameter L in the problem, resulting in optimal convergence guarantees without requiring tuning of learning rates. This is an important aspect of the algorithm: even prior analyses that meet the same time, space and communication costs [9, 12, 13] require the user to input the smoothness parameter to tune a learning rate. Incorrect values for this parameter can result in failure to converge, not just slower convergence. In contrast, our algorithm automatically adapts to the true value of L with no tuning. Empirically, we find that the parallelized implementation of a serial algorithm matches the performance of the serial algorithm in terms of sample-complexity, while bestowing significant runtime savings.

2 Prior Work

One popular strategy for parallelized stochastic optimization is minibatch-SGD [7], in which one computes m gradients at a fixed point in parallel and then averages these gradients to produce a single SGD step. When m is not too large compared to the variance in $\mathcal{D}$, this procedure gives a linear speedup in theory and uses constant memory. Unfortunately, minibatch-SGD obtains a communication complexity that scales as $\sqrt{N}$ (or $N^{1/4}$ for accelerated variants). In modern problems when N is extremely large, this overhead is prohibitively large. We achieve a communication complexity that is logarithmic in N, allowing our algorithm to be run as a near-constant number of map-reduce jobs even for very large N. We summarize the state of the art for some prior algorithms algorithms in Table 1.

Many prior approaches to reducing communication complexity can be broadly categorized into those that rely on Newton's method and those that rely on the variance-reduction techniques introduced in the SVRG algorithm [11]. Algorithms that use Newton's method typically make the assumption that $\mathcal{D}$ is a distribution over quadratic losses [20, 22, 16, 21], and leverage the fact that the expected Hessian is constant to compute a Newton step in parallel. Although quadratic losses are an excellent starting point, it is not clear how to generalize these approaches to arbitrary non-quadratic smooth losses such as encountered in logistic regression.

Alternative strategies stemming from SVRG work by alternating between a "batch phase" in which one computes a very accurate gradient estimate using a large batch of examples and an "SGD phase" in which one runs SGD, using the batch gradient to reduce the variance in the updates [9, 12, 18, 10]. Our approach also follows this overall strategy (see Section 3 for a more detailed discussion of this procedure). However, all prior algorithms in this category make use of carefully specified learning rates in the SGD phase, while our approach makes use of *any* adaptive serial optimization algorithm, even ones that do not resemble SGD at all, such as [6, 14]. This results in a streamlined analysis and a more general final algorithm. Not only do we recover prior results, we can leverage the adaptivity of our base algorithm to obtain better results on sparse losses and to avoid any dependencies on the smoothness parameter L, resulting in a much more robust procedure.

The rest of this paper is organized as follows. In Section 3 we provide a high-level overview of our strategy. In Section 4 we introduce some basic facts about the analysis of adaptive algorithms using online learning, in Section 5 we sketch our intuition for combining SVRG and the online learning analysis, and in Section 6 we describe and analyze our algorithm. In Section 7 we show that the convergence rate is statistically optimal and show that a parallelized implementation achieves the stated complexities. Finally in Section 9 we give some experimental results.

Table 1: Comparison of distributed optimization algorithms with a dataset of size N and m machines. Logarithmic factors and all constants not depending on N or m have been dropped.

Method	Quadratic Loss	Space	Communication	Adapts to L
Newton inspired [20, 22, 16, 21]	Needed	N/m	1	No
accel. minibatch-SGD [5]	Not Needed	1	$N^{1/4}$	No
prior SVRG-like [9, 12, 18, 10]	Not Needed	1	1	No
This work	Not Needed	1	1	Yes

3 Overview of Approach

Our overall strategy for parallelizing a serial SGD algorithm is based upon the stochastic variance-reduced gradient (SVRG) algorithm [11]. SVRG is a technique for improving the sample complexity of SGD given access to a stream of i.i.d. samples $f \sim \mathcal{D}$ (as in our setting), as well as the ability to compute *exact* gradients $\nabla F(v)$ in a potentially expensive operation. The basic intuition is to use an exact gradient $\nabla F(v)$ at some "anchor point" $v \in W$ as a kind of "hint" for what the exact gradient is at nearby points w. Specifically, SVRG leverages the theorem that $\nabla f(w) - \nabla f(v) + \nabla F(v)$ is an unbiased estimate of $\nabla F(w)$ with variance approximately bounded by $L(F(v) - F(w_\star))$ (see (8) in [11]). Using this fact, the SVRG strategy is:

1. Choose an "anchor point" $v = w_0$.

2. Compute an exact gradient $\nabla F(v)$ (this is an expensive operation).

3. Perform T SGD updates: $w_{t+1} = w_t - \eta(\nabla f(w_t) - \nabla f(v) + \nabla F(v))$ for T i.i.d. samples $f \sim \mathcal{D}$ using the fixed anchor v.

4. Choose a new anchor point v by averaging the T SGD iterates, set $w_0 = v$ and repeat 2-4.

By reducing the suboptimality of the anchor point v, the variance in the gradients also decreases, producing a virtuous cycle in which optimization progress reduces noise, which allows faster optimization progress. This approach has two drawbacks that we will address. First, it requires computing the exact gradient $\nabla F(v)$, which is impossible in our stochastic optimization setup. Second, prior analyses require specific settings for η that incorporate L and fail to converge with incorrect settings, requiring the user to manually tune η to obtain the desired performance. To deal with the first issue, we can approximate $\nabla F(v)$ by averaging gradients over a mini-batch, which allows us to approximate SVRG's variance-reduced gradient estimate, similar to [9, 12]. This requires us to keep track of the errors introduced by this approximation. To deal with the second issue, we incorporate analysis techniques from online learning which allow us replace the constant step-size SGD with any adaptive stochastic optimization algorithm in a black-box manner. This second step forms the core of our theoretical contribution, as it both simplifies analysis and allows to adapt to L.

The overall roadmap for our analysis has five steps:

1. We model the errors introduced by approximating the anchor-point gradient $\nabla F(v)$ by a minibatch-average as a "bias", so that we think of our algorithm as operating on slightly biased but low-variance gradient estimates.

2. Focusing first only the bias aspect, we analyze the performance of online learning algorithms with biased gradient estimates and show that so long as the bias is sufficiently small, the algorithm will still converge quickly (Section 4).

3. Next focusing on the variance-reduction aspect, we show that any online learning algorithm which enjoys a sufficiently adaptive convergence guarantee produces a similar "virtuous cycle" as observed with constant-step-size SGD in the analysis of SVRG, resulting in fast convergence (sketched in Section 5, proved in Appendices C and D).

4. Combine the previous three steps to show that applying SVRG using these approximate variance-reduced gradients and an adaptive serial SGD algorithm achieves $O(L/\sqrt{N})$ suboptimality using only $O(\sqrt{N})$ serial SGD updates (Sections 6 and 7).

5. Observe that the batch processing in step 3 can be done in parallel, that this step consumes the vast majority of the computation, and that it only needs to be repeated logarithmically many times (Section 7).

4   Biased Online Learning

A popular way to analyze stochastic gradient descent and related algorithms is through online learning [19]. In this framework, an algorithm repeatedly outputs vectors w_t for $t = 1, 2, \ldots$ in some convex space W, and receives gradients g_t such that $\mathbb{E}[g_t] = \nabla F(w_t)$ for some convex objective function F.[2] Typically one attempts to bound the linearized *regret*:

$$R_T(w_\star) = \sum_{t=1}^{T} g_t \cdot (w_t - w_\star)$$

Where $w_\star = \operatorname{argmin} F$. We can apply online learning algorithms to stochastic optimization via online-to-batch conversion [3], which tells us that

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \tfrac{\mathbb{E}[R_T(w_\star)]}{T}$$

where $\overline{w} = \frac{1}{T} \sum_{t=1}^{T} w_t$.

Thus, an algorithm that guarantees small regret immediately guarantees convergence in stochastic optimization. Online learning algorithms typically obtain some sort of (deterministic!) guarantee like

$$R_T(w_\star) \leq R(w_\star, \|g_1\|, \ldots, \|g_T\|)$$

where R is increasing in each $\|g_t\|$. For example, when the convex space W has diameter D, AdaGrad [8] obtains $R_T(w_\star) \leq D\sqrt{2\sum_{t=1}^{T} \|g_t\|^2}$.

As foreshadowed in Section 3, we will need to consider the case of *biased gradients*. That is, $\mathbb{E}[g_t] = \nabla F(w_t) + b_t$ for some unknown bias vector b_t. Given these biased gradients, a natural question is: to what extent does controlling the regret $R_T(w_\star) = \sum_{t=1}^{T} g_t \cdot (w_t - w_\star)$ affect our ability to control the suboptimality $\mathbb{E}[\sum_{t=1}^{T} F(w_t) - F(w_\star)]$? We answer this question with the following simple result:

Proposition 1. *Define* $R_T(w_\star) = \sum_{t=1}^{T} g_t \cdot (w_t - w_\star)$ *and* $\overline{w} = \frac{1}{T} \sum_{t=1}^{T} w_t$ *where* $\mathbb{E}[g_t] = \nabla F(w_t) + b_t$. *Then*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \tfrac{\mathbb{E}[R_T(w_\star)]}{T} + \tfrac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]$$

If the domain V has diameter D, then $\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \tfrac{\mathbb{E}[R_T(w_\star)]}{T} + \tfrac{D}{T} \sum_{t=1}^{T} \mathbb{E}[\|b_t\|]$

Our main convergence results will require algorithms with regret bounds of the form $R(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T} \|g_t\|^2}$ or $R(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T} \|g_t\|}$ for various ψ. This is an acceptable restriction because there are many examples of such algorithms, including AdaGrad [8], SOLO [15], PiSTOL [14] and FreeRex [6]. Further, in Proposition 3 we show a simple trick to remove the dependence on $\|w_t - w_\star\|$, allowing our results to extend to unbounded domains.

5   Variance-Reduced Online Learning

In this section we sketch an argument that using variance reduction in conjunction with a online learning algorithm guaranteeing regret $R(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T} \|g_t\|^2}$ results in a very fast convergence of $\sum_{t=1}^{T} \mathbb{E}[F(w_t) - F(w_\star)] = O(1)$ up to log factors. A similar result holds for regret guarantees like $R(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T} \|g_t\|}$ via a similar argument, which we leave to Appendix D. To do

---

[2]The online learning literature often allows for adversarially generated g_t, but we consider only stochastically generated g_t here.

this we make use of a critical lemma of variance reduction which asserts that a variance-reduced gradient estimate g_t of $\nabla F(w_t)$ with anchor point v_t has $\mathbb{E}[\|g_t\|^2] \leq L(F(w_t) + F(v_t) - 2F(w_\star))$ up to constants. This gives us the following informal result:

Proposition 2. *[Informal statement of Proposition 8] Given a point $w_t \in W$, let g_t be an unbiased estimate of $\nabla F(w_t)$ such that $\mathbb{E}[\|g_t\|^2] \leq L(F(w_t) + F(v_t) - 2F(w_\star))$. Suppose $w_1, \ldots, w_T$ are generated by an online learning algorithm with regret at most $R(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^T \|g_t\|^2}$. Then*

$$\mathbb{E}\left[\sum_{t=1}^T F(w_t) - F(w_\star)\right] = O\left(L\psi(w_\star)^2 + \psi(w_\star)\sqrt{\sum_{t=1}^T L\,\mathbb{E}[F(v_t) - F(w_\star)]}\right) \qquad (2)$$

Proof. The proof is remarkably simple, and we sketch it in one line here. The full statement and proof can be found in Appendix D.

$$\mathbb{E}\left[\sum_{t=1}^T F(w_t) - F(w_\star)\right] \leq \psi(w_\star)\,\mathbb{E}\left[\sqrt{\sum_{t=1}^T \|g_t\|^2}\right]$$

$$\leq \psi(w_\star)\sqrt{\sum_{t=1}^T L\,\mathbb{E}[F(w_t) - F(w_\star) + F(v_t) - F(w_\star)]}$$

Now square both sides and use the quadratic formula to solve for $\mathbb{E}\left[\sum_{t=1}^T F(w_t) - F(w_\star)\right]$. $\qquad \square$

Notice that in Proposition 2, the online learning algorithm's regret guarantee $\psi(w_\star)\sqrt{\sum_{t=1}^T \|g_t\|^2}$ does not involve the smoothness parameter L, and yet nevertheless L shows up in equation (2). It is this property that will allow us to adapt to L without requiring any user-supplied information.

Algorithm 1 SVRG OL (**SVRG** with **O**nline **L**earning)

1: **Initialize:** Online learning algorithm $\mathcal{A}$; Batch size $\hat{N}$; epoch lengths $0 = T_0, \ldots, T_K$; Set $T_{a:b} = \sum_{i=a}^b T_i$.
2: Get initial vector w_1 from $\mathcal{A}$, set $v_t \leftarrow w_1$.
3: **for** $k = 1$ **to** K **do**
4:     Sample $\hat{N}$ functions $f_1, \ldots, f_{\hat{N}} \sim \mathcal{D}$
5:     $\nabla\hat{F}(v_k) \leftarrow \frac{1}{\hat{N}}\sum_{i=1}^{\hat{N}} \nabla f_i(v_k)$
6:     (this step can be done in parallel).
7:     **for** $t = T_{0:k-1} + 1$ **to** $T_{0:k}$ **do**
8:         Sample $f \sim \mathcal{D}$.
9:         Give $g_t = \nabla f(w_t) - \nabla f(v_k) + \nabla\hat{F}(v_k)$ to $\mathcal{A}$.
10:         Get updated vector w_{t+1} from $\mathcal{A}$.
11:     **end for**
12:     $v_{k+1} \leftarrow \frac{1}{T_k}\sum_{t=T_{0:k-1}+1}^{T_{0:k}} w_t$.
13: **end for**

Variance reduction allows us to generate estimates g_t satisfying the hypothesis of Proposition 2, so that we can control our convergence rate by picking appropriate v_ts. We want to change v_t very few times because changing anchor points requires us to compute a high-accuracy estimate of $\nabla F(v_t)$. Thus we change v_t only when t is a power of 2 and set v_{2^n} to be the average of the last 2^{n-1} iterates w_t. By Jensen, this allows us to bound $\sum_{t=1}^T \mathbb{E}[F(v_t) - F(w_\star)]$ by $\sum_{t=1}^T \mathbb{E}[F(w_t) - F(w_\star)]$, and so applying Proposition 2 we can conclude $\sum_{t=1}^T \mathbb{E}[F(w_t) - F(w_\star)] = O(1)$.

6 SVRG with Online Learning

With the machinery of the previous sections, we are now in a position to derive and analyze our main algorithm, presented in SVRG OL.

SVRG OL implements the procedure described in Section 3. For each of a series of K rounds, we compute a batch gradient estimate $\nabla \hat{F}(v_k)$ for some "anchor point" v_k. Then we run T_k iterations of an online learning algorithm. To compute the tth gradient g_t given to the online learning algorithm in response to an output point w_t, SVRG OL approximates the variance-reduction trick of SVRG, setting $g_t = \nabla f(w_t) - \nabla f(v_k) + \nabla \hat{F}(v_k)$ for some new sample $f \sim \mathcal{D}$. After the T_k iterations have elapsed, a new anchor point v_{k+1} is chosen and the process repeats.

In this section we characterize SVRG OL's performance when the base algorithm $\mathcal{A}$ has a regret guarantee of $\psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|^2}$. We can also perform essentially similar analysis for regret guarantees like $\psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|}$, but we postpone this to Appendix E.

In order to analyze SVRG OL, we need to bound the error $\|\nabla \hat{F}(v_k) - \nabla F(v_k)\|$ uniformly for all $k \le K$. This can be accomplished through an application of Hoeffding's inequality:

Lemma 1. *Suppose that $\mathcal{D}$ is a distribution over G-Lipschitz functions. Then with probability at least $1 - \delta$, $\max_k \|\nabla \hat{F}(v_k) - \nabla F(v_k)\| \le \sqrt{\frac{2G^2 \log(K/\delta) + G^2}{\hat{N}}}$.*

The proof of Lemma 1 is deferred to Appendix A. The following Theorem is now an immediate consequences of the concentration bound Lemma 1 and Propositions 8 and 9 (see Appendix).

Theorem 1. *Suppose the online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \le \psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|^2}$. Set $b_t = \|\nabla \hat{F}(v_k) - \nabla F(v_k)\|$ for $t \in [T_{0:k-1} + 1, T_{1:k}]$ (where $T_{a:b} := \sum_{i=a}^{b} T_i$). Suppose that $T_k/T_{k-1} \le \rho$ for all k. Then for $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$,*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \le \frac{32(1 + \rho)\psi(w_\star)^2 L}{T} + \frac{2\sum_{t=1}^{T} \mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]}{T}$$

$$+ \frac{2\psi(w_\star)\sqrt{8LT_1 \mathbb{E}[F(v_1) - F(w_\star)] + 2\sum_{t=1}^{T} \mathbb{E}[\|b_t\|^2]}}{T}$$

In particular, if $\mathcal{D}$ is a distribution over G-Lipschitz functions, then with probability at least $1 - \frac{1}{T}$ we have $\|b_t\| \le \sqrt{\frac{2G^2 \log(KT) + G^2}{\hat{N}}}$ for all t. Further, if $\hat{N} > T^2$ and V has diameter D, then this implies

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \le \frac{32(1 + \rho)\psi(w_\star)^2 L}{T} + \frac{4\psi(w_\star)\sigma\sqrt{\log(KT)}}{T\sqrt{T}}$$

$$+ \frac{8\psi(w_\star)\sqrt{LT_1 \mathbb{E}[F(v_1) - F(w_\star)]}}{T} + \frac{GD}{T} + 2\frac{G\sqrt{2\log(KT) + 1}D}{T}$$

$$= O\left(\frac{\sqrt{\log(KT)}}{T}\right)$$

We note that although this theorem requires a finite diameter for the second result, we present a simple technique to deal with unbounded domains and retain the same result in Appendix D

7 Statistical and Computational Complexity

In this section we describe how to choose the batch size $\hat{N}$ and epoch sizes T_k in order to obtain optimal statistical complexity and computational complexity. The total amount of data used by SVRG OL is $N = K\hat{N} + T_{0:K} = K\hat{N} + T$. If we choose $\hat{N} = T^2$, this is $O(K\hat{N})$. Set $T_k = 2T_{k-1}$, with some $T_1 > 0$ so that $\rho = \max T_k/T_{k-1} = 2$ and $O(K = \log(N))$. Then our Theorem 1 guarantees suboptimality $O(\sqrt{\log(TK)}/T)$, which is $O(\sqrt{K \log(TK)}/\sqrt{K\hat{N}}) = O(\sqrt{K \log(TK)}/\sqrt{N})$. This matches the optimal $O(1/\sqrt{N})$ up to logarithmic factors and constants.

The parallelization step is simple: we parallelize the computation of $\nabla \hat{F}(v_k)$ by having m machines compute and sum gradients for $\hat{N}/m$ new examples each, and then averaging these m sums together

on one machine. Notice that this can be done with constant memory footprint by streaming the examples in - the algorithm will not make any further use of these examples so it's safe to forget them. Then we run the T_k steps of the inner loop in serial, which again can be done in constant memory footprint. This results in a total runtime of $O(K\hat{N}/m + T)$ - a linear speedup so long as $m < KN/T$. For algorithms with regret bounds matching the conditions of Theorem 1, we get optimal convergence rates by setting $\hat{N} = T^2$, in which case our total data usage is $N = O(K\hat{N})$. This yields the following calculation:

Theorem 2. *Set $T_k = 2T_{k-1}$. Suppose the base optimizer $\mathcal{A}$ in SVRG OL guarantees regret $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T} \|g_t\|^2}$, and the domain W has finite diameter D. Let $\hat{N} = \Theta(T^2)$ and $N = K\hat{N} + T$ be the total number of data points observed. Suppose we compute the batch gradients $\nabla\hat{F}(v_k)$ in parallel on m machines with $m < \sqrt{N}$. Then for $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$ we obtain*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] = \tilde{O}\left(\frac{1}{\sqrt{N}}\right)$$

in time $\tilde{O}(N/m)$, and space $O(1)$, and $K = \tilde{O}(1)$ communication rounds.

8 Implementation

8.1 Linear Losses and Dense Batch Gradients

Many losses of practical interest take the form $f(w) = \ell(w \cdot x, y)$ for some label y and feature vector $x \in \mathbb{R}^d$ where d is extremely large, but x is s-sparse. These losses have the property that $\nabla f(w) = \ell'(w \cdot x, y)x$ is also s-sparse. Since d can often be very large, it is extremely desirable to perform all parameter updates in $O(s)$ time rather than $O(d)$ time. This is relatively easy to accomplish for most SGD algorithms, but our strategy involves correcting the variance in $\nabla f(w)$ using a *dense* batch gradient $\nabla\hat{F}(v_k)$ and so we are in danger of losing the significant computational speedup that comes from taking advantage of sparsity. We address this problem through an importance-sampling scheme.

Suppose the ith coordinate of x is non-zero with probability p_i. Given a vector v, let $I(v)$ be the vector whose ith component is 0 if $w_i = 0$, or $1/p_i$ is $w_i \neq 0$. Then $\mathbb{E}[I(\nabla f(w))]$ is equal to the all-ones vector. Using this notation, we replace the variance-reduced gradient estimate $\nabla f(w) - \nabla f(v_k) + \nabla\hat{F}(v_k)$ with $\nabla f(w) - \nabla f(v_k) + \nabla\hat{F}(v_k) \odot I(\nabla f(w))$, where $\odot$ indicates component-wise multiplication. Since $\mathbb{E}[I(\nabla f(w))]$ is the all-ones vector, $\mathbb{E}[\nabla\hat{F}(v_k) \odot I(\nabla f(w))] = \nabla\hat{F}(v_k)$ and so the expected value of this estimate has not changed. However, it is clear that the *sparsity* of the estimate is now equal to the sparsity of $\nabla f(w)$. Performing this transformation introduces additional variance into the estimate, and could slow down our convergence by a constant factor. However, we find that even with this extra variance we still see impressive speedups (see Section 9).

8.2 Spark implementation

Implementing our algorithm in the Spark environment is fairly straightforward. SVRG OL switches between two phases: a batch gradient computation phase and a serial phase in whicn we run the online learning algorithm. The serial phase is carried out by the driver while the batch gradient is computed by executors. We initially divide the training data into C approximately 100M chunks, and we use $\min(1000, C)$ executors. Tree aggregation with depth of 5 is used to gather the gradient from the executors, which is similar to the operation implemented by Vowpal Wabbing (VW) [1]. We use asynchronous collects to move the instances used in the next serial SGD phase of SVRG OL to the driver while the batch gradient is being computed. We used feature hashing with 23 bits to limit memory consumption.

8.3 Batch sizes

Our theoretical analysis asks for exponentially increasing serial phase lengths T_k and a batch size of of $\hat{N} = T^2$. In practice we use slightly different settings. We have a constant serial phase length $T_k = T_0$ for all k, and an increasing batch size $\hat{N}_k = kC$ for some constant C. We usually set

$C = T_0$. The constant T_k is motivated by observing that the requirement for exponentially increasing T_k comes from a desire to offset potential poor performance in the first serial phase (which gives the dependence on T_1 in Theorem 1). In practice we do not expect this to be an issue. The increasing batch size is motivated by the empirical observation that earlier serial phases (when we are farther from the optimum) typically do not require as accurate a batch gradient in order to make fast progress.

Table 2: Statistics of the datasets. The compressed size of the data is reported. B=Billion, M=Million

Data	# Instance		Data size (Gb)		# Features	Avg # feat	% positives
–	Train	Test	Train	Test			
KDD10	19.2M	0.74M	0.5	0.02	29 890 095	29.34	86.06%
KDD12	119.7M	29.9M	1.6	0.5	54 686 452	11.0	4.44%
ADS SMALL	1.216B	0.356B	155.0	40.3	2 970 211	92.96	8.55%
ADS LARGE	5.613B	1.097B	1049.1	486.1	12 133 899	95.72	9.42%
EMAIL	1.236B	0.994B	637.4	57.6	37 091 273	132.12	18.74%

9 Experiments

To verify our theoretical results, we carried out experiments on large-scale (order 100 million datapoints) public datasets, such as KDD10 and KDD12 [3] and on proprietary data (order 1 billion datapoints), such as click-prediction for ads [4] and email click-prediction datasets [2]. The main statistics of the datasets are shown in Table 2. All of these are large datasets with sparse features, and heavily imbalanced in terms of class distribution. We solved these binary classification tasks with logistic regression. We tested two well-know scalable logistic regression implementation: Spark ML 2.2.0 and Vowpal Wabbit 7.10.0 (VW) [4]. To optimize the logistic loss we used the L-BFGS algorithm implemented by both packages. We also tested minibatch SGD and non-adaptive SVRG implementations. However, we observe that the relationship between non-adaptive SVRG updates and the updates in our algorithm are analogous to the relationship between the updates in constant-step-size SGD and (for example) AdaGrad. Since our experiments involved sparse high-dimensional data, adaptive step sizes are very important and one should not expect these algorithms to be competitive (and indeed they were not).
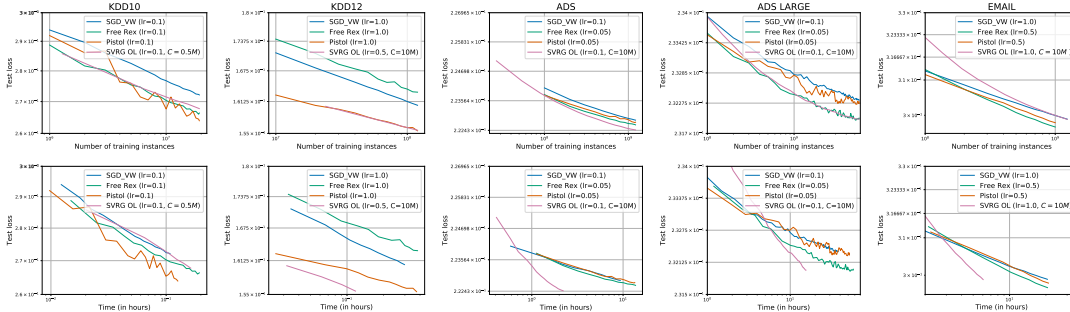

Figure 1: Test loss of three SGD algorithms (PiSTOL [14], Vowpal Wabbit (labeled as SGD VW) [17] and FreeRex [6]) and SVRG OL (labeled as SVRG OL, using FreeRex as the base optimizer) on the benchmark datasets.

First we compared SVRG OL to several non-parallelized baseline SGD optimizers on the different datasets. We plot the loss a function of the number of datapoints processed, as well as the total runtime (Figure 1). Measuring the number of datapoints processed gives us a sense of the statistical efficiency of the algorithm and gives a metric that is independent of implementation quality details. We see that, remarkably, SVRG OL's actually performs well as a function of number of datapoints

---

processed and so is a competitive *serial* algorithm before even any parallelization. Thus it is no surprise that we see significant speedups when the batch computation is parallelized.

To assess the trend of the speed-up with the size of the training data, we plotted the relative speed-up of SVRG OL versus FreeRex which is used as base optimizer in SVRG OL. Figure 2 shows the fraction of running time of non-parallel and parallel algorithms needed to achieve the same performance in terms of test loss. The x-axis scales with the running time of the parallel SVRG OL algorithm. The speed-up increases with training time, and thus the number of training instances processed. This result suggests that our method will indeed match with the theoretical guarantees in case of large enough datasets, although this trend is hard to verify rigorously in our test regime.
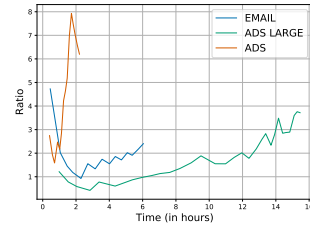

Figure 2: The speed-up ratio of SVRG OL versus FreeRex on various datasets.

In our second experiment, we proceed to compare SVRG OL to Spark ML and VW in Table 4. These two LBFGS-based algorithms were superior in all metrics to minibatch SGD and non-adaptive SVRG algorithms and so we report only the comparison to Spark ML and VW (see Section F for full results). We measure the number of communication rounds, the total training error, the error on a held-out test set, the Area Under the Curve (AUC), and total runtime in minutes. Table 4 illustrates that SVRG OL compares well to both Spark ML and VW. Notably, SVRG OL uses dramatically fewer communication rounds. On the smaller KDD datasets, we also see much faster runtimes, possibly due to overhead costs associated with the other algorithms. It is important to note that our SVRG OL makes only one pass over the dataset, while the competition makes one pass per communication round, resulting in 100s of passes. Nevertheless, we obtain competitive final error due to SVRG OL's high statistical efficiency.

Table 3: Average loss and AUC achieved by Logistic regression implemented in Spark ML, VW and SVRG OL. "Com." refers to number of communication rounds and time is measured in minutes. The results on KDD10, ADS LARGE and EMAIL data is presented in App. F due to lack of space.

Dataset	KDD12					ADS SMALL				
	Com.	Train	Test	AUC	Time	Com.	Train	Test	AUC	Time
Spark ML	100	0.15756	0.15589	75.485	36	100	0.23372	0.22288	83.356	42
Spark ML	550	0.15755	0.15570	75.453	180	500	0.23365	0.22286	83.365	245
VW	100	0.15398	0.15725	77.871	44	100	0.23381	0.22347	83.214	114
VW	500	**0.14866**	0.15550	**78.881**	150	500	0.23157	0.22251	**83.499**	396
SVRG OL	4	0.152740	**0.154985**	78.431	8	14	**0.23147**	**0.22244**	83.479	94

10 Conclusion

We have presented SVRG OL, a generic stochastic optimization framework which combines adaptive online learning algorithms with variance reduction to obtain communication efficiency in parallel architectures. Our analysis significantly streamlines previous work by making black-box use of any adaptive online learning algorithm, thus disentangling the variance-reduction and serial phases of SVRG algorithms. We require only a logarithmic number of communication rounds, and we automatically adapt to an unknown smoothness parameter, yielding both fast performance and robustness to hyperparameter tuning. We developed a Spark implementation of SVRG OL and solved real large scale sparse learning problems with competitive performance to L-BFGS implemented by VW and Spark ML.

References

[1] Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *The Journal of Machine Learning Research*, 15(1):1111–1133, 2014.

[2] Alina Beygelzimer, Robert Busa-Fekete, Guy Halawi, and Francesco Orabona. Algorithmic notifications for mail: Debiasing mail data to predict user actions. In *under review*, 2018.

[3] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

[4] Haibin Cheng and Erick Cantú-Paz. Personalized click prediction in sponsored search. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 351–360, 2010. doi: 10.1145/1718487.1718531. URL http://doi.acm.org/10.1145/1718487.1718531.

[5] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 1647–1655, 2011.

[6] Ashok Cutkosky and Kwabena Boahen. Online learning without prior information. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 643–677, Amsterdam, Netherlands, 07–10 Jul 2017. PMLR. URL http://proceedings.mlr.press/v65/cutkosky17a.html.

[7] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202, 2012.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[9] Roy Frostig, Rong Ge, Sham M. Kakade, and Aaron Sidford. Competing with the empirical risk minimizer in a single pass. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 728–763, 2015.

[10] Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečný, and Scott Sallinen. Stopwasting my gradients: Practical svrg. In *Advances in Neural Information Processing Systems*, pages 2251–2259, 2015.

[11] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 315–323, 2013.

[12] Lihua Lei and Michael I Jordan. Less than a single pass: Stochastically controlled stochastic gradient method. *arXiv preprint arXiv:1609.03261*, 2016.

[13] Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, pages 2345–2355, 2017.

[14] Francesco Orabona. Simultaneous model selection and optimization through parameter-free stochastic learning. In *Advances in Neural Information Processing Systems*, pages 1116–1124, 2014.

[15] Francesco Orabona and Dávid Pál. Scale-free online learning. *arXiv preprint arXiv:1601.01974*, 2016.

[16] Sashank J Reddi, Jakub Konečný, Peter Richtárik, Barnabás Póczós, and Alex Smola. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

[17] Stéphane Ross, Paul Mineiro, and John Langford. Normalized online learning. *arXiv preprint arXiv:1305.6646*, 2013.

[18] Vatsal Shah, Megasthenis Asteris, Anastasios Kyrillidis, and Sujay Sanghavi. Trading-off variance and complexity in stochastic gradient descent. *arXiv preprint arXiv:1603.06861*, 2016.

[19] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[20] Ohad Shamir, Nathan Srebro, and Tong Zhang. Communication efficient distributed optimization using an approximate newton-type method. *CoRR*, abs/1312.7853, 2013. URL http://arxiv.org/abs/1312.7853.

[21] Jialei Wang, Weiran Wang, and Nathan Srebro. Memory and communication efficient distributed stochastic optimization with minibatch prox. *CoRR*, abs/1702.06269, 2017. URL http://arxiv.org/abs/1702.06269.

[22] Yuchen Zhang and Lin Xiao. Communication-efficient distributed optimization of self-concordant empirical loss. *CoRR*, abs/1501.00263, 2015. URL http://arxiv.org/abs/1501.00263.

Supplementary material for "Distributed Stochastic Optimization via Adaptive SGD"

The appendix starts with the proof of Lemma 1 in Appendix A. Next, in Section B we provide the proofs from Section 4. In Section C we review prior results about the properties of smooth convex functions as variance reduction. In Section D we then combine the previous two sections' results to prove the convergence of our SVRG OL. Finally, in Section F we provide additional information about our experiments, including statistics of the various datasets as well as a complete reporting of the performance of all tested algorithms.

A   Proof of Lemma 1

Proof. The assumption on $\mathcal{D}$ implies that $\|\nabla f(v_k)\|$ is bounded by G and so $\nabla f(v_k)$ is G^2-subgaussian. Therefore we can apply the Hoeffding and union bounds to obtain tail bounds on $\nabla f(v_k)$:

$$\text{Prob}\left(\left\|\frac{1}{\hat{N}}\sum_{i=1}^{\hat{N}}\nabla f_i(v_k) - \mathbb{E}[\nabla f(v_k)]\right\| \geq \epsilon \text{ for all } k\right)$$

$$\leq K\exp\left[-\left(\hat{N}\epsilon - G\sqrt{\hat{N}}\right)^2/(2\hat{N}G^2)\right]$$

rearranging, with probability at least $1-\delta$, for all k we have

$$\left\|\frac{1}{\hat{N}}\sum_{i=1}^{N}\nabla f_i(v_k) - \mathbb{E}[\nabla f(v_k)]\right\| \leq \sqrt{\frac{2G^2\log(K/\delta) + G^2}{\hat{N}}}$$

as desired. $\qquad\square$

B   Proofs from Section 4

Proposition 1. *Define* $R_T(w_\star) = \sum_{t=1}^{T} g_t \cdot (w_t - w_\star)$ *and* $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$ *where* $\mathbb{E}[g_t] = \nabla F(w_t) + b_t$. *Then*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{\mathbb{E}[R_T(w_\star)]}{T} + \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]$$

If the domain V has diameter D, then $\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{\mathbb{E}[R_T(w_\star)]}{T} + \frac{D}{T}\sum_{t=1}^{T}\mathbb{E}[\|b_t\|]$

Proof. The proof follows from Cauchy-Schwarz, triangle inequality, and convexity of F:

$$\mathbb{E}[R_T] = \mathbb{E}\left[\sum_{t=1}^{T}g_t(w_t - w_\star)\right]$$

$$= \mathbb{E}\left[\sum_{t=1}^{T}\nabla F(w_t)\cdot(w_t - w_\star) + b_t\cdot(w_t - w_\star)\right]$$

$$\geq \mathbb{E}\left[\sum_{t=1}^{T}F(w_t) - F(w_\star) - \|b_t\|\|w_t - w_\star\|\right]$$

Now rearrange and apply Jensen's inequality to recover the first line of the Proposition. The second statement follows from observing that $\|w_t - w_\star\| \leq D$. $\qquad\square$

Proposition 1 shows that the suboptimality increases when both $\|b_t\|$ and $\|w_t\|$ becomes large. Although the online learning algorithm does not have the ability to control $\|b_t\|$, it does have the ability to control $\|w_t\|$, and so we can design a reduction to compensate for $\|b_t\|$. The reduction is simple: instead of g_t, provide the algorithm with $g_t + B\frac{w_t}{\|w_t\|}$, where B is a bound such that $B \geq \|b_t\|$ for all t, and by abuse of notation we take $w_t/\|w_t\| = 0$ when $w_t = 0$. Proposition 3 below, tells us that, so long as we know the bound B, we can obtain an increase in suboptimality that depends only on B and not w_t.

Proposition 3. *Suppose an online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \leq R(w_\star, \|g_1\|, \ldots, \|g_T\|)$, where R is an increasing function of each $\|g_t\|$. Then if we run $\mathcal{A}$ on gradients $g_t + B\frac{w_t}{\|w_t\|}$, we obtain:*

$$\mathbb{E}\left[\sum_{t=1}^{T}F(w_t) - F(w_\star)\right] \leq R(w_\star, \|g_1\| + B, \ldots, \|g_T\| + B) + 2TB$$

and, thus $\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{1}{T}R(w_\star, \|g_1\| + B, \ldots, \|g_T\| + B) + 2B$.

Proof. Observe that $B\frac{w_t}{\|w_t\|} = \nabla h(w_t)$ where $h(x) = B\|x\|$, so that by convexity we have

$$\mathbb{E}\left[\sum_{t=1}^{T} F(w_t) - F(w_\star) + b_t(w_t - w_\star) + B\|w_t\| - B\|w_\star\|\right]$$

$$\leq R\left(w_\star, \left\|g_1 + B\frac{w_t}{\|w_t\|}\right\|, \cdots\right)$$

$$\leq R(w_\star, \|g_1\| + B, \dots, \|g_T\| + B)$$

Now observe that $b_t(w_t - w_\star) + B\|w_t\| - B\|w_\star\| \geq -2B\|w_\star\|$ to obtain:

$$\mathbb{E}\left[\sum_{t=1}^{T} F(w_t) - F(w_\star)\right] - 2TB\|w_\star\|$$

$$\leq R(w_\star, \|g_1\| + B, \dots, \|g_T\| + B)$$

$$\mathbb{E}\left[\sum_{t=1}^{T} F(w_t) - F(w_\star)\right]$$

$$\leq R(w_\star, \|g_1\| + B, \dots, \|g_T\| + B) + 2TB$$

Finally, use Jensen's inequality to concude the Proposition $\qquad\square$

C Smooth Losses

In the following sections we consider applying an online learning algorithm to gradients g_t of the form $g_t = \nabla f_t(w_t) - \nabla f_t(\hat{x}) + \nabla F(\hat{x}) + b_t$ where each f_t is an i.i.d. smooth convex function with $\mathbb{E}[f_t] = F$ and $\hat{x}$ is some fixed vector. In order to leverage the structure of this kind of g_t, we'll need two lemmas from the literature about smooth convex functions (which can be found, for example, in [11]):

Lemma 2. *If f is an L-smooth convex function and x, y are fixed vectors, then*

$$\|\nabla f(x) - \nabla f(y)\|^2 \leq 2L(f(x) - f(y) - \nabla f(y)(x - y))$$

Proof. Set $\tilde{f}(w) = f(w) - f(y) - \nabla f(y) \cdot (w - y)$. Then $\tilde{f}$ is still convex and L-smooth and $\nabla \tilde{f}(y) = 0$ so that $\tilde{f}(y) \leq \tilde{f}(q)$ for all q. Therefore We have

$$0 = \tilde{f}(y) \leq \inf_z \tilde{f}(x - z\nabla\tilde{f}(x))$$

$$\leq \inf_z \tilde{f}(x) - z\|\nabla\tilde{f}(x)\|^2 + \frac{L}{2}z^2\|\nabla\tilde{f}(x)\|^2$$

$$= \tilde{f}(x) - \frac{1}{2L}\|\nabla\tilde{f}(x)\|^2$$

$$= f(x) - f(y) - \nabla f(y) \cdot (x - y) - \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|^2$$

from which the Lemma follows. $\qquad\square$

We can use Lemma 2 to show the following useful fact:

Lemma 3. *Suppose D is a distribution over L-smooth convex functions f and $F = \mathbb{E}[f]$. Let $w_\star \in \operatorname{argmin} F$. Then for all x we have $\mathbb{E}[\|\nabla f(x) - \nabla f(w_\star)\|^2] \leq 2L[F(x) - F(w_\star)]$.*

Proof. From Lemma 2 we have

$$\mathbb{E}[\|\nabla f(x) - \nabla f(w_\star)\|^2] \leq 2L\,\mathbb{E}[f(x) - f(w_\star) - \nabla f(w_\star) \cdot (x - w_\star)]$$

and now the result follows since $E[\nabla f(w_\star)] = \nabla f(w_\star) = 0$. $\qquad\square$

With these in hand, we can prove

Proposition 4. *With $g_t = \nabla f(w_t) - \nabla f(v_k) + \nabla F(v_k) + b_t$ for some points $w_t, v_k \in W$ and $b_t \in \mathbb{R}$, we have*

$$\mathbb{E}[\|g_t\|^2] \leq 8L\,\mathbb{E}[F(w_t) - F(w_\star)] + 8L\,\mathbb{E}[F(v_k) - F(w_\star)] + 2\,\mathbb{E}[\|b_t\|^2]$$

Proof. Observe that $\mathbb{E}[\nabla f(w_\star) - \nabla f(v_k)] = \nabla F(w_\star) - \nabla F(v_k)$, so that by Lemma 3

$$\mathbb{E}[\|\nabla f(w_\star) - \nabla f(v_k) + \nabla F(v_k) - \nabla F(w_\star)\|^2 \leq \mathbb{E}[\|\nabla f(w_\star) - \nabla f_t(v_k)\|^2] \\ \leq 2L(F(v_k) - F(w_\star))$$

Using this we have

$$\begin{aligned} \mathbb{E}[\|g_t\|^2] &\leq \mathbb{E}[\|\nabla f(w_t) - \nabla f(v_k) + \nabla F(v_k) + b_t\|^2] \\ &\leq \mathbb{E}[2\|\nabla f(w_t) - \nabla f(v_k) + \nabla F(v_k)\|^2 + 2\|b_t\|^2] \\ &\leq \mathbb{E}[4\|\nabla f(w_t) - \nabla f(w_\star)\|^2 + 4\|\nabla f(w_\star) - \nabla f(v_k) + \nabla F(v_k) - \nabla F(w_\star)\|^2 + 2\|b_t\|^2] \\ &\leq 8L\,\mathbb{E}[F(w_t) - F(w_\star)] + 4\,\mathbb{E}[\|\nabla f(w_\star) - \nabla f(v_k)\|^2] + 2\,\mathbb{E}[\|b_t\|^2] \\ &\leq 8L\,\mathbb{E}[F(w_t) - F(w_\star)] + 8L\,\mathbb{E}[F(v_k) - F(w_\star)] + 2\,\mathbb{E}[\|b_t\|^2] \end{aligned}$$

$\square$

D   Biased Online Learning with SVRG updates

In this section are finally prepared to analyze SVRG OL. In order to do so, we restate the algorithm as Algorithm 2. This algorithm is identical to SVRG OL, but we have introduced the additional notation $b_t = \nabla \hat{F}(v_k) - \nabla F(v_k)$ so that we can write $g_t = \nabla f(w_t) - \nabla f(v_k) + \nabla F(v_k) + b_t$ instead of $g_t = \nabla f(w_t) - \nabla f(v_k) + \nabla \hat{F}(v_k)$. Factoring out the term b_t and writing g_t in this way makes clearer how we are able to apply the analysis of biased gradients to the analysis of online learning in the previous section. We analyze Algorithm 2 for online learning algorithms $\mathcal{A}$ that obtain *second-order* regret guarantees, $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^T \|g_t\|^2}$ as well as ones that obtain *first-order* regret guarantees $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^T \|g_t\|}$. Algorithm in these families include the well-known AdaGrad [8] algorithm and its unconstrained variant SOLO [15] (second order, $\psi(w_\star) = O(\|w_\star\|^2)$) as well as FreeRex [6] and PiSTOL [14] (first order, $\psi(w_\star) = \tilde{O}(\|w_\star\|)$). We will show that for sufficiently small b_t, such algorithms result in $\overline{w} = \frac{1}{T}\sum_{t=1}^T w_t$ such that $\mathbb{E}[F(\overline{w}) - F(w_\star)] = O(1/T)$.

Algorithm 2 Online Learning with Biased Variance-Reduced Gradients
---

    Initialize: initial point w_1, epoch lengths $0 = T_0, T_1, \ldots, T_K$ online learning algorithm $\mathcal{A}$.
    Get initial vector w_1 from $\mathcal{A}$.
    for $k = 1$ **to** K **do**
        for $t = T_{0:k-1} + 1$ **to** $T_{0:k}$ **do**
            Sample $f \sim \mathcal{D}$.
            $g_t \leftarrow \nabla f(w_t) - \nabla f(v_k) + \nabla F(v_k) + b_t$.
            Send g_t to the online learning algorithm $\mathcal{A}$.
            Get updated vector w_{t+1} from $\mathcal{A}$.
        end for
        $v_{k+1} \leftarrow \frac{1}{T_k}\sum_{t=T_{1:k-1}+1}^{T_{1:k}} w_t$.
    end for

Proposition 5. *If* $T_k/T_{k-1} \leq \rho$ *for all* k, *then*

$$\sum_{k=1}^K T_k\,\mathbb{E}[F(v_k) - F(w_\star)] \leq T_1\,\mathbb{E}[F(v_1) - F(w_\star)] + \rho\sum_{t=1}^T \mathbb{E}[F(w_t) - F(w_\star)]$$

Proof. First we show that for all $k > 1$,

$$T_k\,\mathbb{E}[F(v_k) - F(w_\star)] \leq \rho\sum_{t=T_{0:k-2}+1}^{T_{0:k-1}} \mathbb{E}[F(w_t) - F(w_\star)]$$

This follows from Jensen's inequality by convexity of F:

$$\mathbb{E}[F(v_k) - F(w_\star)] = \mathbb{E}\left[F\left(\frac{1}{T_{k-1}}\sum_{t=T_{0:k-2}+1}^{T_{0:k-1}} w_t\right) - F(w_\star)\right]$$

$$\leq \frac{1}{T_{k-1}}\mathbb{E}\left[\sum_{t=T_{0:k-2}+1}^{T_{0:k-1}} F(w_t) - \mathbb{E}[F(w_\star)]\right]$$

$$\leq \frac{\rho}{T_k}\sum_{t=T_{0:k-2}+1}^{T_{0:k-1}} \mathbb{E}[F(w_t) - F(w_\star)]$$

Now the result of the proposition is immediate:

$$\sum_{k=1}^{K} T_k\,\mathbb{E}[F(v_k) - F(w_\star)] \leq T_1\,\mathbb{E}[F(v_1) - F(w_\star)] + \sum_{k=2}^{K}\rho\sum_{t=T_{0:k-2}}^{T_{0:k-1}} \mathbb{E}[F(w_t) - F(w_\star)]$$

$$\leq T_1\,\mathbb{E}[F(v_1) - F(w_\star)] + \sum_{k=0}^{K}\rho\sum_{t=T_{0:k}}^{T_{0:k}} \mathbb{E}[F(w_t) - F(w_\star)]$$

$$= T_1\,\mathbb{E}[F(v_1) - F(w_\star)] + \rho\sum_{t=1}^{T} \mathbb{E}[F(w_t) - F(w_\star)]$$

$\square$

Proposition 6. *Suppose the online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|^2}$, and $T_k/T_{k-1} \leq \rho$ for some constant ρ. Then*

$$\mathbb{E}\left[\sum_{t=1}^{T} F(w_t) - F(w_\star)\right] \leq \psi(w_\star)\sqrt{8(1+\rho)L\sum_{t=1}^{T}\mathbb{E}[F(w_t) - F(w_\star)] + 8LT_1\,\mathbb{E}[F(v_1) - F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}$$

$$+ \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]$$

Proof. The proof follows by applying, in order, Propositions 1, 4, and 5:

$$\mathbb{E}[\sum_{t=1}^{T} F(w_t) - F(w_\star)] \leq \mathbb{E}[R_T(w_\star)] + \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]$$

$$\leq \psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|^2} + \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]$$

$$\leq \psi(w_\star)\sqrt{8L\sum_{t=1}^{T}\mathbb{E}[F(w_t) - F(w_\star)] + 8L\sum_{k=1}^{K} T_k\,\mathbb{E}[F(v_k) - F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}$$

$$+ \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]$$

$$\leq \psi(w_\star)\sqrt{8(1+\rho)L\sum_{t=1}^{T}\mathbb{E}[F(w_t) - F(w_\star)] + 8LT_1\,\mathbb{E}[F(v_1) - F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}$$

$$+ \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]$$

$\square$

In order to use the above Proposition 6, we need a small technical result:

Proposition 7. *If a, b, c and d are non-negative constants such that*

$$x \leq a\sqrt{bx + c} + d$$

Then

$$x \leq 4a^2 b + 2a\sqrt{c} + 2d$$

Proof. Suppose $x \geq 2d$. Then we have

$$\frac{x}{2} \leq x - d \leq a\sqrt{bx + c}$$
$$x^2 \leq 4a^2 bx + 4a^2 c$$

Now we use the quadratic formula to obtain

$$x \leq \frac{4a^2 b}{2} + \frac{\sqrt{16a^4 b^2 + 16a^2 c}}{2}$$
$$\leq 4a^2 b + 2a\sqrt{c}$$

Since we assumed $x \geq 2d$ to obtain this bound, we conclude that x is at most the maximum of $4a^2 b + 2a\sqrt{c}$ and $2d$, which is bounded by their sum. $\square$

Now we apply this result to obtain the formal statement of Proposition 2:

Proposition 8. *Suppose the online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T} \|g_t\|^2}$. Then for $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$,*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{32(1+\rho)\psi(w_\star)^2 L}{T} + \frac{2\psi(w_\star)\sqrt{8LT_1 \mathbb{E}[F(v_1) - F(w_\star)] + 2\sum_{t=1}^{T} \mathbb{E}[\|b_t\|^2]}}{T}$$
$$+ \frac{2\sum_{t=1}^{T} \mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]}{T}$$

In particular, if $\|b_t\| \leq \frac{\sigma}{T}$ for all t for some σ, and V has diameter D, then

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{32(1+\rho)\psi(w_\star)^2 L}{T} + \frac{4\psi(w_\star)\sigma}{T\sqrt{T}} + \frac{8\psi(x)\sqrt{T_1 \mathbb{E}[F(v_1) - F(w_\star)]}}{T} + 2\frac{\sigma D}{T}$$

Proof. Applying Propositions 5 to the result of Proposition 6, we have

$$\mathbb{E}[\sum_{t=1}^{T} F(w_t) - F(w_\star)] \leq 32(1+\rho)\psi(w_\star)^2 L + 2\psi(w_\star)\sqrt{8LT_1 \mathbb{E}[F(v_1) - F(w_\star)] + 2\sum_{t=1}^{T} \mathbb{E}[\|b_t\|^2]}$$
$$+ 2\sum_{t=1}^{T} \mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]$$

Now divide by T and use Jensen's inequality to conclude the first result. The second follows since $\sqrt{a + b} \leq \sqrt{2}(\sqrt{a} + \sqrt{b})$ for all positive a, b. $\square$

Now let's prove bounds using the regularization trick from Proposition 3 that allows us to avoid assuming a finite-diameter domain.

Proposition 9. *Suppose the online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T} \|g_t\|^2}$. Let B be a uniform upper bound on $\|b_t\|$. Run $\mathcal{A}$ on the gradients $\|g_t\| + B\frac{w_t}{\|w_t\|}$. Then for $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$,*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{64(1+\rho)\psi(w_\star)^2 L}{T} + \frac{2\psi(w_\star)\sqrt{16LT_1 \mathbb{E}[F(v_1) - F(w_\star)] + 6TB^2}}{T} + 2B\|w_\star\|$$

In particular, if $B \leq \frac{\sigma}{T}$, we have

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{64(1+\rho)\psi(w_\star)^2 L}{T} + \frac{2\psi(w_\star)\sqrt{16LT_1 \mathbb{E}[F(v_1) - F(w_\star)] + 6\sigma/T}}{T} + 2\frac{\sigma\|w_\star\|}{T}$$

15

Proof.

$$\mathbb{E}[\sum_{t=1}^{T} F(w_t) - F(w_\star)] \leq \mathbb{E}[R_T(w_\star)] + \sum_{t=1}^{T}\mathbb{E}[b_t \cdot (w_\star - w_t)] + B\|w_\star\| - B\|w_t\|$$

$$\leq \mathbb{E}[R_T(w_\star)] + 2TB\|w_\star\|$$

$$\leq \psi(w_\star)\sqrt{\sum_{t=1}^{T} 2\,\mathbb{E}[\|g_t\|^2] + 2TB^2} + 2TB\|w_\star\|$$

$$\leq \psi(w_\star)\sqrt{16L\sum_{t=1}^{T}\mathbb{E}[F(w_t) - F(w_\star)] + 16L\sum_{k=1}^{K} T_k\,\mathbb{E}[F(v_k) - F(w_\star)] + 4\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2] + 2TB^2}$$
$$+ 2TB\|w_\star\|$$

$$\leq \psi(w_\star)\sqrt{16(1+\rho)L\sum_{t=1}^{T}\mathbb{E}[F(w_t) - F(w_\star)] + 16LT_1\,\mathbb{E}[F(v_1) - F(w_\star)] + 4\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2] + 2TB^2}$$
$$+ 2TB\|w_\star\|$$

$$\leq \psi(w_\star)\sqrt{16(1+\rho)L\sum_{t=1}^{T}\mathbb{E}[F(w_t) - F(w_\star)] + 16LT_1\,\mathbb{E}[F(v_1) - F(w_\star)] + 6TB^2}$$
$$+ 2TB\|w_\star\|$$

Now use Proposition 7:

$$\mathbb{E}[\sum_{t=1}^{T} F(w_t) - F(w_\star)] \leq 64(1+\rho)\psi(w_\star)^2 L + 2\psi(w_\star)\sqrt{16LT_1\,\mathbb{E}[F(v_1) - F(w_\star)] + 6TB^2} + 2TB\|w_\star\|$$

$\square$

We can use Proposition 9 to prove an analogue of Theorem 1:

Theorem 3. *Suppose the online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|^2}$.*
Set $b_t = \|\nabla\hat{F}(v_k) - \nabla F(v_k)\|$ for $t \in [T_{0:k-1} + 1, T_{1:k}]$. Suppose that $T_k/T_{k-1} \leq \rho$ for all k. Let B be a uniform upper bound on $\|b_t\|$. Run $\mathcal{A}$ on the gradients $g_t + B\frac{w_t}{\|w_t\|}$. Then for $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$,

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq 2B\|w_\star\| + \frac{64(1+\rho)\psi(w_\star)^2 L}{T} + \frac{2\psi(w_\star)\sqrt{16LT_1\,\mathbb{E}[F(v_1) - F(w_\star)] + 6TB^2}}{T}$$

E   Other Types of Regret Bounds

The above Theorems 1 and 3 postulate a regret bound of the form $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|^2}$. This bound is achieved by, e.g. the AdaGrad [8] or SOLO [15] algorithms, both of which have $\psi(w_\star) = O(\|w_\star\|^2)$. However, there also exist algorithms (e.g. PiSTOL [14] or FreeRex [6]) that improve ψ to $O(\|x\|)$ up to log factors, but in return achieve only a *first-order* regret bound of $R_T(w_\star) \leq \psi(w_\star)\sqrt{G\sum_{t=1}^{T}\|g_t\|}$, where G is a uniform bound on $\|g_t\|$. Such algorithms are sometimes called parameter-free algorithms because they achieve the minimax optimal regret bound of $\|w_\star\|G\sqrt{T}$ in an unconstrained setting without requiring any hyperparameter tuning. In this section we show similar convergence guarantees for these parameter-free algorithms.

We start with a simple LaGrange multipliers argument:

Proposition 10. *Suppose $a_1, \ldots, a_t$ are non-negative real numbers. Then*

$$\sum_{t=1}^{T} a_t \leq \sqrt{T}\sqrt{\sum_{t=1}^{T} a_t^2}$$

Proof. Let $S = \sum_{t=1}^{T} a_t^2$. We want to maximize $\sum_{t=1}^{T} a_t$ subject to $\sum_{t=1}^{T} a_t^2 \leq S$. Then applying the method of LaGrange multipliers, we see that for the optimizing values of a_t, there is some multiplier λ such that for all t, either $\lambda = 2a_t$ or $a_t = 0$. Therefore $a_t \leq \sqrt{S/T}$ and so $\sum_{t=1}^{T} a_t \leq \sqrt{TS} = \sqrt{T}\sqrt{\sum_{t=1}^{T} a_t^2}$. $\square$

Using Proposition 10, we can prove an analogue of Proposition 6:

Proposition 11. *Suppose the online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|}$, and $T_k/T_{k-1} \leq \rho$ for some constant ρ. Then*

$$\mathbb{E}\left[\sum_{t=1}^{T}F(w_t) - F(w_\star)\right] \leq \psi(w_\star)\sqrt{\sqrt{T}\sqrt{8(1+\rho)L\sum_{t=1}^{T}\mathbb{E}[F(w_t)-F(w_\star)] + 8LT_1\,\mathbb{E}[F(v_1)-F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}}$$

$$+ \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t-w_\star\|)]$$

Proof. The proof is nearly identical to that of Proposition 6:

$$\mathbb{E}[\sum_{t=1}^{T}F(w_t)-F(w_\star)] \leq \mathbb{E}[R_T(w_\star)] + \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t-w_\star\|)]$$

$$\leq \psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|} + \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t-w_\star\|)]$$

$$\leq \psi(w_\star)\sqrt{\sqrt{T}\sqrt{\sum_{t=1}^{T}\|g_t\|^2}} + \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t-w_\star\|)]$$

$$\leq \psi(w_\star)\sqrt{\sqrt{T}\sqrt{8L\sum_{t=1}^{T}\mathbb{E}[F(w_t)-F(w_\star)] + 8L\sum_{k=1}^{K}T_k\,\mathbb{E}[F(v_k)-F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}}$$

$$+ \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t-w_\star\|)]$$

$$\leq \psi(w_\star)\sqrt{\sqrt{T}\sqrt{8(1+\rho)L\sum_{t=1}^{T}\mathbb{E}[F(w_t)-F(w_\star)] + 8LT_1\,\mathbb{E}[F(v_1)-F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}}$$

$$+ \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t-w_\star\|)]$$

$\square$

Now in order to use Proposition 11 we need a slightly different technical result than Proposition 7:

Proposition 12. *If a, b, c, d and e, and f are non-negative constants such that*

$$x \leq a\sqrt{b\sqrt{cx+d}+e}+f$$

then

$$x \leq 2^{7/3}a^{4/3}b^{2/3}c^{1/3} + 2a\sqrt{2b\sqrt{2d}+2e}+2f$$

Proof. Since $\sqrt{x+y} \leq \sqrt{2x}+\sqrt{2y}$ for all non-negative x and y, we have

$$x \leq a\sqrt{b\sqrt{2cx}+b\sqrt{2d}+e}+f$$

$$\leq a\sqrt{2b\sqrt{2cx}} + a\sqrt{2b\sqrt{2d}+2e}+f$$

Now suppose $x \geq 2a\sqrt{2b\sqrt{2d}+2e}+2f$. Then

$$\frac{x}{2} \leq a\sqrt{2b\sqrt{2cx}}$$

$$x^4 \leq 2^7 a^4 b^2 cx$$

$$x \leq 2^{7/3}a^{4/3}b^{2/3}c^{1/3}$$

So that taking into account our condition $x \geq 2a\sqrt{2b\sqrt{2d} + 2e} + 2f$, we have

$$x \leq 2^{7/3}a^{4/3}b^{2/3}c^{1/3} + 2a\sqrt{2b\sqrt{2d} + 2e} + 2f$$

as desired. $\qquad\square$

With this in hand, we can prove an analogue of Proposition 8:

Proposition 13. *Suppose the online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T} \|g_t\|}$. Then for $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$,*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{2^{10/3}\psi(x^*)^{4/3}(1+\rho)^{1/3}L^{1/3}}{T^{2/3}}$$

$$+ \frac{2^{7/4}\psi(x^*)\sqrt{8LT_1\,\mathbb{E}[F(v_1) - F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}}{T^{3/4}} + 2\frac{\sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]}{T}$$

In particular, if $\|b_t\| \leq \frac{\sigma}{T^{2/3}}$ for all t for some σ, and V has diameter D, then

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{2^{10/3}\psi(x^*)^{4/3}(1+\rho)^{1/3}L^{1/3}}{T^{2/3}}$$

$$+ \frac{2^{11/4}\sigma}{T^{13/12}} + \frac{2^{15/4}\psi(x^*)\sqrt{LT_1\,\mathbb{E}[F(v_1) - F(w_\star)]}}{T^{3/4}} + 2\frac{\sigma D}{T}$$

Proof. The proof is directly analogous to the proof of Proposition 8; we simply apply Proposition 12 to Proposition 11:

$$\mathbb{E}\left[\sum_{t=1}^{T} F(w_t) - F(w_\star)\right] \leq \psi(w_\star)\sqrt{\sqrt{T}\sqrt{8(1+\rho)L\sum_{t=1}^{T}\mathbb{E}[F(w_t) - F(w_\star)] + 8LT_1\,\mathbb{E}[F(w_1) - F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}}$$

$$+ \sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]$$

$$\mathbb{E}\left[\sum_{t=1}^{T} F(w_t) - F(w_\star)\right] \leq 2^{10/3}\psi(x^*)^{4/3}T^{1/3}(1+\rho)^{1/3}L^{1/3}$$

$$+ 2^{7/4}\psi(x^*)T^{1/4}\sqrt{8LT_1\,\mathbb{E}[F(w_1) - F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]}$$

Now divide by T to obtain the result. $\qquad\square$

Finally, we prove the analogue of Proposition 9:

Proposition 14. *Suppose the online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T} \|g_t\|^2}$. Let B be a uniform upper bound on $\|b_t\|$. Run $\mathcal{A}$ on the gradients $\|g_t\| + B\frac{w_t}{\|w_t\|}$. Then for $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$,*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq 2^4\psi(x^*)^{4/3}T^{1/3}(1+\rho)^{1/3}L^{1/3}$$

$$+ 2^{9/4}\psi(x^*)T^{1/4}\sqrt{8LT_1\,\mathbb{E}[F(v_1) - F(w_\star)] + 2TB^2} + 2\psi(w_\star)\sqrt{2TB} + 4TB\|w_\star\|$$

In particular, if $B \leq \frac{\sigma}{T^{2/3}}$, we have

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{2^4\psi(x^*)^{4/3}(1+\rho)^{1/3}L^{1/3}}{T^{2/3}}$$

$$+ \frac{2^{17/4}\psi(x^*)\sqrt{LT_1\,\mathbb{E}[F(v_1) - F(w_\star)]}}{T^{3/4}} + \frac{2^{13/4}\psi(x^*)\sigma}{T^{11/12}} + \frac{2\psi(w_\star)\sqrt{2\sigma}}{T^{5/6}} + \frac{4\|w_\star\|}{T^{2/3}}$$

18

Proof.

$$\mathbb{E}[\sum_{t=1}^{T} F(w_t) - F(w_\star)] \le \mathbb{E}[R_T(w_\star)] + \sum_{t=1}^{T} \mathbb{E}[b_t \cdot (w_\star - w_t)] + B\|w_\star\| - B\|w_t\|$$

$$\le \mathbb{E}[R_T(w_\star)] + 2TB\|w_\star\|$$

$$\le \psi(w_\star)\sqrt{\sum_{t=1}^{T} \mathbb{E}[\|g_t\|]} + TB + 2TB\|w_\star\|$$

$$\le \psi(w_\star)\sqrt{2\sum_{t=1}^{T} \mathbb{E}[\|g_t\|]} + \psi(w_\star)\sqrt{2TB} + 2TB\|w_\star\|$$

$$\le \psi(w_\star)\sqrt{2\sqrt{T}\sqrt{8L\sum_{t=1}^{T}\mathbb{E}[F(w_t) - F(w_\star)] + 8L\sum_{k=1}^{K} T_k\, \mathbb{E}[F(v_k) - F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}}$$
$$+ \psi(w_\star)\sqrt{2TB} + 2TB\|w_\star\|$$

$$\le \psi(w_\star)\sqrt{2\sqrt{T}\sqrt{8(1+\rho)L\sum_{t=1}^{T}\mathbb{E}[F(w_t) - F(w_\star)] + 8LT_1\, \mathbb{E}[F(v_1) - F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}}$$
$$+ \psi(w_\star)\sqrt{2TB} + 2TB\|w_\star\|$$

Now we apply Proposition 12:

$$\mathbb{E}[\sum_{t=1}^{T} F(w_t) - F(w_\star)] \le 2^4 \psi(x^*)^{4/3} T^{1/3}(1+\rho)^{1/3} L^{1/3}$$

$$+ 2^{9/4}\psi(x^*)T^{1/4}\sqrt{8LT_1\, \mathbb{E}[F(v_1) - F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2] + 2\psi(w_\star)\sqrt{2TB} + 4TB\|w_\star\|}$$

$$\le 2^4 \psi(x^*)^{4/3} T^{1/3}(1+\rho)^{1/3} L^{1/3}$$
$$+ 2^{9/4}\psi(x^*)T^{1/4}\sqrt{8LT_1\, \mathbb{E}[F(v_1) - F(w_\star)] + 2TB^2} + 2\psi(w_\star)\sqrt{2TB} + 4TB\|w_\star\|$$

$$\square$$

Combining all these together, we can prove analogues of Theorems 1 and 3:

Theorem 4. *Suppose the online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \le \psi(w_\star)\sqrt{\sum_{t=1}^{T} \|g_t\|}$.
Set $b_t = \|\nabla\hat{F}(v_k) - \nabla F(v_k)\|$ for $t \in [T_{0:k-1} + 1, T_{1:k}]$. Suppose that $T_k/T_{k-1} \le \rho$ for all k. Then for $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$,*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \le \frac{2^{10/3}\psi(x^*)^{4/3}(1+\rho)^{1/3}L^{1/3}}{T^{2/3}} + \frac{2^{7/4}\psi(x^*)\sqrt{8LT_1\,\mathbb{E}[F(v_1) - F(w_\star)] + 2\sum_{t=1}^{T}\mathbb{E}[\|b_t\|^2]}}{T^{3/4}}$$
$$+ 2\frac{\sum_{t=1}^{T}\mathbb{E}[\|b_t\|(\|w_t - w_\star\|)]}{T}$$

In particular, with probability at least $1 - \frac{1}{T}$ we have $\|b_t\| \le \frac{\sigma\sqrt{\log(KT)}}{\sqrt{\hat{N}}}$ for some σ, and if $\hat{N} > T^{4/3}$ and V has diameter D, then

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \le \frac{2^{10/3}\psi(x^*)^{4/3}(1+\rho)^{1/3}L^{1/3}}{T^{2/3}} + \frac{2^{11/4}\sigma\sqrt{\log(KT)}}{T^{13/12}}$$
$$+ \frac{2^{15/4}\psi(x^*)\sqrt{LT_1\,\mathbb{E}[F(v_1) - F(w_\star)]}}{T^{3/4}} + 2\frac{\sigma\sqrt{\log(KT)}D}{T} + \frac{GD}{T}$$
$$= O\left(\frac{\sqrt{\log(KT)}}{T} + \frac{1}{T^{2/3}}\right)$$

Theorem 5. *Suppose the online learning algorithm $\mathcal{A}$ guarantees regret $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|}$. Set $b_t = \|\nabla\hat{F}(v_k) - \nabla F(v_k)\|$ for $t \in [T_{0:k-1}+1, T_{1:k}]$. Suppose that $T_k/T_{k-1} \leq \rho$ for all k. Let B be a uniform upper bound on $\|b_t\|$. Run $\mathcal{A}$ on the gradients $g_t + B\frac{w_t}{\|w_t\|}$. Then for $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$,*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] \leq \frac{2^4\psi(x^*)^{4/3}(1+\rho)^{1/3}L^{1/3}}{T^{2/3}} + \frac{2^{9/4}\psi(x^*)\sqrt{8LT_1\,\mathbb{E}[F(v_1) - F(w_\star)] + 2TB^2}}{T^{3/4}}$$
$$+ \frac{2\psi(w_\star)\sqrt{2B}}{\sqrt{T}} + 4B\|w_\star\|$$

We also have an analogue of Theorem 2 using essentially the same argument, but using the setting $\hat{N} = \Theta(T^{4/3})$ rather than $\hat{N} = \Theta(T^2)$:

Theorem 6. *Set $T_k = 2T_{k-1}$. Suppose the base optimizer $\mathcal{A}$ in SVRG OL guarantees regret $R_T(w_\star) \leq \psi(w_\star)\sqrt{\sum_{t=1}^{T}\|g_t\|}$, and the domain W has finite diameter D. Let $\hat{N} = \Theta(T^{4/3})$ and $N = K\hat{N} + T$ be the total number of data points observed. Suppose we compute the batch gradients $\nabla\hat{F}(v_k)$ in parallel on m machines with $m < N^{1/4}$. Then for $\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$ we obtain*

$$\mathbb{E}[F(\overline{w}) - F(w_\star)] = \tilde{O}\left(\frac{1}{\sqrt{N}}\right)$$

in time $\tilde{O}(N/m)$ and space $O(1)$

F Benchmark results

Table 4: Average loss and AUC achieved by Logistic regression implemented in Spark ML, VW and SVRG OL. "Comm." refers to number of communication rounds and time is measured in minutes. In case of MiniBatch SGD and SVRG, we computed the full batch gradient in each iteration based on the whole training data.

Data	Comm.	Train	Test	AUC	Time
		KDD10			
Spark ML	100	0.25380	0.26317	84.778	52
Spark ML	500	0.25303	**0.26234**	**84.880**	101
VW	100	0.24462	0.26849	84.525	38
VW	500	**0.18879**	0.27817	83.967	96
MiniBatch SGD	100	0.36051	0.36711	80.603	33
MiniBatch SGD	500	0.35292	0.359483	80.932	121
SVRG	100	0.31743	0.32981	81.121	35
SVRG	500	0.31354	0.32346	81.928	102
SVRG OL	4	0.26085	0.26525	84.459	6
		KDD12			
Spark ML	100	0.15756	0.15589	75.485	36
Spark ML	500	0.15755	0.15570	75.453	180
VW	100	0.15398	0.15725	77.871	44
VW	500	**0.14866**	0.15550	**78.881**	150
MiniBatch SGD	100	0.27689	0.27420	70.207	43
MiniBatch SGD	500	0.27420	0.27689	70.207	225
SVRG	100	0.24325	0.23754	72.543	54
SVRG	500	0.22784	0.22397	73.764	202
SVRG OL	4	0.152740	**0.154985**	78.431	8
		ADS SMALL			
Spark ML	100	0.23372	0.22288	83.356	42
Spark ML	500	0.23365	0.22286	83.365	245
VW	100	0.23381	0.22347	83.214	114
VW	500	0.23157	0.22251	**83.499**	396
SVRG OL	14	**0.23147**	**0.22244**	83.479	94
Data		ADS LARGE			
Spark ML	100	0.23965	0.23263	82.646	216
Spark ML	500	0.23958	0.23256	82.655	1214
VW	100				FAIL
SVRG OL	31	**0.23753**	**0.23197**	**82.830**	334
Data		EMAIL			
Spark ML	100	0.27837	0.29598	88.852	117
Spark ML	500	**0.27773**	**0.28887**	**88.863**	601
VW	100	0.35812	0.33919	84.414	74
VW	500	0.33094	0.33010	85.854	358
SVRG OL	14	0.30567	0.29889	88.321	110