Better Mini-Batch Algorithms via Accelerated Gradient Methods

Andrew Cotter
Toyota Technological Institute at Chicago
cotter@ttic.edu

Ohad Shamir
Microsoft Research, NE
ohadsh@microsoft.com

Nathan Srebro
Toyota Technological Institute at Chicago
nati@ttic.edu

Karthik Sridharan
Toyota Technological Institute at Chicago
karthik@ttic.edu

Abstract

Mini-batch algorithms have been proposed as a way to speed-up stochastic convex optimization problems. We study how such algorithms can be improved using accelerated gradient methods. We provide a novel analysis, which shows how standard gradient methods may sometimes be insufficient to obtain a significant speed-up and propose a novel accelerated gradient algorithm, which deals with this deficiency, enjoys a uniformly superior guarantee and works well in practice.

1 Introduction

We consider a stochastic convex optimization problem of the form $\min_{\mathbf{w}\in\mathcal{W}} L(\mathbf{w})$, where $L(\mathbf{w}) = \mathbb{E}_z\left[\ell(\mathbf{w}, z)\right]$, based on an empirical sample of instances $z_1, \ldots, z_m$. We assume that $\mathcal{W}$ is a convex subset of some Hilbert space (which in this paper, we will take to be Euclidean space), and ℓ is non-negative, convex and smooth in its first argument (i.e. has a Lipschitz-constinuous gradient). The classical learning application is when $z = (\mathbf{x}, y)$ and $\ell(\mathbf{w}, (\mathbf{x}, y))$ is a prediction loss. In recent years, there has been much interest in developing efficient first-order stochastic optimization methods for these problems, such as mirror descent [2, 6] and dual averaging [9, 16]. These methods are characterized by incremental updates based on subgradients $\partial\ell(\mathbf{w}, z_i)$ of individual instances, and enjoy the advantages of being highly scalable and simple to implement.

An important limitation of these methods is that they are inherently sequential, and so problematic to parallelize. A popular way to speed-up these algorithms, especially in a parallel setting, is via *mini-batching*, where the incremental update is performed on an average of the subgradients with respect to several instances at a time, rather than a single instance (i.e., $\frac{1}{b}\sum_{j=1}^{b}\partial\ell(\mathbf{w}, z_{i+j})$). The gradient computations for each mini-batch can be parallelized, allowing these methods to perform faster in a distributed framework (see for instance [11]). Recently, [10] has shown that a mini-batching distributed framework is capable of attaining asymptotically optimal speed-up in general (see also [1]).

A parallel development has been the popularization of *accelerated* gradient descent methods [7, 8, 15, 5]. In a deterministic optimization setting and for general smooth convex functions, these methods enjoy a rate of $O(1/n^2)$ (where n is the number of iterations) as opposed to $O(1/n)$ using standard methods. However, in a stochastic setting (which is the relevant one for learning problems), the rate of both approaches have an $O(1/\sqrt{n})$ dominant term in general, so the benefit of using accelerated methods for learning problems is not obvious.

Algorithm 1 Stochastic Gradient Descent with Mini-Batching (SGD)

Parameters: Step size η, mini-batch size b.
Input: Sample $z_1, \ldots, z_m$
$\mathbf{w}_1 = 0$
for $i = 1$ to $n = m/b$ **do**
  Let $\ell_i(\mathbf{w}_i) = \frac{1}{b} \sum_{t=b(i-1)+1}^{bi} \ell(\mathbf{w}_i, z_t)$
  $\mathbf{w}'_{i+1} := \mathbf{w}_i - \eta \nabla \ell_i(\mathbf{w}_i))$
  $\mathbf{w}_{i+1} := P_{\mathcal{W}}(\mathbf{w}'_{i+1})$
end for
Return $\bar{\mathbf{w}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{w}_i$

Algorithm 2 Accelerated Gradient Method (AG)

Parameters: Step sizes (γ_i, β_i), mini-batch size b
Input: Sample $z_1, \ldots, z_m$
$\mathbf{w} = 0$
for $i = 1$ to $n = m/b$ **do**
  Let $\ell_i(\mathbf{w}_i) := \frac{1}{b} \sum_{t=b(i-1)+1}^{bi} \ell(\mathbf{w}, z_t)$
  $\mathbf{w}_i^{\mathrm{md}} := \beta_i^{-1} \mathbf{w}_i + (1 - \beta_i^{-1}) \mathbf{w}_i^{\mathrm{ag}}$
  $\mathbf{w}'_{i+1} := \mathbf{w}_i^{\mathrm{md}} - \gamma_i \nabla \ell_i(\mathbf{w}_i^{\mathrm{md}})$
  $\mathbf{w}_{i+1} := P_{\mathcal{W}}(\mathbf{w}'_{i+1})$
  $\mathbf{w}_{i+1}^{\mathrm{ag}} \leftarrow \beta_i^{-1} \mathbf{w}_{i+1} + (1 - \beta_i^{-1}) \mathbf{w}_i^{\mathrm{ag}}$
end for
Return $\mathbf{w}_n^{\mathrm{ag}}$

In this paper, we study the application of accelerated methods for mini-batch algorithms, and provide theoretical results, a novel algorithm, and empirical experiments. The main resulting message is that by using an appropriate accelerated method, we obtain significantly better stochastic optimization algorithms in terms of convergence speed. Moreover, in certain regimes acceleration is actually *necessary* in order to allow a significant speedups. The potential benefit of acceleration to mini-batching has been briefly noted in [4], but here we study this issue in much more depth. In particular, we make the following contributions:

- We develop novel convergence bounds for the standard gradient method, which refines the result of [10, 4] by being dependent on $L(\mathbf{w}^\star)$, the expected loss of the best predictor in our class. For example, we show that in the regime where the desired suboptimality is comparable or larger than $L(\mathbf{w}^\star)$, including in the separable case $L(\mathbf{w}^\star) = 0$, mini-batching does not lead to significant speed-ups with standard gradient methods.
- We develop a novel variant of the stochastic accelerated gradient method [5], which is optimized for a mini-batch framework and implicitly adaptive to $L(\mathbf{w}^\star)$.
- We provide an analysis of our accelerated algorithm, refining the analysis of [5] by being dependent on $L(\mathbf{w}^\star)$, and show how it always allows for significant speed-ups via mini-batching, in contrast to standard gradient methods. Moreover, its performance is uniformly superior, at least in terms of theoretical upper bounds.
- We provide an empirical study, validating our theoretical observations and the efficacy of our new method.

2 Preliminaries

As discussed in the introduction, we focus on a stochastic convex optimization problem, where we wish to minimize $L(\mathbf{w}) = \mathbb{E}_z \left[\ell(\mathbf{w}, z)\right]$ over some convex domain $\mathcal{W}$, using an i.i.d. sample $z_1, \ldots, z_m$. Throughout this paper we assume that the instantaneous loss $\ell(\cdot, z)$ is convex, non-negative and H-smooth for each $z \in \mathcal{Z}$. Also in this paper, we take $\mathcal{W}$ to be the set $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\| \leq D\}$, although our results can be generalized.

We discuss two stochastic optimization approaches to deal with this problem: stochastic gradient descent (SGD), and accelerated gradient methods (AG). In a mini-batch setting, both approaches iteratively average sub-gradients with respect to several instances, and use this average to update the predictor. However, the update is done in different ways.

The stochastic gradient descent algorithm (which in more general settings is known as mirror descent, e.g. [6]) is summarized as Algorithm 1. In the pseudocode, $P_{\mathcal{W}}$ refers to the projection on to the ball $\mathcal{W}$, which amounts to rescaling $\mathbf{w}$ to have norm at most D.

The accelerated gradient method (e.g., [5]) is summarized as Algorithm 2.

In terms of existing results, for the SGD algorithm we have [4, Section 5.1]

$$\mathbb{E}\left[L(\bar{\mathbf{w}})\right] - L(\mathbf{w}^\star) \leq \mathcal{O}\left(\sqrt{\frac{1}{m}} + \frac{b}{m}\right),$$

whereas for an accelerated gradient algorithm, we have [5]

$$\mathbb{E}\left[L(\mathbf{w}_n^{\mathrm{ag}})\right] - L(\mathbf{w}^\star) \leq \mathcal{O}\left(\sqrt{\frac{1}{m}} + \frac{b^2}{m^2}\right).$$

Thus, as long as $b = o(\sqrt{m})$, both methods allow us to use a large mini-batch size b without significantly degrading the performance of either method. This allows the number of iterations $n = m/b$ to be smaller, potentially resulting in faster convergence speed. However, these bounds do not show that accelerated methods have a significant advantage over the SGD algorithm, at least when $b = o(\sqrt{m})$, since both have the same first-order term $1/\sqrt{m}$. To understand the differences between these two methods better, we will need a more refined analysis, to which we now turn.

3 Convergence Guarantees

The following theorems provide a refined convergence guarantee for the SGD algorithm and the AG algorithm, which improves on the analysis of [10, 4, 5] by being explicitly dependent on $L(\mathbf{w}^\star)$, the expected loss of the best predictor $\mathbf{w}^\star$ in $\mathcal{W}$. Due to lack of space, the proofs are only sketched. The full proofs are deferred to the supplementary material.

Theorem 1. *For the Stochastic Gradient Descent algorithm with* $\eta = \min\left\{\frac{1}{2H}, \frac{\sqrt{\frac{bD^2}{L(\mathbf{w}^\star)Hn}}}{1+\sqrt{\frac{HD^2}{L(\mathbf{w}^\star)bn}}}\right\}$*, assuming* $L(0) \leq HD^2$*, we get that*

$$\mathbb{E}\left[L(\bar{\mathbf{w}})\right] - L(\mathbf{w}^\star) \leq \sqrt{\frac{HD^2\ L(\mathbf{w}^\star)}{2bn}} + \frac{2HD^2}{n} + \frac{9HD^2}{bn}$$

Theorem 2. *For the Accelerated Descent algorithm with* $\beta_i = \frac{i+1}{2}$*,* $\gamma_i = \gamma i^p$ *where*

$$\gamma = \min\left\{\frac{1}{4H},\ \sqrt{\frac{bD^2}{412HL(\mathbf{w}^\star)(n-1)^{2p+1}}},\ \left(\frac{b}{1044H(n-1)^{2p}}\right)^{\frac{p+1}{2p+1}}\left(\frac{D^2}{4HD^2+\sqrt{4HD^2L(\mathbf{w}^\star)}}\right)^{\frac{p}{2p+1}}\right\} \tag{1}$$

and

$$p = \min\left\{\max\left\{\frac{\log(b)}{2\log(n-1)}, \frac{\log\log(n)}{2\left(\log(b(n-1)) - \log\log(n)\right)}\right\}, 1\right\}\quad , \tag{2}$$

as long as $n \geq 904$*, we have that*

$$\mathbb{E}\left[L(\mathbf{w}_n^{\mathrm{ag}})\right] - L(\mathbf{w}^\star) \leq 358\sqrt{\frac{HD^2L(\mathbf{w}^\star)}{b(n-1)}} + \frac{1545HD^2}{\sqrt{b}(n-1)} + \frac{1428HD^2\sqrt{\log n}}{b(n-1)} + \frac{4HD^2}{(n-1)^2}$$

We emphasize that Theorem 2 gives more than a theoretical bound: it actually specifies a novel accelerated gradient strategy, where the step size γ_i scales *polynomially* in i, in a way dependent on the minibatch size b and $L(\mathbf{w}^\star)$. While $L(\mathbf{w}^\star)$ may not be known in advance, it does have the practical implication that choosing $\gamma_i \propto i^p$ for some $p < 1$, as opposed to just choosing $\gamma_i \propto i$ as in [5]), might yield superior results.

The key observation used for analyzing the dependence on $L(\mathbf{w}^\star)$ is that for any non-negative H-smooth convex function $f : \mathbf{W} \mapsto \mathbb{R}$, we have [13]:

$$\|\nabla f(\mathbf{w})\| \leq \sqrt{4Hf(\mathbf{w})} \tag{3}$$

This self-bounding property tells us that the norm of the gradient is small at a point if the loss is itself small at that point. This self-bounding property has been used in [14] in the online setting and in [13] in the stochastic setting to get better (faster) rates of convergence for non-negative smooth losses. The implication of this observation are that for any $\mathbf{w} \in \mathbf{W}$, $\|\nabla L(\mathbf{w})\| \leq \sqrt{4HL(\mathbf{w})}$ and $\forall z \in \mathcal{Z}, \|\ell(\mathbf{w}, z)\| \leq \sqrt{4H\ell(\mathbf{w}, z)}$.

Proof sketch for Theorem 1. The proof for the stochastic gradient descent bound is mainly based on the proof techniques in [5] and its extension to the mini-batch case in [10]. Following the line of analysis in [5], one can show that

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} L(\mathbf{w}_i)\right] - L(\mathbf{w}^\star) \leq \frac{\eta}{n-1}\sum_{i=1}^{n-1}\mathbb{E}\left[\|\nabla L(\mathbf{w}_i) - \nabla \ell_i(\mathbf{w}_i)\|^2\right] + \frac{D^2}{2\eta(n-1)}$$

In the case of [5], $\mathbb{E}\left[\|\nabla L(\mathbf{w}_i) - \nabla \ell_i(\mathbf{w}_i)\|\right]$ is bounded by the variance, and that leads to the final bound provided in [5] (by setting η appropriately). As noticed in [10], in the minibatch setting we have $\nabla \ell_i(\mathbf{w}_i) = \frac{1}{b}\sum_{t=b(i-1)+1}^{bi} \ell(\mathbf{w}_i, z_t)$ and so one can further show that

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} L(\mathbf{w}_i)\right] - L(\mathbf{w}^\star) \leq \frac{\eta}{b^2(n-1)}\sum_{i=1}^{n-1}\sum_{\substack{t= \\ (i-1)b+1}}^{ib}\mathbb{E}\,\|\nabla L(\mathbf{w}_i) - \nabla \ell(\mathbf{w}_i, z_t)\|^2 + \frac{D^2}{2\eta(n-1)} \tag{4}$$

In [10], each of $\|\nabla L(\mathbf{w}_i) - \nabla \ell(\mathbf{w}_i, z_t)\|$ is bounded by σ_0 and so setting η, the mini-batch bound provided there is obtained. In our analysis we further use the self-bounding property to (4) and get that

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} L(\mathbf{w}_i)\right] - L(\mathbf{w}^\star) \leq \frac{16H\eta}{b(n-1)}\sum_{i=1}^{n-1}\mathbb{E}\left[L(\mathbf{w}_i)\right] + \frac{D^2}{2\eta(n-1)}$$

rearranging and setting η appropriately gives the final bound. □

Proof sketch for Theorem 2. The proof of the accelerated method starts in a similar way as in [5]. For the γ_i'a and β_i's mentioned in the theorem, following similar lines of analysis as in [5] we get the preliminary bound

$$\mathbb{E}\left[L(\mathbf{w}_n^{\mathrm{ag}})\right] - L(\mathbf{w}^\star) \leq \frac{2\gamma}{(n-1)^{p+1}}\sum_{i=1}^{n-1} i^{2p}\,\mathbb{E}\left[\left\|\nabla L(\mathbf{w}_i^{\mathrm{md}}) - \nabla \ell_i(\mathbf{w}_i^{\mathrm{md}})\right\|^2\right] + \frac{D^2}{\gamma(n-1)^{p+1}}$$

In [5] the step size $\gamma_i = \gamma(i+1)/2$ and $\beta_i = (i+1)/2$ which effectively amounts to $p = 1$ and further similar to the stochastic gradient descent analysis. Furthermore, each $\mathbb{E}\left[\left\|\nabla L(\mathbf{w}_i^{\mathrm{md}}) - \nabla \ell_i(\mathbf{w}_i^{\mathrm{md}})\right\|^2\right]$ is assumed to be bounded by some constant, and thus leads to the final bound provided in [5] by setting γ appropriately. On the other hand, we first notice that due to the mini-batch setting, just like in the proof of stochastic gradient descent,

$$\mathbb{E}\left[L(\mathbf{w}_n^{\mathrm{ag}})\right] - L(\mathbf{w}^\star) \leq \frac{2\gamma}{b^2(n-1)^{p+1}}\sum_{i=1}^{n-1} i^{2p}\sum_{\substack{t= \\ b(i-1)+1}}^{ib}\mathbb{E}\left[\left\|\nabla L(\mathbf{w}_i^{\mathrm{md}}) - \nabla \ell(\mathbf{w}_i^{\mathrm{md}}, z_t)\right\|^2\right] + \frac{D^2}{\gamma(n-1)^{p+1}}$$

Using smoothness, the self bounding property some manipulations, we can further get the bound

$$\mathbb{E}\left[L(\mathbf{w}_n^{\mathrm{ag}})\right] - L(\mathbf{w}^\star) \leq \tfrac{64H\gamma}{b(n-1)^{1-p}} \sum_{i=1}^{n-1} \left(\mathbb{E}\left[L(\mathbf{w}_i^{\mathrm{ag}})\right] - L(\mathbf{w}^\star)\right) + \tfrac{64H\gamma L(\mathbf{w}^\star)(n-1)^p}{b} + \tfrac{D^2}{\gamma(n-1)^{p+1}} + \tfrac{32HD^2}{b(n-1)}$$

Notice that the above recursively bounds $\mathbb{E}\left[L(\mathbf{w}_n^{\mathrm{ag}})\right] - L(\mathbf{w}^\star)$ in terms of $\sum_{i=1}^{n-1}\left(\mathbb{E}\left[L(\mathbf{w}_i^{\mathrm{ag}})\right] - L(\mathbf{w}^\star)\right)$. While unrolling the recursion all the way down to 2 does not help, we notice that for any $\mathbf{w} \in \mathcal{W}$, $L(\mathbf{w}) - L(\mathbf{w}^\star) \leq 12HD^2 + 3L(\mathbf{w}^\star)$. Hence we unroll the recursion to M steps and use this inequality for the remaining sum. Optimizing over number of steps up to which we unroll and also optimizing over the choice of γ, we get the bound,

$$\mathbb{E}\left[L(\mathbf{w}_n^{\mathrm{ag}})\right] - L(\mathbf{w}^\star) \leq \sqrt{\tfrac{1648HD^2L(\mathbf{w}^\star)}{b(n-1)}} + \tfrac{348(6HD^2+2L(\mathbf{w}^\star))}{b(n-1)}(b(n-1))^{\frac{p}{p+1}} + \tfrac{32HD^2}{b(n-1)} + \tfrac{4HD^2}{(n-1)^{p+1}} + \tfrac{36HD^2}{b(n-1)} \tfrac{\log(n)}{(b(n-1))^{\frac{p}{2p+1}}}$$

Using the p as given in the theorem statement, and few simple manipulations, gives the final bound. □

4 Optimizing with Mini-Batches

To compare our two theorems and understand their implications, it will be convenient to treat H and D as constants, and focus on the more interesting parameters of sample size m, minibatch size b, and optimal expected loss $L(\mathbf{w}^\star)$. Also, we will ignore the logarithmic factor in Theorem 2, since we will mostly be interested in significant (i.e. polynomial) differences between the two algorithms, and it is quite possible that this logarithmic factor is merely an artifact of our analysis. Using $m = nb$, we get that the bound for the SGD algorithm is

$$\mathbb{E}\left[L(\bar{\mathbf{w}})\right] - L(\mathbf{w}^\star) \;\leq\; \tilde{\mathcal{O}}\left(\sqrt{\frac{L(\mathbf{w}^\star)}{bn}} + \frac{1}{n}\right) \;=\; \tilde{\mathcal{O}}\left(\sqrt{\frac{L(\mathbf{w}^\star)}{m}} + \frac{b}{m}\right), \tag{5}$$

and the bound for the accelerated gradient method we propose is

$$\mathbb{E}\left[L(\mathbf{w}_n^{\mathrm{ag}})\right] - L(\mathbf{w}^\star) \;\leq\; \tilde{\mathcal{O}}\left(\sqrt{\frac{L(\mathbf{w}^\star)}{bn}} + \frac{1}{\sqrt{b}n} + \frac{1}{n^2}\right) \;=\; \tilde{\mathcal{O}}\left(\sqrt{\frac{L(\mathbf{w}^\star)}{m}} + \frac{\sqrt{b}}{m} + \frac{b^2}{m^2}\right). \tag{6}$$

To understand the implication these bounds, we follow the approach described in [3, 12] to analyze large-scale learning algorithms. First, we fix a desired suboptimality parameter ϵ, which measures how close to $L(\mathbf{w}^\star)$ we want to get. Then, we assume that both algorithms are ran till the suboptimality of their outputs is at most ϵ. Our goal would be to understand the *runtime* each algorithm needs, till attaining suboptimality ϵ, as a function of $L(\mathbf{w}^\star), \epsilon, b$.

To measure this runtime, we need to discern two settings here: a *parallel* setting, where we assume that the mini-batch gradient computations are performed in parallel, and a *serial* setting, where the gradient computations are performed one after the other. In a parallel setting, we can take the number of iterations n as a rough measure of the runtime (note that in both algorithms, the runtime of a single iteration is comparable). In a serial setting, the relevant parameter is m, the number of data accesses.

To analyze the dependence on m and n, we upper bound (5) and (6) by ϵ, and invert them to get the bounds on m and n. Ignoring logarithmic factors, for the SGD algorithm we get

$$n \leq \frac{1}{\epsilon}\left(\frac{L(\mathbf{w}^\star)}{\epsilon} \cdot \frac{1}{b} + 1\right) \qquad m \leq \frac{1}{\epsilon}\left(\frac{L(\mathbf{w}^\star)}{\epsilon} + b\right), \tag{7}$$

and for the AG algorithm we get

$$n \leq \frac{1}{\epsilon}\left(\frac{L(\mathbf{w}^\star)}{\epsilon}\cdot\frac{1}{b}+\frac{1}{\sqrt{b}}+\sqrt{\epsilon}\right) \qquad m \leq \frac{1}{\epsilon}\left(\frac{L(\mathbf{w}^\star)}{\epsilon}+\sqrt{b}+b\sqrt{\epsilon}\right). \tag{8}$$

First, let us compare the performance of these two algorithms in the parallel setting, where the relevant parameter to measure runtime is n. Analyzing which of the terms in each bound dominates, we get that for the SGD algorithm, there are 2 regimes, while for the AG algorithm, there are 2-3 regimes depending on the relationship between $L(\mathbf{w}^\star)$ and ϵ. The following two tables summarize the situation (again, ignoring constants):

SGD Algorithm

Regime	n
$b \leq \sqrt{L(\mathbf{w}^\star)m}$	$\frac{L(\mathbf{w}^\star)}{\epsilon^2 b}$
$b \geq \sqrt{L(\mathbf{w}^\star)m}$	$\frac{1}{\epsilon}$

AG Algorithm

	Regime	n
$\epsilon \leq L(\mathbf{w}^\star)^2$	$b \leq L(\mathbf{w}^\star)^{1/4}m^{3/4}$	$\frac{L(\mathbf{w}^\star)}{\epsilon^2 b}$
	$b \geq L(\mathbf{w}^\star)^{1/4}m^{3/4}$	$\frac{1}{\sqrt{\epsilon}}$
$\epsilon \geq L(\mathbf{w}^\star)^2$	$b \leq L(\mathbf{w}^\star)m$	$\frac{L(\mathbf{w}^\star)}{\epsilon^2 b}$
	$L(\mathbf{w}^\star)m \leq b \leq m^{2/3}$	$\frac{1}{\epsilon\sqrt{b}}$
	$b \geq m^{2/3}$	$\frac{1}{\sqrt{\epsilon}}$

From the tables, we see that for both methods, there is an initial linear speedup as a function of the minibatch size b. However, in the AG algorithm, this linear speedup regime holds for much larger minibatch sizes[1]. Even beyond the linear speedup regime, the AG algorithm still maintains a $\sqrt{b}$ speedup, for the reasonable case where $\epsilon \geq L(\mathbf{w}^\star)^2$. Finally, in all regimes, the runtime bound of the AG algorithm is equal or significantly smaller than that of the SGD algorithm.

We now turn to discuss the serial setting, where the runtime is measured in terms of m. Inspecting (7) and (8), we see that a larger size of b actually requires m to increase for both algorithms. This is to be expected, since mini-batching does not lead to large gains in a serial setting. However, using mini-batching in a serial setting might still be beneficial for implementation reasons, resulting in constant-factor improvements in runtime (e.g. saving overhead and loop control, and via pipelining, concurrent memory accesses etc.). In that case, we can at least ask what is the largest mini-batch size that won't degrade the runtime guarantee by more than a constant. Using our bounds, the mini-batch size b for the SGD algorithm can scale as much as L/ϵ, vs. a larger value of $L/\epsilon^{3/2}$ for the AG algorithm.

Finally, an interesting point is that the AG algorithm is sometimes actually *necessary* to obtain significant speed-ups via a mini-batch framework (according to our bounds). Based on the table above, this happens when the desired suboptimality ϵ is not much bigger then $L(\mathbf{w}^\star)$, i.e. $\epsilon = \Omega(L(\mathbf{w}^\star))$. This includes the "separable" case, $L(\mathbf{w}^\star) = 0$, and in general a regime where the "estimation error" ϵ and "approximation error" $L(\mathbf{w}^\star)$ are roughly the same—an arguably very relevant one in machine learning. For the SGD algorithm, the critical mini-batch value $\sqrt{L(\mathbf{w}^\star)m}$ can be shown to equal $L(\mathbf{w}^\star)/\epsilon$, which is $O(1)$ in our case. So with SGD we get no non-constant parallel speedup. However, with AG, we still enjoy a speedup of at least $\Theta(\sqrt{b})$, all the way up to mini-batch size $b = m^{2/3}$.

5 Experiments

We implemented both the SGD algorithm (Algorithm 1) and the AG algorithm (Algorithm 2, using step-sizes of the form $\gamma_i = \gamma i^p$ as suggested by Theorem 2) on two publicly-available binary classification problems, astro-physics and CCAT. We used the smoothed hinge loss $\ell(\mathbf{w}; \mathbf{x}, y)$, defined as $0.5 - y\mathbf{w}^\top\mathbf{x}$ if $y\mathbf{w}^\top\mathbf{x} \leq 0$; 0 if $y\mathbf{w}^\top\mathbf{x} > 1$, and $0.5(1-y\mathbf{w}^\top\mathbf{x})^2$ otherwise.

While both datasets are relatively easy to classify, we also wished to understand the algorithms' performance in the "separable" case $L(\mathbf{w}^\star) = 0$, to see if the theory in Section 4

[1]Since it is easily verified that $\sqrt{L(\mathbf{w}^\star)m}$ is generally smaller than both $L(\mathbf{w}^\star)^{1/4}m^{3/4}$ and $L(\mathbf{w}^\star)m$

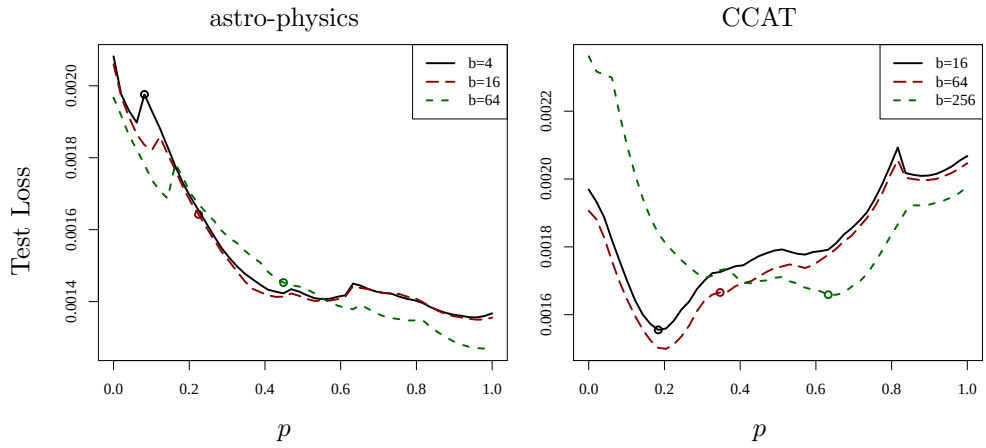

Figure 1: Left: Test smoothed hinge loss, as a function of p, after training using the AG algorithm on 6361 examples from astro-physics, for various batch sizes. Right: the same, for 18578 examples from CCAT. In both datasets, margin violations were removed before training so that $L(\mathbf{w}^\star) = 0$. The circled points are the theoretically-derived values $p = \ln b/(2\ln(n-1))$ (see Theorem 2).

holds in practice. To this end, we created an additional version of each dataset, where $L(\mathbf{w}^\star) = 0$, by training a classifier on the entire dataset and removing margin violations.

In all of our experiments, we used up to half of the data for training, and one-quarter each for validation and testing. The validation set was used to determine the step sizes η and γ_i. We justify this by noting that our goal is to compare the performance of the SGD and AG algorithms, independently of the difficulties in choosing their stepsizes. In the implementation, we neglected the projection step, as we found it does not significantly affect performance when the stepsizes are properly selected.

In our first set of experiments, we attempted to determine the relationship between the performance of the AG algorithm and the p parameter, which determines the rate of increase of the step sizes γ_i. Our experiments are summarized in Figure 5. Perhaps the most important conclusion to draw from these plots is that neither the "traditional" choice $p = 1$, nor the constant-step-size choice $p = 0$, give the best performance in all circumstances. Instead, there is a complicated data-dependent relationship between p, and the final classifier's performance. Furthermore, there appears to be a weak trend towards higher p performing better for larger minibatch sizes b, which corresponds neatly with our theoretical predictions.

In our next experiment, we directly compared the performance of the SGD and AG. To do so, we varied the minibatch size b while holding the total amount of training data ($m = nb$) fixed. When $L(\mathbf{w}^\star) > 0$ (top row of Figure 5), the total sample size m is high and the suboptimality ϵ is low (red and black plots), we see that for small minibatch size, both methods do not degrade as we increase b, corresponding to a linear parallel speedup. In fact, SGD is actually overall better, but as b increases, its performance degrades more quickly, eventually performing worse than AG. That is, even in the least favorable scenario for AG (high $L(\mathbf{w}^\star)$ and small ϵ, see the tables in Sec. 4), it does give benefits with large enough minibatch sizes. Further, we see that once the suboptimality ϵ is roughly equal to L^*, AG significantly outperforms SGD, even with small minibatches, agreeing with theory.

Turning to the case $L(\mathbf{w}^\star) = 0$ (bottom two rows of Figure 5), which is theoretically more favorable to AG, we see it is indeed mostly better, in terms of retaining linear parallel speedups for larger minibatch sizes, even for large data set sizes corresponding to small suboptimality values, and might even be advantageous with small minibatch sizes.

6 Summary

In this paper, we presented novel contributions to the theory of first order stochastic convex optimization (Theorems 1 and 2, generalizing results of [4] and [5] to be sensitive to $L(\mathbf{w}^\star)$),

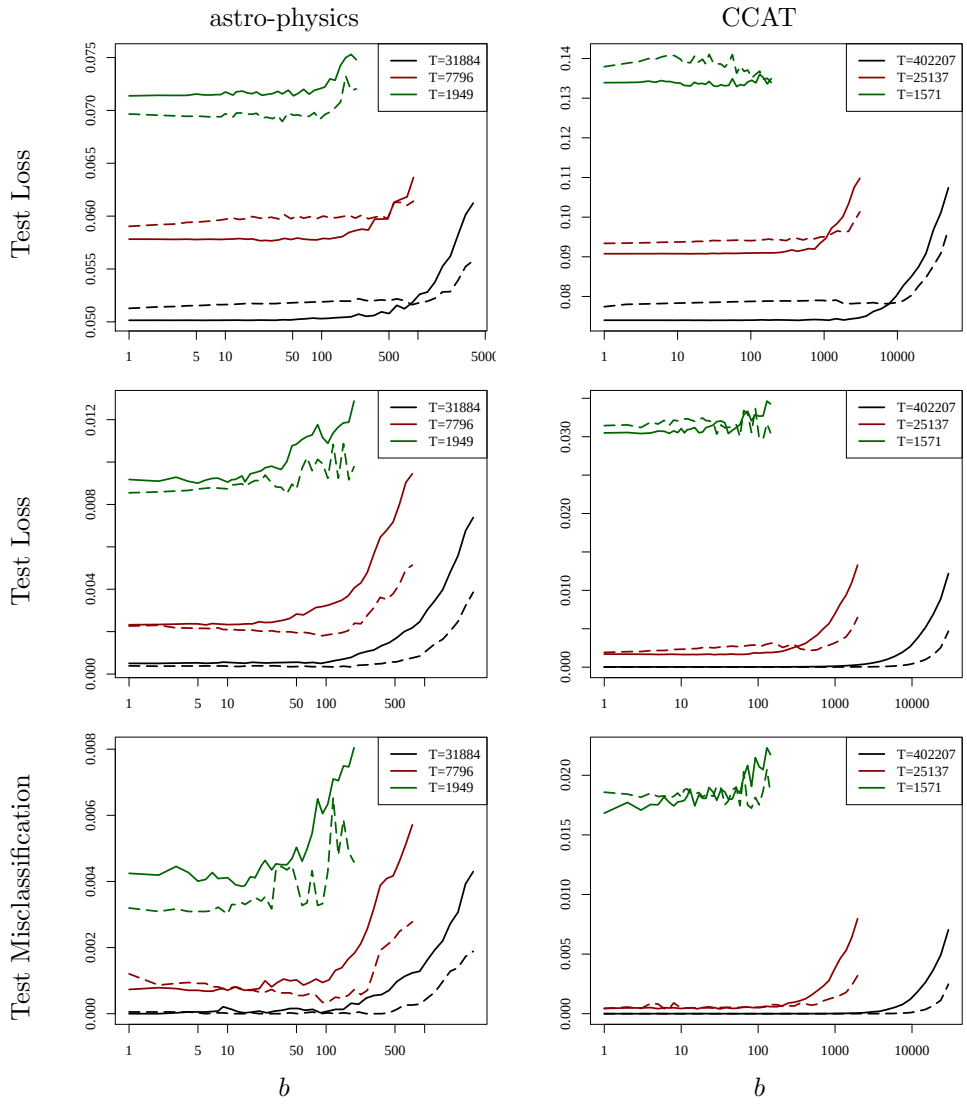

Figure 2: Test loss on astro-physics and CCAT as a function of mini-batch size b (in log-scale), where the total amount of training data $m = nb$ is held fixed. Solid lines and dashed lines are for SGD and AG respectively (for AG, we used $p = \ln b/(2\ln(n-1))$ as in Theorem 2). The upper row shows the smoothed hinge loss on the test set, using the original (uncensored) data. The bottom rows show the smoothed hinge loss and misclassification rate on the test set, using the modified data where $L(\mathbf{w}^\star) = 0$. All curves are averaged over three runs.

developed a novel step size strategy for the accelerated method that we used in order to obtain our results and we saw works well in practice, and provided a more refined analysis of the effects of minibatching which paints a different picture then previous analyses [4, 1] and highlights the benefit of accelerated methods.

A remaining open practical and theoretical question is whether the bound of Theorem 2 is tight. Following [5], the bound is tight for $b = 1$ and $b \to \infty$, i.e. the first and third terms are tight, but it is not clear whether the $1/(\sqrt{b}n)$ dependence is indeed necessary. It would be interesting to understand whether with a more refined analysis, or perhaps different step-sizes, we can avoid this term, whether an altogether different algorithm is needed, or whether this term does represent the optimal behavior for any method based on b-aggregated stochastic gradient estimates.

References

[1] A. Agarwal and J. Duchi. Distributed delayed stochastic optimization. Technical report, arXiv, 2011.

[2] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167 – 175, 2003.

[3] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2007.

[4] O. Dekel, R. Gilad Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. Technical report, arXiv, 2010.

[5] G. Lan. An optimal method for stochastic convex optimization. Technical report, Georgia Institute of Technology, 2009.

[6] 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.

[7] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. *Doklady AN SSSR*, 269:543–547, 1983.

[8] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2005.

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

[10] O. Shamir O. Dekel, R. Gilad-Bachrach and L. Xiao. Optimal distributed online prediction. In *ICML*, 2011.

[11] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: primal estimated sub-gradient solver for SVM. *Math. Program.*, 127(1):3–30, 2011.

[12] S. Shalev-Shwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In *ICML*, 2008.

[13] N. Srebro, K. Sridharan, and A. Tewari. Smoothness, low noise and fast rates. In *NIPS*, 2010.

[14] S.Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, Hebrew University of Jerusalem, 2007.

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

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