8-30-2016

Batched Stochastic Gradient Descent with Weighted Sampling

Deanna Needell
Claremont McKenna College

Rachel Ward
University of Texas at Austin

Recommended Citation

D. Needell and R. Ward. "Batched Stochastic Gradient Descent with Weighted Sampling." 2016.

BATCHED STOCHASTIC GRADIENT DESCENT WITH WEIGHTED SAMPLING

DEANNA NEEDELL AND RACHEL WARD

ABSTRACT. We analyze a batched variant of Stochastic Gradient Descent (SGD) with weighted sampling distribution for smooth and non-smooth objective functions. We show that by distributing the batches computationally, a significant speedup in the convergence rate is provably possible compared to either batched sampling or weighted sampling alone. We propose several computationally efficient schemes to approximate the optimal weights, and compute proposed sampling distributions explicitly for the least squares and hinge loss problems. We show both analytically and experimentally that substantial gains can be obtained.

1. MATHEMATICAL FORMULATION

We consider minimizing an objective function of the form

$$F(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}) = \mathbb{E} f_i(\boldsymbol{x}). \tag{1.1}$$

One important such objective function is the least squares objective for linear systems. Given an $n \times m$ matrix $\boldsymbol{A}$ with rows $\boldsymbol{a}_1, \dots, \boldsymbol{a}_n$ and a vector $\boldsymbol{b} \in \mathbb{R}^n$, one searches for the least squares solution $\boldsymbol{x}_{LS}$ given by

$$\boldsymbol{x}_{LS} \stackrel{\text{def}}{=} \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^m} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^m} \frac{1}{n}\sum_{i=1}^{n}\frac{n}{2}(\boldsymbol{b}_i - \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle)^2 = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^m} \mathbb{E} f_i(\boldsymbol{x}), \tag{1.2}$$

where the functionals are defined by $f_i(\boldsymbol{x}) = \frac{n}{2}(\boldsymbol{b}_i - \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle)^2$.

Another important example is the setting of support vector machines where one wishes to minimize the hinge loss objective given by

$$\boldsymbol{x}_{HL} \stackrel{\text{def}}{=} \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^m} \frac{1}{n}\sum_{i=1}^{n}[1 - y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle]_+ + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2. \tag{1.3}$$

Here, the data is given by the matrix $\boldsymbol{X}$ with rows $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ and the labels $y_i \in \{-1, 1\}$. The function $[z]_+ \stackrel{\text{def}}{=} \max(0, z)$ denotes the positive part. We view the problem (1.3) in the form (1.1) with $f_i(\boldsymbol{w}) = [1 - y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle]_+$ and regularizer $\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$.

The stochastic gradient descent (SGD) method solves problems of the form (1.1) by iteratively moving in the gradient direction of a randomly selected functional. SGD can be described succinctly by the update rule:

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \gamma \nabla f_{i_k}(\boldsymbol{x}_k),$$

where index i_k is selected randomly in the kth iteration, and an initial estimation $\boldsymbol{x}_0$ is chosen arbitrarily. Typical implementations of SGD select the functionals uniformly at random, although if the problem at hand allows a one-pass preprocessing of the functionals, certain *weighted* sampling distributions preferring functionals with larger variation can provide better convergence (see e.g. [NSW16, ZZ15] and references therein). In particular, Needell et al. show that selecting a functional with probability proportional to the Lipschitz constant of its gradient yields a convergence rate depending on the *average* of all such Lipschitz constants, rather than the supremum [NSW16]. An analogous result in the same work shows that for non-smooth functionals, the probabilities should be chosen proportional to the Lipschitz constant of the functional itself.

Date: August 30, 2016.

Another variant of SGD utilizes so-called *mini-batches*; in this variant, a batch of functionals is selected in each iteration rather than a single one [CSSS11, AD11, DGBSX12, TBRS13]. The computations over the batches can then be run in parallel and speedups in the convergence are often quite significant.

Contribution. Our main contribution is to propose a weighted sampling scheme to be used in mini-batch SGD. We show that when the batches can be implemented in parallel, significant speedup in convergence is possible. In particular, we analyze the convergence using computed distributions for the least squares and hinge loss objectives, the latter being especially challenging since it is non-smooth. We demonstrate theoretically and empirically that weighting the distribution and utilizing batches of functionals per iteration together form a complementary approach to accelerating convergence.

Organization. We next briefly discuss some related work on SGD, weighted distributions, and batching methods. We then combine these ideas into one cohesive framework and discuss the benefits in various settings. Section 2 focuses on the impact of weighting the distribution. In Section 3 we analyze SGD with weighting and batches for smooth objective functions, considering the least squares objective as a motivating example. We analyze the non-smooth case along with the hinge loss objective function in Section 4. We display experimental results for the least squares problem in Section 5 that serve to highlight the relative tradeoffs of using both batches and weighting, along with different computational approaches. We conclude in Section 6.

Related work. Stochastic gradient descent, stemming from the work [RM51], has recently received renewed attention for its effectiveness in treating large-scale problems arising in machine learning [BB11, Bot10, NJLS09, SSS08]. Importance sampling in stochastic gradient descent, as in the case of mini-batching (which we also refer to simply as *batching* here), also leads to variance reduction in stochastic gradient methods and, in terms of theory, leads to improvement of the leading constant in the complexity estimate, typically via replacing the maximum of certain data-dependent quantities by their average. Such theoretical guarantees were shown for the case of solving least squares problems where stochastic gradient descent coincides with the randomized Kaczmarz method in [SV09]. This method was extended to handle noisy linear systems in [Nee10]. Later, this strategy was extended to the more general setting of smooth and strongly convex objectives in [NSW16], building on an analysis of stochastic gradient descent in [BM11]. Later, [ZZ15] considered a similar importance sampling strategy for convex but not necessarily smooth objective functions. Importance sampling has also been considered in the related setting of stochastic coordinate descent/ascent methods [Nes12, RT15, QRZ15, CQR15]. Other papers exploring advantages of importance sampling in various adaptations of stochastic gradient descent include but are not limited to [LS13, SRB13, XZ14, DB15].

Mini-batching in stochastic gradient methods refers to pooling together several random examples in the estimate of the gradient, as opposed to just a single random example at a time, effectively reducing the variance of each iteration [SSSSC11]. On the other hand, each iteration also increases in complexity as the size of the batch grows. However, if parallel processing is available, the computation can be done concurrently at each step, so that the “per-iteration cost" with batching is not higher than without batching. Ideally, one would like the consequence of using batch size b to result in a convergence rate speed-up by factor of b, but this is not always the case [BCNW12]. Still, [TBRS13] showed that by incorporating parallelization or multiple cores, this strategy can only improve on the convergence rate over standard stochastic gradient, and can improve the convergence rate by a factor of the batch size in certain situations, such as when the matrix has nearly orthonormal rows. Other recent papers exploring the advantages of mini-batching in different settings of stochastic optimization include [CSSS11, DGBSX12, NW13, KLRT16, LZCS14].

The recent paper [CR16] also considered the combination of importance sampling and mini-batching for a stochastic dual coordinate ascent algorithm in the general setting of empirical risk minimization, wherein the function to minimize is smooth and convex. There the authors provide a theoretical optimal sampling strategy that is not practical to implement but can be approximated via alternating minimization. They also provide a computationally efficient formula that yields better sample complexity than uniform mini-batching, but without quantitative bounds on the gain. In particular, they do not provide general assumptions under which one achieves provable speed-up in convergence depending on an average Lipschitz constant rather than a maximum.

For an overview of applications of stochastic gradient descent and its weighted/batched variants in large-scale matrix inversion problems, we refer the reader to [GR16].

2. SGD WITH WEIGHTING

Recall the objective function (1.1). We assume in this section that the function F and the functionals f_i satisfy the following convexity and smoothness conditions:

Convexity and smoothness conditions

(1) Each f_i is continuously differentiable and the gradient function ∇f_i has Lipschitz constant bounded by L_i: $\|\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y})\|_2 \le L_i \|\boldsymbol{x} - \boldsymbol{y}\|_2$ for all vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

(2) F has strong convexity parameter μ; that is, $\langle \boldsymbol{x} - \boldsymbol{y}, \nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})\rangle \ge \mu \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$ for all vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

(3) At the unique minimizer $\boldsymbol{x}_* = \operatorname{argmin} F(\boldsymbol{x})$, the average gradient norm squared $\|\nabla f_i(\boldsymbol{x}_*)\|_2^2$ is not too large, in the sense that
$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(\boldsymbol{x}_*)\|_2^2 \le \sigma^2.$$

An unbiased gradient estimate for $F(\boldsymbol{x})$ can be obtained by drawing i uniformly from $[n] \overset{\text{def}}{=} \{1, 2, \ldots, n\}$ and using $\nabla f_i(\boldsymbol{x})$ as the estimate for $\nabla F(\boldsymbol{x})$. The standard SGD update with fixed step size γ is given by

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \gamma \nabla f_{i_k}(\boldsymbol{x}_k) \tag{2.1}$$

where each i_k is drawn uniformly from $[n]$. The idea behind weighted sampling is that, by drawing i from a weighted distribution $\mathcal{D}^{(p)} = \{p(1), p(2), \ldots, p(n)\}$ over $[n]$, the weighted sample $\frac{1}{p(i_k)}\nabla f_{i_k}(\boldsymbol{x}_k)$ is still an unbiased estimate of the gradient $\nabla F(\boldsymbol{x})$. This motivates the weighted SGD update

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \frac{\gamma}{np(i_k)} \nabla f_{i_k}(\boldsymbol{x}_k), \tag{2.2}$$

In [NSW16], a family of distributions $\mathcal{D}^{(p)}$ whereby functions f_i with larger Lipschitz constants are more likely to be sampled was shown to lead to an improved convergence rate in SGD over uniform sampling. In terms of the distance $\|\boldsymbol{x}_k - \boldsymbol{x}_*\|_2^2$ of the kth iterate to the unique minimum, starting from initial distance $\varepsilon_0 = \|\boldsymbol{x}_0 - \boldsymbol{x}_*\|_2^2$, Corollary 3.1 in [NSW16] is as follows.

Proposition 2.1. *Assume the convexity and smoothness conditions are in force. For any desired $\varepsilon > 0$, and using a stepsize of*
$$\gamma = \frac{\mu\varepsilon}{4(\varepsilon\mu\frac{1}{n}\sum_{i=1}^{n} L_i + \sigma^2)},$$
we have that after
$$k = 4\log(2\varepsilon_0/\varepsilon)\left(\frac{\frac{1}{n}\sum_{i=1}^{n} L_i}{\mu} + \frac{\sigma^2}{\mu^2\varepsilon}\right) \tag{2.3}$$
iterations of weighted SGD (2.2) *with weights*
$$p(i) = \frac{1}{2n} + \frac{1}{2n}\cdot\frac{L_i}{\frac{1}{n}\sum_i L_i}, \tag{2.4}$$
the following holds in expectation with respect to the weighted distribution (2.4)*:* $\mathbb{E}^{(p)}\|\mathbf{x}_k - \mathbf{x}_*\|_2^2 \le \varepsilon$.

Remark. This should be compared to the result for uniform sampling SGD [NSW16]: using step-size $\gamma = \frac{\mu\varepsilon}{4(\varepsilon\mu(\sup_i L_i)+\sigma^2)}$, one obtains the comparable error guarantee $\mathbb{E}\|\mathbf{x}_k - \mathbf{x}_*\|_2^2 \le \varepsilon$ after a number of iterations

$$k = 2\log(2\varepsilon_0/\varepsilon)\left(\frac{\sup_i L_i}{\mu} + \frac{\sigma^2}{\mu^2\varepsilon}\right). \tag{2.5}$$

Since the average Lipschitz constant $\frac{1}{n}\sum_i L_i$ is always at most $\sup_i L_i$, and can be up to n times smaller than $\sup_i L_i$, SGD with weighted sampling requires twice the number of iterations of uniform SGD in the worst case, but can potentially converge much faster, specifically, in the regime where

$$\frac{\sigma^2}{\mu^2\varepsilon} \le \frac{\frac{1}{n}\sum_{i=1}^n L_i}{\mu} \ll \frac{\sup_i L_i}{\mu}.$$

3. Mini-batch SGD with weighting: the smooth case

Here we present a weighting and mini-batch scheme for SGD based on Proposition 2.1. For practical purposes, we assume that the functions $f_i(\boldsymbol{x})$ such that $F(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{x})$ are initially partitioned into fixed batches of size b and denote the partition by $\{\tau_1, \tau_2, \dots \tau_d\}$ where $|\tau_i| = b$ for all $i < d$ and $d = \lceil n/b \rceil$ (for simplicity we will henceforth assume that $d = n/b$ is an integer). We will randomly select from this pre-determined partition of batches; however, our analysis extends easily to the case where a batch of size b is randomly selected each time from the entire set of functionals. With this notation, we may re-formulate the objective given in (1.1) as follows:

$$F(\boldsymbol{x}) = \frac{1}{d}\sum_{i=1}^d g_{\tau_i}(\boldsymbol{x}) = \mathbb{E} g_{\tau_i}(\boldsymbol{x}),$$

where now we write $g_{\tau_i}(\boldsymbol{x}) = \frac{1}{b}\sum_{j\in\tau_i} f_j(\boldsymbol{x})$. We can apply Proposition 2.1 to the functionals g_{τ_i}, and select batch τ_i with probability proportional the Lipschitz constant of ∇g_{τ_i} (or of g_{τ_i} in the non-smooth case, see Section 4). Note that

- The strong convexity parameter μ for the function F remains invariant to the batching rule.
- The residual error σ_τ^2 such that $\frac{1}{d}\sum_{i=1}^d \|\nabla g_{\tau_i}(x_*)\|_2^2 \le \sigma_\tau^2$ can only **decrease** with increasing batch size, since
$$\sigma_\tau^2 = \frac{1}{d}\sum_{k=1}^d \|\frac{1}{b}\nabla\left(\sum_{k\in\tau_i} f_k(\boldsymbol{x})\right)\|_2^2 \le \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(\boldsymbol{x})\|_2^2 \le \sigma^2.$$
- The average Lipschitz constant $\overline{L}_\tau = \frac{1}{d}\sum_{i=1}^d L_{\tau_i}$ of the gradients of the batched functions g_{τ_i} can only **decrease** with increasing batch size, since by the triangle inequality, $L_{\tau_i} \le \frac{1}{b}\sum_{k\in\tau_i} L_k$, and thus
$$\frac{1}{d}\sum_{i=1}^d L_{\tau_i} \le \frac{1}{n}\sum_{k=1}^n L_k = \overline{L}.$$

Incorporating these observations, applying Proposition 2.1 in the batched weighted setting implies that incorporating weighted sampling and mini-batching in SGD results in a convergence rate that equals or improves on the rate obtained using weights alone:

Theorem 3.1. *Assume that the convexity and smoothness conditions on $F(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{x})$ are in force. Consider the $d = n/b$ batches $g_{\tau_i}(\boldsymbol{x}) = \frac{1}{b}\sum_{k\in\tau_i} f_k(\boldsymbol{x})$, and the batched weighted SGD iteration*

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \frac{\gamma}{d\cdot p(\tau_{i_k})}\nabla g_{\tau_{i_k}}(\boldsymbol{x}_k)$$

where batch τ_i is selected at iteration k with probability

$$p(\tau_i) = \frac{1}{2d} + \frac{1}{2d}\cdot\frac{L_{\tau_i}}{\overline{L}_\tau}. \tag{3.1}$$

For any desired ε, and using a stepsize of

$$\gamma = \frac{\mu\varepsilon}{4(\varepsilon\mu\overline{L}_\tau + \sigma_\tau^2)},$$

we have that after a number of iterations

$$k = 4\log(2\varepsilon_0/\varepsilon)\left(\frac{\overline{L}_\tau}{\mu} + \frac{\sigma_\tau^2}{\mu^2\varepsilon}\right),$$

the following holds in expectation with respect to the weighted distribution (3.1)*:* $\mathbb{E}^{(p)}\|\mathbf{x}_k - \mathbf{x}_*\|_2^2 \le \varepsilon$.

Remark. Since

$$\left(\frac{\overline{L}_\tau}{\mu} + \frac{\sigma_\tau^2}{\mu^2\varepsilon}\right) \le \left(\frac{\overline{L}}{\mu} + \frac{\sigma^2}{\mu^2\varepsilon}\right),$$

this implies that batching and weighting can only improve the convergence rate of SGD compared to weighting alone.

To completely justify the strategy of batching + weighting, we must also take into account the precomputation cost in computing the weighted distribution (3.1), which increases with the batch size b. In the next section, we refine Theorem 3.1 precisely this way in the case of the least squares objective, where we can quantify more precisely the gain achieved by weighting and batching. We give several explicit bounds and sampling strategies on the Lipschitz constants in this case that can be used for computationally efficient sampling.

3.1. **Least Squares Objective.**

Consider the least squares objective

$$F(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}),$$

where $f_i(\boldsymbol{x}) = \frac{n}{2}(b_i - \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle)^2$. We assume the matrix $\boldsymbol{A}$ has full row-rank, so that there is a unique minimizer $\boldsymbol{x}_*$ to the least squares problem:

$$\boldsymbol{x}_{LS} = \boldsymbol{x}_* = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2.$$

Note that the convexity and smoothness conditions are satisfied for such functions. Indeed, observe that $\nabla f_i(\boldsymbol{x}) = n(\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle - b_i)\boldsymbol{a}_i$, and

(1) The individual Lipschitz constants are bounded by $L_i = n\|\boldsymbol{a}_i\|_2^2$, and the average Lipschitz constant by $\frac{1}{n}\sum_i L_i = \|\boldsymbol{A}\|_F^2$ (where $\|\cdot\|_F$ denotes the Frobenius norm),
(2) The strong convexity parameter is $\mu = \frac{1}{\|\boldsymbol{A}^{-1}\|^2}$ (where $\|\boldsymbol{A}^{-1}\| = \sigma_{\min}^{-1}(\boldsymbol{A})$ is the reciprocal of the smallest singular value of $\boldsymbol{A}$),
(3) The residual is $\sigma^2 = n\sum_i \|\boldsymbol{a}_i\|_2^2|\langle \boldsymbol{a}_i, \boldsymbol{x}_*\rangle - \boldsymbol{a}_i|^2$.

In the batched setting, we compute

$$g_{\tau_i}(\boldsymbol{x}) = \frac{1}{b}\sum_{k\in\tau_i} f_k(\boldsymbol{x}) = \frac{n}{2b}\sum_{k\in\tau_i}(b_k - \langle \boldsymbol{a}_k, \boldsymbol{x}\rangle)^2 = \frac{d}{2}\|\boldsymbol{A}_{\tau_i}\boldsymbol{x} - \boldsymbol{b}_{\tau_i}\|_2^2, \tag{3.2}$$

where we have written $\boldsymbol{A}_{\tau_i}$ to denote the submatrix of $\boldsymbol{A}$ consisting of the rows indexed by τ_i.

Denote by σ_τ^2 the residual in the batched setting. Since $\nabla g_{\tau_i} = d\sum_{k\in\tau_i}(\langle \boldsymbol{a}_k, \boldsymbol{x}\rangle - b_k)\boldsymbol{a}_k$,

$$\begin{aligned}\sigma_\tau^2 = \frac{1}{d}\sum_{i=1}^{d}\|\nabla g_{\tau_i}(\boldsymbol{x}_*)\|_2^2 &= d\sum_{i=1}^{d}\|\sum_{k\in\tau_i}(\langle \boldsymbol{a}_k, \boldsymbol{x}_*\rangle - b_k)\boldsymbol{a}_k\|_2^2 \\ &= d\sum_{i=1}^{d}\|\boldsymbol{A}_{\tau_i}^*(\boldsymbol{A}_{\tau_i}\boldsymbol{x}_* - \boldsymbol{b}_{\tau_i})\|_2^2 \le d\sum_{i=1}^{d}\|\boldsymbol{A}_{\tau_i}\|^2\|\boldsymbol{A}_{\tau_i}\boldsymbol{x}_* - \boldsymbol{b}_{\tau_i}\|_2^2.\end{aligned}$$

Denote by L_{τ_i} the Lipschitz constant of ∇g_{τ_i}. Then we also have

$$\begin{aligned}L_{\tau_i} &= \sup_{\boldsymbol{x},\boldsymbol{y}} \frac{\|\nabla g_{\tau_i}(\boldsymbol{x}) - \nabla g_{\tau_i}(\boldsymbol{y})\|_2}{\|\boldsymbol{x}-\boldsymbol{y}\|_2}\\ &= \frac{n}{b}\sup_{\boldsymbol{x},\boldsymbol{y}} \frac{\|\sum_{k\in\tau_i}\left[(\langle \boldsymbol{a}_k,\boldsymbol{x}\rangle - b_k)\boldsymbol{a}_k - (\langle \boldsymbol{a}_k,\boldsymbol{y}\rangle - b_k)\boldsymbol{a}_k\right]\|_2}{\|\boldsymbol{x}-\boldsymbol{y}\|_2}\\ &= \frac{n}{b}\sup_{\boldsymbol{z}} \frac{\|\sum_{k\in\tau_i}\langle \boldsymbol{a}_k,\boldsymbol{z}\rangle \boldsymbol{a}_k\|_2}{\|\boldsymbol{z}\|_2}\\ &= \frac{n}{b}\sup_{\boldsymbol{z}} \frac{\|\boldsymbol{A}_{\tau_i}^*\boldsymbol{A}_{\tau_i}\boldsymbol{z}\|_2}{\|\boldsymbol{z}\|_2}\\ &= \frac{n}{b}\|\boldsymbol{A}_{\tau_i}^*\boldsymbol{A}_{\tau_i}\|\\ &= d\|\boldsymbol{A}_{\tau_i}\|_2^2,\end{aligned}$$

where we have written $\|\boldsymbol{B}\|$ to denote the spectral norm of the matrix $\boldsymbol{B}$, and $\boldsymbol{B}^*$ the adjoint of the matrix.

We see thus that *if there exists a partition such that $\|\boldsymbol{A}_{\tau_i}\|$ are as small as possible for all τ_i in the partition, then both σ_τ^2 and $L_\tau = \frac{1}{d}\sum_i L_{\tau_i}$ are decreased by a factor of the batch size b compared to the unbatched setting.* These observations are summed up in the following corollary of Theorem 3.1 for the least squares case.

Corollary 3.2. *Consider $F(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 = \frac{1}{2}\sum_{i=1}^d \|\boldsymbol{A}_{\tau_i}\boldsymbol{x}-\boldsymbol{b}_{\tau_i}\|_2^2$. Consider the batched weighted SGD iteration*

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \frac{\gamma}{p(\tau_i)}\sum_{j\in\tau_i}(\langle \boldsymbol{a}_j,\boldsymbol{x}_k\rangle - b_j)\boldsymbol{a}_j. \tag{3.3}$$

with weights

$$p(\tau_i) = \frac{b}{2n} + \frac{1}{2}\cdot\frac{\|\boldsymbol{A}_{\tau_i}\|^2}{\sum_{i=1}^d \|\boldsymbol{A}_{\tau_i}\|^2}. \tag{3.4}$$

For any desired ε, and using a stepsize of

$$\gamma = \frac{\frac{1}{4}\varepsilon}{\varepsilon\sum_{i=1}^d \|\boldsymbol{A}_{\tau_i}\|^2 + d\|\boldsymbol{A}^{-1}\|^2\sum_{i=1}^d \|\boldsymbol{A}_{\tau_i}\|^2\|\boldsymbol{A}_{\tau_i}\boldsymbol{x}_* - \boldsymbol{b}_{\tau_i}\|_2^2}, \tag{3.5}$$

we have that after

$$k = 4\log(2\varepsilon_0/\varepsilon)\left(\|\boldsymbol{A}^{-1}\|^2\sum_{i=1}^d \|\boldsymbol{A}_{\tau_i}\|^2 + \frac{d\|\boldsymbol{A}^{-1}\|^4\sum_{i=1}^d \|\boldsymbol{A}_{\tau_i}\|^2\|\boldsymbol{A}_{\tau_i}\boldsymbol{x}_* - \boldsymbol{b}_{\tau_i}\|_2^2}{\varepsilon}\right) \tag{3.6}$$

iterations of (3.3), $\mathbb{E}^{(p)}\|\mathbf{x}_k - \mathbf{x}_*\|_2^2 \le \varepsilon$ *where* $\mathbb{E}^{(p)}[\cdot]$ *means the expectation with respect to the index at each iteration drawn according to the weighted distribution* (3.4).

This corollary suggests a heuristic for batching and weighting in SGD for least squares problems, in order to optimize the convergence rate:

(1) Find a partition $\tau_1, \tau_2, \ldots, \tau_d$ that roughly minimizes $\sum_{i=1}^d \|\boldsymbol{A}_{\tau_i}\|_2^2$ among all such partitions
(2) Apply the weighted SGD algorithm (2.2) using weights

$$p(\tau_i) = \frac{1}{2d} + \frac{1}{2}\cdot\frac{\|\boldsymbol{A}_{\tau_i}\|_2^2}{\sum_{i=1}^d \|\boldsymbol{A}_{\tau_i}\|_2^2}.$$

We can compare the results of Corollary 3.2 to the results for weighted SGD when a single functional is selected in each iteration, where the number of iterations to achieve expected error ε is

$$k = 4\log(2\varepsilon_0/\varepsilon)\left(\|\boldsymbol{A}^{-1}\|^2\sum_{i=1}^n \|\boldsymbol{a}_i\|^2 + \frac{n\|\boldsymbol{A}^{-1}\|^4\sum_{i=1}^n \|\boldsymbol{a}_i\|^2\|\langle \boldsymbol{a}_i,\boldsymbol{x}_*\rangle - b_i\|_2^2}{\varepsilon}\right); \tag{3.7}$$

That is, the ratio between the standard weighted number of iterations k_{stand} in (3.7) and the batched weighted number of iterations k_{batch} in (3.6) is

$$\frac{k_{stand}}{k_{batch}} = \frac{\varepsilon \sum_{i=1}^{n} \|\boldsymbol{a}_i\|^2 + n\|\boldsymbol{A}^{-1}\|^2 \sum_{i=1}^{n} \|\boldsymbol{a}_i\|^2 \|\langle \boldsymbol{a}_i, \boldsymbol{x}_* \rangle - b_i\|_2^2}{\varepsilon \sum_{i=1}^{d} \|\boldsymbol{A}_{\tau_i}\|^2 + d\|\boldsymbol{A}^{-1}\|^2 \sum_{i=1}^{d} \|\boldsymbol{A}_{\tau_i}\|^2 \|\boldsymbol{A}_{\tau_i}\boldsymbol{x}_* - \boldsymbol{b}_{\tau_i}\|_2^2} \tag{3.8}$$

In case the least squares residual error is uniformly distributed over the n indices, that is, $\|\langle \boldsymbol{a}_i, \boldsymbol{x}_* \rangle - b_i\|_2^2 \approx \frac{1}{n}\|\boldsymbol{A}\boldsymbol{x}_* - \boldsymbol{b}\|^2$ for each $i \in [n]$, this factor reduces to

$$\frac{k_{stand}}{k_{batch}} = \frac{\|\boldsymbol{A}\|_F^2}{\sum_{i=1}^{d} \|\boldsymbol{A}_{\tau_i}\|^2} \tag{3.9}$$

It follows thus that the combination of batching and weighting in this setting always reduces the iteration complexity compared to weighting alone, and can result in up to a factor of b speed-up:

$$1 \le \frac{k_{stand}}{k_{batch}} \le b;$$

In the remainder of this section, we consider several families of matrices where the maximal speedup is achieved, $\frac{k_{stand}}{k_{batch}} \approx b$. We also take into account the computational cost of computing the norms $\|\boldsymbol{A}_{\tau_i}\|_2^2$ which determine the weighted sampling strategy.

Orthonormal systems: It is clear that the advantage of mini-batching is strongest when the rows of $\boldsymbol{A}$ in each batch are orthonormal. In the extreme case where $\boldsymbol{A}$ has orthonormal rows, we have

$$\overline{L}_\tau = \sum_{i=1}^{d} \|\boldsymbol{A}_{\tau_i}^* \boldsymbol{A}_{\tau_i}\| = \frac{n}{b} = \frac{1}{b}\overline{L}.$$

Thus for orthonormal systems, we gain a factor of b by using mini-batches of size b. However, there is little advantage to weighting in this case as all Lipschitz constants are the same.

Incoherent systems: More generally, the advantage of mini-batching is strong when the rows $\boldsymbol{a}_i$ within any particular batch are *nearly* orthogonal. Suppose that each of the batches is well-conditioned in the sense that

$$\sum_{i=1}^{n} \|\boldsymbol{a}_i\|_2^2 \ge C'n, \qquad \|\boldsymbol{A}_{\tau_i}^* \boldsymbol{A}_{\tau_i}\| = \|\boldsymbol{A}_{\tau_i} \boldsymbol{A}_{\tau_i}^*\| \le C, \qquad i = 1, \ldots, d, \tag{3.10}$$

For example, if $\boldsymbol{A}^*$ has the *restricted isometry property* [CT05] of level δ at sparsity level b, (3.10) holds with $C \le 1+\delta$. Alternatively, if $\boldsymbol{A}$ has unit-norm rows and is incoherent, i.e. $\max_{i \ne j} |\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle| \le \frac{\alpha}{b-1}$, then (3.10) holds with constant $C \le 1 + \alpha$ by Gershgorin circle theorem.

If the incoherence condition (3.10) holds, we gain a factor of b by using weighted mini-batches of size b:

$$\overline{L}_\tau = \sum_{i=1}^{d} \|\boldsymbol{A}_{\tau_i}^* \boldsymbol{A}_{\tau_i}\| \le C\frac{n}{b} \le \frac{C}{C'}\frac{\overline{L}}{b}.$$

Incoherent systems, variable row norms: More generally, consider the case where the rows of $\boldsymbol{A}$ are nearly orthogonal to each other, but not normalized as in (3.10). We can then write $\boldsymbol{A} = \boldsymbol{D}\boldsymbol{\Psi}$, where $\boldsymbol{D}$ is an $n \times n$ diagonal matrix with entry $d_{ii} = \|\boldsymbol{a}_i\|_2$, and $\boldsymbol{\Psi}$ with normalized rows satisfies

$$\|\boldsymbol{\Psi}_{\tau_i}^* \boldsymbol{\Psi}_{\tau_i}\| = \|\boldsymbol{\Psi}_{\tau_i} \boldsymbol{\Psi}_{\tau_i}^*\| \le C, \qquad i = 1, \ldots, d,$$

as is the case if, e.g., $\boldsymbol{\Psi}$ has the restricted isometry property or $\boldsymbol{\Psi}$ is incoherent.

In this case, we have

$$\begin{aligned} \|\boldsymbol{A}^*_{\tau_i}\boldsymbol{A}_{\tau_i}\| &= \|\boldsymbol{A}_{\tau_i}\boldsymbol{A}^*_{\tau_i}\| = \|\boldsymbol{D}_{\tau_i}\boldsymbol{\Psi}_{\tau_i}\boldsymbol{\Psi}^*_{\tau_i}\boldsymbol{D}_{\tau_i}\| \\ &\le \max_{k\in\tau_i}\|\boldsymbol{a}_k\|_2^2\|\boldsymbol{\Psi}_{\tau_i}\boldsymbol{\Psi}^*_{\tau_i}\| \\ &\le C\max_{k\in\tau_i}\|\boldsymbol{a}_k\|_2^2, \qquad i=1,\dots,d. \end{aligned} \tag{3.11}$$

Thus,

$$\overline{L}_\tau = \sum_{i=1}^{d}\|\boldsymbol{A}^*_{\tau_i}\boldsymbol{A}_{\tau_i}\| \le C\sum_{i=1}^{d}\max_{k\in\tau_i}\|\boldsymbol{a}_k\|_2^2. \tag{3.12}$$

In order to minimize the expression on the right hand side over all partitions into blocks of size b, we partition the rows of $\boldsymbol{A}$ according to the order of the decreasing rearrangement of their row norms. This batching strategy results in a factor of b gain in iteration complexity compared to weighting without batching:

$$\begin{aligned} \overline{L}_\tau &\le C\sum_{i=1}^{d}\|\boldsymbol{a}_{((i-1)b+1)}\|_2^2 \\ &\le \frac{C}{b-1}\sum_{i=1}^{n}\|\boldsymbol{a}_i\|_2^2 \\ &\le \frac{C'}{b}\overline{L}. \end{aligned} \tag{3.13}$$

We now turn to the practicality of computing the distribution given by the constants L_{τ_i}. We propose several options to efficiently compute these values given the ability to parallelize over b cores.

Max-norm: The discussion above suggests the use of the maximum row norm of a batch as a proxy for the Lipschitz constant. Indeed, (3.11) shows that the row norms give an upper bound on these constants. Then, (3.13) shows that up to a constant factor, such a proxy still has the potential to lead to an increase in the convergence rate by a factor of b. Of course, computing the maximum row norm of each batch costs on the order of mn flops (the same as the non-batched weighted SGD case).

Power method: In some cases, we may utilize the power method to approximate $\|\boldsymbol{A}^*_{\tau_i}\boldsymbol{A}_{\tau_i}\|$ efficiently. Suppose that for each batch we can approximate this quantity by $\hat{Q}_{\tau_i}$. Classical results on the power method allow one to approximate the norm to within an arbitrary additive error, with a number of iterations that depends on the spectral gap of the matrix. An alternative approach, that we consider here, can be used to obtain approximations leading to a *multiplicative* factor difference in the convergence rate, without dependence on the eigenvalue gaps λ_1/λ_2 within batches. For example, [KL96, Lemma 5] show that with high probability with respect to a randomized initial direction to the power method, after $T \ge \varepsilon^{-1}\log(\varepsilon^{-1}b)$ iterations of the power method, one can guarantee that

$$\|\boldsymbol{A}^*_{\tau_i}\boldsymbol{A}_{\tau_i}\| \ge \hat{Q}_{\tau_i} \ge \frac{\|\boldsymbol{A}^*_{\tau_i}\boldsymbol{A}_{\tau_i}\|}{1+\varepsilon}.$$

At b^2 computations per iteration of the power method, the total computational cost (to compute all quantities in the partition), shared over all b cores, is $b\varepsilon^{-1}\log(\varepsilon^{-1}\log(b))$. This is actually potentially much *lower* than the cost to compute all row norms $L_i = \|\boldsymbol{a}_i\|_2^2$ as in the standard non-batched weighted method. In this case, the power method yields

$$\overline{L}_\tau \ge \frac{b}{n}\sum_{i=1}^{d}\frac{n}{b}\hat{Q}_{\tau_i} \ge \frac{\overline{L}_\tau}{1+\varepsilon},$$

for a constant ε.

4. Mini-batch SGD with weighting: the non-smooth case

We next present analogous results to the previous section for objectives which are strongly convex but lack the smoothness assumption. Like the least squares objective in the previous section, our motivating example here will be the support vector machine (SVM) with hinge loss objective.

A classical result (see e.g. [Nes04, SZ12, RSS12]) for SGD establishes a convergence bound of SGD with non-smooth objectives. In this case, rather than taking a step in the gradient direction of a functional, we move in a direction of a subgradient. Instead of utilizing the Lipschitz constants of the gradient terms, we utilize the Lipschitz constants of the actual functionals themselves. Concretely, a classical bound is of the following form.

Proposition 4.1. *Let the objective $F(\boldsymbol{x}) = \mathbb{E}g_i(\boldsymbol{x})$ with minimizer $\boldsymbol{x}_\star$ be a μ-strongly convex (possibly non-smooth) objective. Run SGD using a subgradient h_i of a randomly selected functional g_i at each iteration. Assume that $\mathbb{E}h_i \in \partial F(\boldsymbol{x}_k)$ and that*

$$\max_{\boldsymbol{x},\boldsymbol{y}} \frac{\|g_i(\boldsymbol{x}) - g_i(\boldsymbol{y})\|}{\|\boldsymbol{x}-\boldsymbol{y}\|} \leq \max_{\boldsymbol{x}} \|h_i(\boldsymbol{x})\| \leq G_i.$$

Set $\overline{G^2} = \mathbb{E}(G_i^2)$. Using step size $\gamma = \gamma_k = 1/(\mu k)$, we have

$$\mathbb{E}\left[F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)\right] \leq \frac{C\overline{G^2}(1+\log k)}{\mu k}, \tag{4.1}$$

where C is an absolute constant.

Such a result can be improved by utilizing averaging of the iterations; for example, if $\boldsymbol{x}_k^\alpha$ denotes the average of the last αk iterates, then the convergence rate bound (4.1) can be improved to:

$$\mathbb{E}\left[F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)\right] \leq \frac{C\overline{G^2}\left(1+\log\frac{1}{\min(\alpha,(1+1/k)-\alpha)}\right)}{\mu k} \leq \frac{C\overline{G^2}\left(1+\log\frac{1}{\min(\alpha,1-\alpha)}\right)}{\mu k}.$$

Setting $m_\alpha = \min(\alpha, 1-\alpha)$, we see that to obtain an accuracy of $\mathbb{E}\left[F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)\right] \leq \varepsilon$, we need

$$k \geq \frac{C\overline{G^2}m_\alpha}{\mu\varepsilon}.$$

In either case, it is important to notice the dependence on $\overline{G^2} = \mathbb{E}(G_i^2)$. By using weighted sampling with weights $p(i) = G_i/\sum_i G_i$, we can improve this dependence to one on $(\overline{G})^2$, where $\overline{G} = \mathbb{E}G_i$ [NSW16, ZZ15]. Since $\overline{G^2} - (\overline{G})^2 = \mathrm{Var}(G_i)$, this improvement reduces the dependence by an amount equal to the variance of the Lipschitz constants G_i. Like in the smooth case, we now consider not only weighting the distribution, but also by batching the functionals g_i. This yields the following result, which we analyze for the specific instance of SVM with hinge loss below.

Theorem 4.2. *Instate the assumptions and notation of Proposition 4.1. Consider the $d = n/b$ batches $g_{\tau_i}(\boldsymbol{x}) = \frac{1}{b}\sum_{j\in\tau_i} g_j(\boldsymbol{x})$, and assume each batch g_{τ_i} has Lipschitz constant G_{τ_i}. Write $\overline{G}_\tau = \mathbb{E}G_{\tau_i}$. Run the weighted batched SGD method with averaging as described above, with step size $\gamma/p(\tau_i)$. For any desired ε, it holds that after*

$$k = \frac{C(\overline{G}_\tau)^2 m_\alpha}{\mu\varepsilon}$$

iterations with weights

$$p(\tau_i) = \frac{G_{\tau_i}}{\sum_j G_{\tau_j}}, \tag{4.2}$$

we have $\mathbb{E}^{(p)}\left[F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)\right] \leq \varepsilon$ where $\mathbb{E}^{(p)}[\cdot]$ means the expectation with respect to the index at each iteration drawn according to the weighted distribution (4.2).

Proof. Applying weighted SGD with weights $p(\tau_i)$, we re-write the objective $P(\boldsymbol{x}) = \mathbb{E}\big(g_i(\boldsymbol{x})\big)$ as $P(\boldsymbol{x}) = \mathbb{E}^{(p)}\left(\hat{g}_{\tau_i}(\boldsymbol{x})\right)$, where

$$\hat{g}_{\tau_i}(x) = \left(\frac{1}{n}\sum_j G_{\tau_j}\right)\left(\frac{1}{G_{\tau_i}}\sum_{j\in\tau_i} g_j(\boldsymbol{x})\right) = \left(\frac{b}{n}\sum_j G_{\tau_j}\right)\left(\frac{g_{\tau_i}(\boldsymbol{x})}{G_{\tau_i}}\right).$$

Then, the Lipschitz constant $\hat{G}_i$ of $\hat{g}_{\tau_i}$ is bounded above by $\hat{G}_i = \frac{b}{n}\sum_j G_{\tau_j}$, and so

$$\mathbb{E}^{(p)}\hat{G}_i^2 = \sum_i \frac{G_{\tau_i}}{\sum_j G_{\tau_j}}\left(\frac{b}{n}\sum_j G_{\tau_j}\right)^2 = \left(\frac{b}{n}\sum_j G_{\tau_j}\right)^2 = (\mathbb{E}G_{\tau_i})^2 = (\overline{G}_\tau)^2.$$

□

We now formalize these bounds and weights for the SVM with hinge loss objective. Other objectives such as L1 regression could also be adapted in a similar fashion, e.g. utilizing an approach as in [YCRM16].

4.1. **SVM with Hinge Loss.**

We now consider the SVM with hinge loss problem as a motivating example for using batched weighted SGD for non-smooth objectives. Recall the SVM with hinge loss objective is

$$P(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^n [y_i\langle \boldsymbol{x}, \boldsymbol{a}_i\rangle]_+ + \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2 = \mathbb{E}g_i(\boldsymbol{x}), \tag{4.3}$$

where $y_i \in \{\pm 1\}$, $[u]_+ = \max(0, u)$, and

$$g_i(\boldsymbol{x}) = [y_i\langle \boldsymbol{x}, \boldsymbol{a}_i\rangle]_+ + \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2.$$

This is a key example where the components are (λ-strongly) convex but no longer smooth. Still, each g_i has a well-defined subgradient:

$$\nabla g_i(\boldsymbol{x}) = \chi_i(\boldsymbol{x}) y_i \boldsymbol{a}_i + \lambda \boldsymbol{x},$$

where $\chi_i(\boldsymbol{x}) = 1$ if $y_i\langle \boldsymbol{x}, \boldsymbol{a}_i\rangle < 1$ and 0 otherwise. It follows that g_i is Lipschitz and its Lipschitz constant is bounded by

$$G_i := \max_{\boldsymbol{x},\boldsymbol{y}} \frac{\|g_i(\boldsymbol{x}) - g_i(\boldsymbol{y})\|}{\|\boldsymbol{x}-\boldsymbol{y}\|} \le \max_{\boldsymbol{x}} \|\nabla g_i(\boldsymbol{x})\| \le \|\boldsymbol{a}_i\|_2 + \lambda.$$

As shown in [ZZ15], [NSW16], in the setting of non-smooth objectives of the form (4.3), where the components are not necessarily smooth, but each g_i is G_i-Lipschitz, the performance of SGD depends on the quantity $\overline{G^2} = \mathbb{E}[G_i^2]$. In particular, the iteration complexity depends linearly on $\overline{G^2}$.

For the hinge loss example, we have calculated that

$$\overline{G^2} = \frac{1}{n}\sum_{i=1}^n (\|\boldsymbol{a}_i\|_2 + \lambda)^2 \le 2\lambda^2 + \frac{2}{n}\sum_{i=1}^n \|\boldsymbol{a}_i\|_2^2.$$

Incorporating (non-batch) weighting to this setting, as discussed in [NSW16], reduces the iteration complexity to depend linearly on $(\overline{G})^2 = (\mathbb{E}[G_i])^2$, which is at most $\overline{G^2}$ and can be as small as $\frac{1}{n}\overline{G^2}$. For the hinge loss example, we have

$$(\overline{G})^2 = \left(\lambda + \frac{1}{n}\sum_{i=1}^n \|\boldsymbol{a}_i\|_2\right)^2.$$

We note here that one can incorporate the dependence on the regularizer term $\frac{\lambda}{2}\|\boldsymbol{x}\|_2^2$ in a more optimal way by bounding the functional norm only over the iterates themselves, as in [TBRS13, RSS12]; however, we choose a crude upper bound on the Lipschitz constant here in order to maintain a dependence on the *average* constant rather than the *maximum*, and only sacrifice a constant factor.

4.1.1. *Batched sampling.* The paper [TBRS13] considered batched SGD for the hinge loss objective. For batches τ_i of size b, let $g_{\tau_i} = \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2 + \frac{1}{b}\sum_{k\in\tau_i}[y_k\langle\boldsymbol{x},\boldsymbol{a}_k\rangle]_+$ and observe

$$P(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n}[y_i\langle\boldsymbol{x},\boldsymbol{a}_i\rangle]_+ + \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2 = \mathbb{E}g_{\tau_i}(\boldsymbol{x}).$$

We now bound the Lipschitz constant G_τ for a batch. Let $\chi = \chi_k(\boldsymbol{x})$ and $\boldsymbol{A}_\tau$ have rows $y_k\boldsymbol{a}_k$ for $k\in\tau$. We have

$$\begin{aligned}\max_{\boldsymbol{x}}\left\|\frac{1}{b}\sum_{k\in\tau_i}\chi_k(\boldsymbol{x})y_k\boldsymbol{a}_k\right\|_2 &= \max_{\boldsymbol{x}}\sqrt{\left\langle\frac{1}{b}\sum_{k\in\tau_i}\chi_k(\boldsymbol{x})y_k\boldsymbol{a}_k, \frac{1}{b}\sum_{k\in\tau_i}\chi_k(\boldsymbol{x})y_k\boldsymbol{a}_k\right\rangle}\\ &= \frac{1}{b}\max_{\boldsymbol{x}}\sqrt{\chi^T\boldsymbol{A}_\tau\boldsymbol{A}_\tau^*\chi}\\ &\le \frac{1}{b}\max_{\boldsymbol{x}}\sqrt{\chi^T\boldsymbol{A}_\tau\boldsymbol{A}_\tau^*\chi}\\ &\le \frac{1}{b}\sqrt{b\|\boldsymbol{A}_\tau\boldsymbol{A}_\tau^*\|}\\ &= \frac{1}{\sqrt{b}}\|\boldsymbol{A}_\tau\|,\end{aligned} \tag{4.4}$$

and therefore $G_\tau \le \frac{1}{\sqrt{b}}\|\boldsymbol{A}_\tau\|+\lambda$. Thus, for batched SGD without weights, the iteration complexity depends linearly on

$$\begin{aligned}\overline{G_\tau^2} &= \frac{b}{n}\sum_{i=1}^{d}G_{\tau_i}^2\\ &\le 2\lambda^2 + \frac{2}{n}\sum_{i=1}^{d}\|\boldsymbol{A}_{\tau_i}\|^2\\ &= 2\lambda^2 + \frac{2}{n}\sum_{i=1}^{d}\|\boldsymbol{A}_{\tau_i}^*\boldsymbol{A}_{\tau_i}\|.\end{aligned}$$

Even without weighting, we already see potential for drastic improvements, as noted in [TBRS13]. For example, in the orthonormal case, where $\|\boldsymbol{A}_{\tau_i}^*\boldsymbol{A}_{\tau_i}\| = 1$ for each τ_i, we see that with appropriately chosen λ, $\overline{G_\tau^2}$ is on the order of $\frac{1}{b}$, which is a factor of b times smaller than $\overline{G^2}\approx 1$. Similar factors are gained for the incoherent case as well, as in the smooth setting discussed above. Of course, we expect even more gains by utilizing both batching and weighting.

4.1.2. *Weighted batched sampling.* Incorporating weighted batched sampling, where we sample batch τ_i with probability proportional to G_τ, the iteration complexity is reduced to a linear dependence on $(\overline{G_\tau})^2$, as in Theorem 4.2. For hinge loss, we calculate

$$(\overline{G_\tau})^2 = \left(\frac{b}{n}\sum_{i=1}^{d}G_{\tau_i}\right)^2 \le \left(\frac{b}{n}\sum_{i=1}^{d}\frac{1}{\sqrt{b}}\|\boldsymbol{A}_{\tau_i}\|+\lambda\right)^2 = \left(\lambda + \frac{\sqrt{b}}{n}\sum_{i=1}^{d}\|\boldsymbol{A}_{\tau_i}\|\right)^2.$$

We thus have the following guarantee for the hinge loss objective.

Corollary 4.3. *Consider $P(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n}[y_i\langle\boldsymbol{x},\boldsymbol{a}_i\rangle]_+ + \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2$. Consider the batched weighted SGD iteration*

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \frac{1}{\mu k p(\tau_i)}\left(\lambda\boldsymbol{x}_k + \frac{1}{b}\sum_{j\in\tau_i}\chi_j(\boldsymbol{x}_k)y_j\boldsymbol{a}_j\right), \tag{4.5}$$

where $\chi_j(\boldsymbol{x}) = 1$ *if* $y_j\langle \boldsymbol{x}, \boldsymbol{a}_j\rangle < 1$ *and* 0 *otherwise. Let* $\boldsymbol{A}_\tau$ *have rows* $y_j\boldsymbol{a}_j$ *for* $j \in \tau$*. For any desired* ε*, we have that after*

$$k = \frac{C\min(\alpha, 1-\alpha)\left(\lambda + \frac{\sqrt{b}}{n}\sum_{i=1}^{d}\|\boldsymbol{A}_{\tau_i}\|\right)^2}{\lambda\varepsilon} \tag{4.6}$$

iterations of (4.5) *with weights*

$$p(\tau_i) = \frac{\|\boldsymbol{A}_{\tau_i}\| + \lambda\sqrt{b}}{\frac{n}{\sqrt{b}}\lambda + \sum_j\|\boldsymbol{A}_{\tau_j}\|}, \tag{4.7}$$

it holds that $\mathbb{E}^{(p)}[P(\mathbf{x}_k) - P(\mathbf{x}_*)] \le \varepsilon$*.*

5. Experiments

In this section we present some simple experimental examples that illustrate the potential of utilizing weighted mini-batching. We consider several test cases as illustration.

Gaussian linear systems: The first case solves a linear system $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A}$ is a matrix with i.i.d. standard normal entries (as is $\boldsymbol{x}$, and $\boldsymbol{b}$ is their product). In this case, we expect the Lipschitz constants of each block to be comparable, so the effect of weighting should be modest. However, the effect of mini-batching in parallel of course still appears. Indeed, Figure 1 (left) displays the convergence rates in terms of iterations for various batch sizes, where each batch is selected with probability as in (3.4). When batch updates can be run in parallel, we expect the convergence behavior to mimic this plot (which displays iterations). We see that in this case, larger batches yield faster convergence. In these simulations, the step size γ was set as in (3.5) (approximations for Lipschitz constants also apply to the step size computation) for the weighted cases and set to the optimal step size as in [NSW16, Corollary 3.2] for the uniform cases. Behavior using uniform selection is very similar (not shown), as expected in this case since the Lipschitz constants are roughly constant. Figure 1 (right) highlights the improvements in our proposed weighted batched SGD method versus the classical, single functional and unweighted, SGD method. The power method refers to the method discussed at the end of Section 3, and max-norm method refers to the approximation using the maximum row norm in a batch, as in (3.11). The notation "(opt)" signifies that the optimal step size was used, rather than the approximation; otherwise in all cases both the sampling probabilities (3.4) and step sizes (3.5) were approximated using the approximation scheme given. Not suprisingly, using large batch sizes yields significant speedup.

Gaussian linear systems with variation: We next test systems that have more variation in the distribution of Lipschitz constants. We construct a matrix $\boldsymbol{A}$ of the same size as above, but whose entries in the kth row are i.i.d. normally distributed with mean zero and variance k^2. We now expect a large effect both from batching and from weighting. In our first experiment, we select the fixed batches randomly at the onset, and compute the probabilities according to the Lipschitz constants of those randomly selected batches, as in (3.4). The results are displayed in the left plot of Figure 2. In the second experiment, we batch sequentially, so that rows with similar Lipschitz constants (row norms) appear in the same batch, and again utilized the weighted sampling. The results are displayed in the center plot of Figure 2. Finally, the right plot of Figure 2 shows convergence when batching sequentially and then employing uniform (unweighted) sampling. As our theoretical results predict, batching sequentially yields better convergence, as does utilizing weighted sampling.

Since this type of system nicely highlights the effects of both weighting and batching, we performed additional experiments using this type of system. Figure 3 highlights the improvements gained by using weighting. In the left plot, we see that for all batch sizes improvements are obtained by using weighting, even more so than in the standard normal case, as expected (note that we cut the curves off when the weighted approach reaches machine precision). In the right plot, we see that the number of iterations to reach a desired threshold is also less using the various weighting schemes; we compare the sampling method using exact computations of the Lipshitz constants (spectral norms), using the maximum row norm as an approximation as in (3.11), and using the power method (using number of iterations equal to $\epsilon^{-1}\log(\epsilon^{-1}b)$ with $\epsilon = 0.01$). Step size γ used on each batch was again set as in (3.5) (approximations for Lipschitz constants also apply to the step size computation) for the weighted cases and as in [NSW16, Corollary 3.2] for the uniform cases. For cases when the exact step size computation was used rather than the corresponding approximation, we write "(opt)". For example, the marker "Max norm (opt)" represents the case when we use the maximum row norm in the batch to approximate the Lipschitz constant, but still use the exact spectral norm when computing the optimal step size. This of course is not practical, but we include these for demonstration. Figure 4 highlights the effect of using batching. The left plot confirms that larger batch sizes yield significant improvement in terms of L2-error and convergence (note that again all curves eventually converge to a straight line due to the error reaching machine precision). The right plot highlights the improvements in our proposed weighted batched SGD methods versus the classical, single functional and unweighted, SGD method.

We next further investigate the effect of using the power method to approximate the Lipschitz constants used for the probability of selecting a given batch. We again create the batches sequentially and fix them throughout the remainder of the method. At the onset of the method, after creating the batches, we run the power method using $\epsilon^{-1}\log(\epsilon^{-1}b)$ iterations (with $\epsilon = 0.01$) per batch, where we assume the work can evenly be divided among the b cores. We then determine the number of computational flops required to reach a specified solution accuracy using various batch sizes b. The results are displayed in Figure 5. The left plot shows the convergence of the method; comparing with the left plot of Figure 2, we see that the convergence is slightly slower than when using the precise Lipschitz constants, as expected. The right plot of Figure 5 shows the number of computational flops required to achieve a specified accuracy, as a function of the batch size. We see that there appears to be an "optimal" batch size, around $b = 40$ for this case, at which the savings in computational time computing the Lipschitz constants and the additional iterations required due to the inaccuracy are balanced.

Correlated linear systems: We next tested the method on systems with correlated rows, using a matrix with i.i.d. entries uniformly distributed on $[0,1]$. When the rows are correlated in this way, the matrix is poorly conditioned and thus convergence speed suffers. Here, we are particularly interested in the behavior when the rows also have high variance; in this case, row k has uniformly distributed entries on $[0,\sqrt{3}k]$ so that each entry has variance k^2 like the Gaussian case above. Figure 6 displays the convergence results when creating the batches randomly and using weighting (left), creating the batches sequentially and using weighting (center), and creating the batches sequentially and using unweighted sampling (right). Like Figure 2, we again see that batching the rows with larger row norms together and then using weighted sampling produces a speedup in convergence.

Orthonormal systems: As mentioned above, we expect the most notable improvement in the case when $\boldsymbol{A}$ is an orthonormal matrix. For this case, we run the method on a 200 × 200 orthonormal discrete Fourier transform (DFT) matrix. As seen in the left plot of Figure 7, we do indeed see significant improvements in convergence with batches in our weighted scheme.

Sparse systems: Lastly, we show convergence for the batched weighted scheme on sparse Gaussian systems. The matrix is generated to have 20% non-zero entries, and each non-zero entry is i.i.d. standard normal. Figure 7 (center) shows the convergence results. The convergence behavior is similar to the non-sparse case, as expected, since our method does not utilize any sparse structure.

Tomography data: The final system we consider is a real system from tomography. The system was generated using the Matlab Regularization Toolbox by P.C. Hansen (`http://www.imm.dtu.dk/~pcha/Regutools/`) [Han07]. This creates a 2D tomography problem $\boldsymbol{Ax} = \boldsymbol{b}$ for an $n \times d$

matrix with $n = fN^2$ and $d = N^2$, where $\boldsymbol{A}$ corresponds to the absorption along a random line through an $N \times N$ grid. We set $N = 20$ and the oversampling factor $f = 3$. Figure 7 (right) shows the convergence results.

Noisy (inconsistent) systems: Lastly, we consider systems that are noisy, i.e. they have no exact solution. We seek convergence to the least squares solution $\boldsymbol{x}_{LS}$. We consider the same Gaussian matrix with variation as desribed above. We first generate a consistent system $\boldsymbol{Ax} = \boldsymbol{b}$ and then add a residual vector $\boldsymbol{e}$ to $\boldsymbol{b}$ that has norm one, $\|\boldsymbol{e}\|_2 = 1$. Since the step size in (3.5) depends on the magnitude of the residual, it will have to be estimated in practice. In our experiments, we estimate this term by an upper bound which is 1.1 times larger in magnitude than the true residual $\|\boldsymbol{Ax}_{LS} - \boldsymbol{b}\|_2$. In addition, we choose an accuracy tolerance of $\varepsilon = 0.1$. Not surprisingly, our experiments in this case show similar behavior to those mentioned above, only the method convergences to a larger error (which can be lowered by adjusting the choice of ε). An example of such results in the correlated Gaussian case are shown in Figure 8.

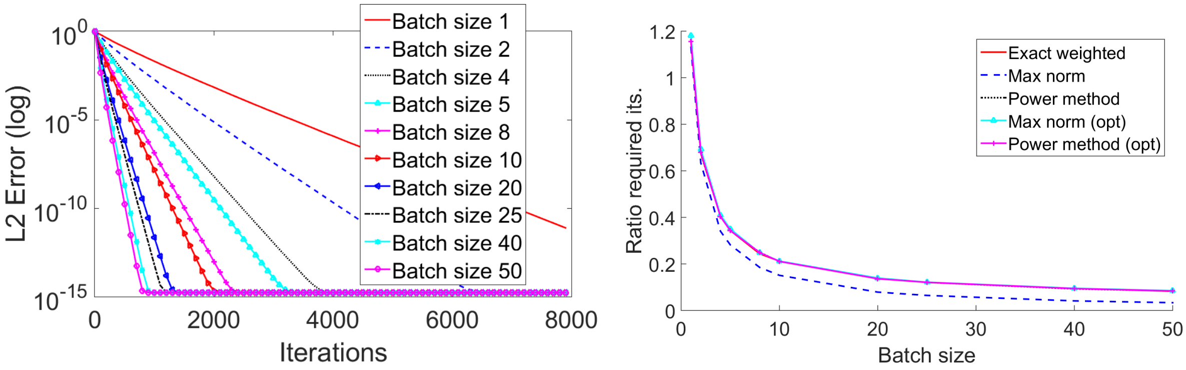

Figure 1 (Gaussian linear systems: convergence) Mini-batch SGD on a Gaussian 1000 × 50 system with various batch sizes; batches created randomly at onset. Graphs show mean L2-error versus iterations (over 40 trials). Step size γ used on each batch was as given in (3.5) for the weighted cases and as in [NSW16, Corollary 3.2] for the uniform comparisons, where in all cases corresponding approximations were used to compute the spectral norms. Left: Batches are selected using proposed weighted selection strategy (3.4). Right: Ratio of the number of iterations required to reach an error of 10^{-5} for weighted batched SGD versus classical (single functional) uniform (unweighted) SGD. The notation "(opt)" signifies that the optimal step size was used, rather than the approximation.

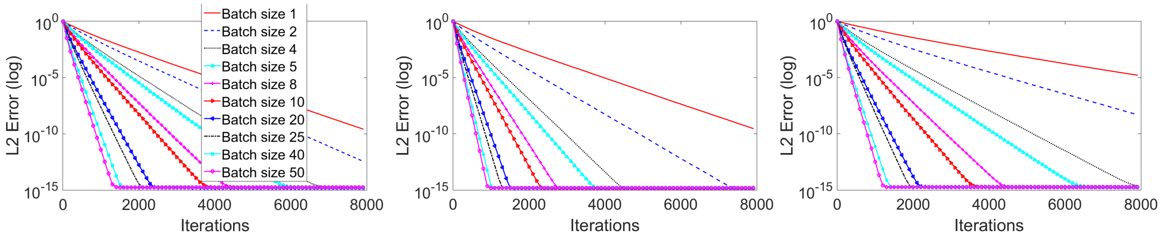

Figure 2 (Gaussian linear systems with variation: convergence) Mini-batch SGD on a Gaussian 1000 × 50 system whose entries in row k have variance k^2, with various batch sizes. Graphs show mean L2-error versus iterations (over 40 trials). Step size γ used on each batch was as given in (3.5) for weighted SGD and the optimal step size as in [NSW16, Corollary 3.2] for uniform sampling SGD. Left: Batches are created randomly at onset, then selected using weighted sampling. Center: Batches are created sequentially at onset, then selected using weighted sampling. Right: Batches are created sequentially at onset, then selected using uniform (unweighted) sampling.

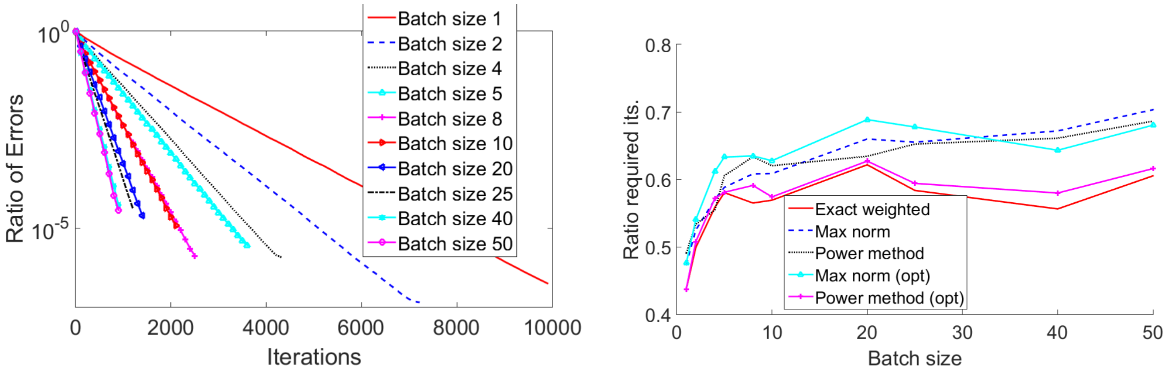

Figure 3 (Gaussian linear systems with variation: effect of weighting) Mini-batch SGD on a Gaussian 1000×50 system whose entries in row k have variance k^2, with various batch sizes; batches created sequentially at onset. Step size γ used on each batch was set as in (3.5) (approximations for Lipschitz constants also apply to the step size computation) for the weighted cases and as in [NSW16, Corollary 3.2] for the uniform cases. Left: Ratio of mean L2-error using weighted versus unweighted random batch selection (improvements appear when plot is less than one). Right: Ratio of the number of iterations required to reach an error of 10^{-5} for various weighted selections versus unweighted random selection. The notation "(opt)" signifies that the optimal step size was used, rather than the approximation.

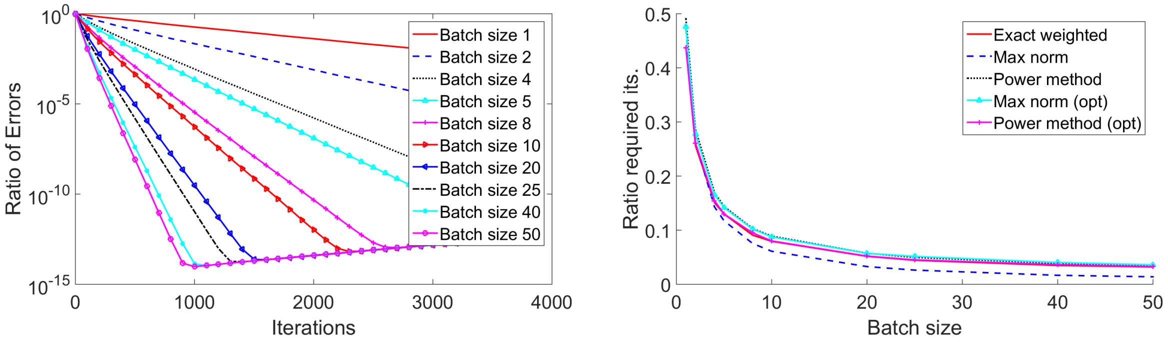

Figure 4 (Gaussian linear systems with variation: effect of batching) Mini-batch SGD on a Gaussian 1000×50 system whose entries in row k have variance k^2, with various batch sizes; batches created sequentially at onset. Step size γ used on each batch was set as in (3.5) (approximations for Lipschitz constants also apply to the step size computation) for the weighted cases and as in [NSW16, Corollary 3.2] for the uniform cases. Left: Ratio of mean L2-error using weighted batched SGD versus classical (single functional) weighted SGD (improvements appear when plot is less than one). Right: Ratio of the number of iterations required to reach an error of 10^{-5} for various weighted selections with batched SGD versus classical (single functional) uniform (unweighted) SGD. The notation "(opt)" signifies that the optimal step size was used, rather than the approximation.

6. Conclusion

We have demonstrated that using a weighted sampling distribution along with batches of functionals in SGD can be viewed as complementary approaches to accelerating convergence. We analyzed the benefits of this combined framework for both smooth and non-smooth functionals, and outlined the specific convergence guarantees for the smooth least squares problem and the non-smooth hinge loss objective. We discussed several computationally efficient approaches to approximating the weights needed in the proposed sampling distributions and showed that one can still obtain approximately the same improved convergence rate. We confirmed our theoretical arguments with experimental evidence that highlight in

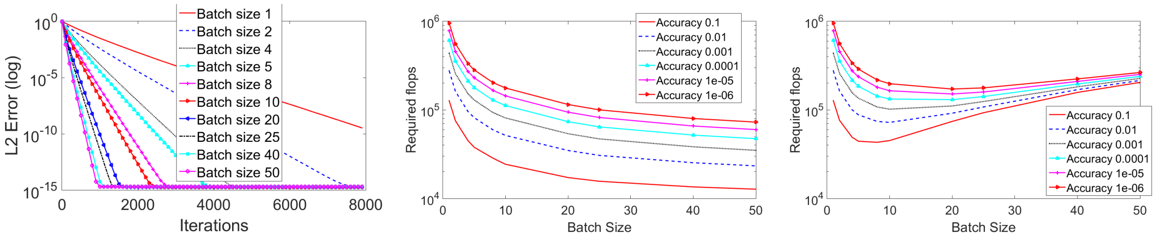

Figure 5 (Gaussian linear systems with variation: using power method) Mini-batch SGD on a Gaussian 1000×50 system whose entries in row k have variance k^2, with various batch sizes; batches created sequentially at onset. Step size γ used on each batch was set as in (3.5) (approximations for Lipschitz constants also apply to the step size computation) for the weighted cases and as in [NSW16, Corollary 3.2] for the uniform cases. Lipschitz constants for batches are approximated by using $\epsilon^{-1}\log(\epsilon^{-1}b)$ (with $\epsilon = 0.01$) iterations of the power method. Left: Convergence of the batched method. Next: Required number of computational flops to achieve a specified accuracy as a function of batch size when computation is shared over b cores (center) or done on a single node (right).

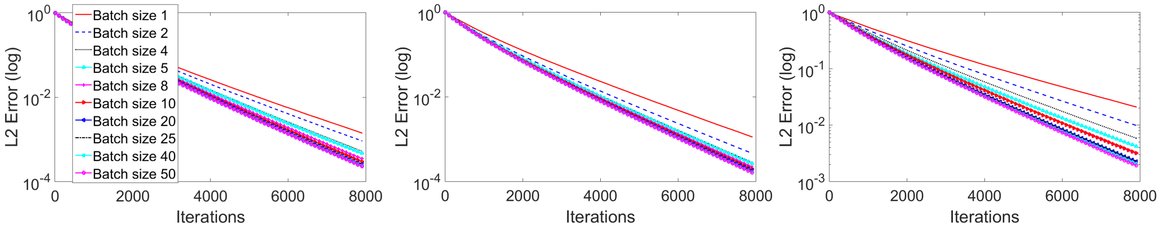

Figure 6 (Correlated systems with variation: convergence) Mini-batch SGD on a uniform 1000×50 system whose entries in row k have variance k^2, with various batch sizes. Graphs show mean L2-error versus iterations (over 40 trials). Step size γ used on each batch was set as in (3.5) (approximations for Lipschitz constants also apply to the step size computation) for the weighted cases and as in [NSW16, Corollary 3.2] for the uniform cases. Left: Batches are created randomly at onset, then selected using weighted sampling. Center: Batches are created sequentially at onset, then selected using weighted sampling. Right: Batches are created sequentially at onset, then selected using uniform (unweighted) sampling.

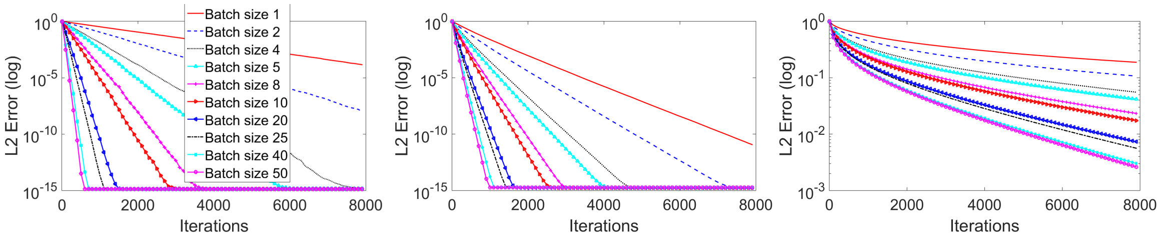

Figure 7 (Orthonormal, sparse, and tomography systems: convergence) Mini-batch SGD on two systems for various batch sizes; batches created randomly at onset. Graphs show mean L2-error versus iterations (over 40 trials). Step size γ used on each batch was set as in (3.5). Left: Matrix is a 200×200 orthonormal discrete Fourier transform (DFT). Center: 1000×50 matrix is a sparse standard normal matrix with density 20%. Right: Tomography data (1200×400 system).

many important settings one can obtain significant acceleration, especially when batches can be computed in parallel. It will be interesting future work to optimize the batch size and other parameters when the parallel computing must be done asynchronously, or in other types of geometric architectures.

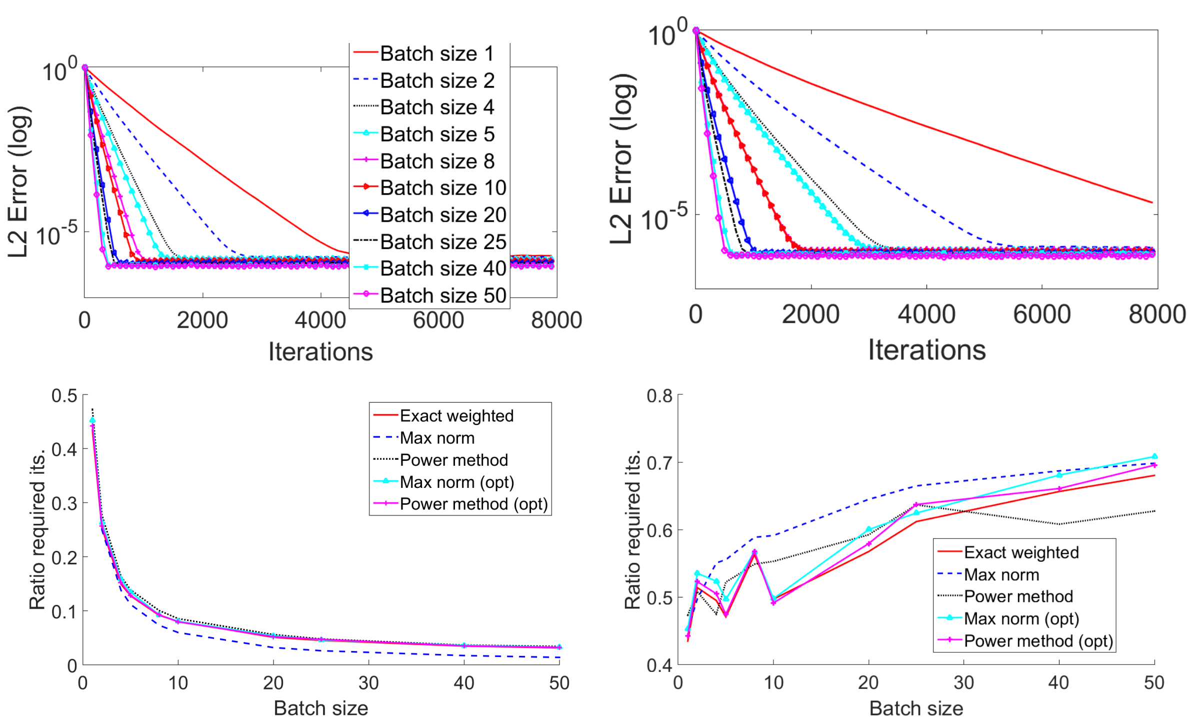

Figure 8 (Noisy systems: convergence) Mini-batch SGD on a Gaussian 1000 × 50 system whose entries in row k have variance k^2, with various batch sizes. Noise of norm 1 is added to system to create an inconsistent system. Graphs show mean L2-error versus iterations (over 40 trials). Step size γ used on each batch was set as in (3.5) for the weighted case and as in [NSW16, Corollary 3.2] for the uniform case; the residual $\boldsymbol{Ax}_{LS} - \boldsymbol{b}$ was upper bounded by a factor of 1.1 in all cases. Upper Left: Batches are created sequentially at onset, then selected using weighted sampling. Upper Right: Batches are created sequentially at onset, then selected using uniform (unweighted) sampling. Lower Left: Ratio of the number of iterations required to reach an error of 10^{-5} for various weighted selections with batched SGD versus classical (single functional) uniform (unweighted) SGD. Lower Right: Ratio of the number of iterations required to reach an error of 10^{-5} for various weighted selections with batched SGD versus classical uniform (unweighted) SGD as a function of batch size.

Acknowledgements

The authors would like to thank Anna Ma for helpful discussions about this paper. Needell was partially supported by NSF CAREER grant #1348721 and the Alfred P. Sloan Foundation. Ward was partially supported by NSF CAREER grant #1255631.

References

[AD11] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

[BB11] L. Bottou and O. Bousquet. The tradeoffs of large-scale learning. *Optimization for Machine Learning*, page 351, 2011.

[BCNW12] R. H. Byrd, G. M. Chin, J. Nocedal, and Y. Wu. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155, 2012.

[BM11] F. Bach and E. Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[Bot10] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[CQR15] D. Csiba, Z. Qu, and P. Richtarik. Stochastic dual coordinate ascent with adaptive probabilities. *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015.

[CR16] D. Csiba and P. Richtarik. Importance sampling for minibatches. *arXiv preprint arXiv:1602.02283*, 2016.

[CSSS11] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *Advances in neural information processing systems*, pages 1647–1655, 2011.

[CT05] E. J. Candès and T. Tao. Decoding by linear programming. *IEEE T. Inform. Theory*, 51:4203–4215, 2005.

[DB15] A. Défossez and F. R. Bach. Averaged least-mean-squares: Bias-variance trade-offs and optimal sampling distributions. In *AISTATS*, 2015.

[DGBSX12] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research*, 13(1):165–202, 2012.

[GR16] R. M. Gower and P. Richtárik. Randomized quasi-newton updates are linearly convergent matrix inversion algorithms. *arXiv preprint arXiv:1602.01768*, 2016.

[Han07] P. C. Hansen. Regularization tools version 4.0 for matlab 7.3. *Numer. Algorithms*, 46(2):189–194, 2007.

[KL96] P. Klein and H.-I. Lu. Efficient approximation algorithms for semidefinite programs arising from max cut and coloring. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 338–347. ACM, 1996.

[KLRT16] J. Konecnỳ, J. Liu, P. Richtarik, and M. Takac. ms2gd: Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255, 2016.

[LS13] Y. T. Lee and A. Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 147–156. IEEE, 2013.

[LZCS14] M. Li, T. Zhang, Y. Chen, and A. J. Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM, 2014.

[Nee10] D. Needell. Randomized Kaczmarz solver for noisy linear systems. *BIT*, 50(2):395–403, 2010.

[Nes04] Y. Nesterov. *Introductory Lectures on Convex Optimization*. Kluwer, 2004.

[Nes12] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optimiz.*, 22(2):341–362, 2012.

[NJLS09] 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.

[NSW16] D. Needell, N. Srebro, and R. Ward. Stochastic gradient descent and the randomized kaczmarz algorithm. *Mathematical Programming Series A*, 155(1):549–573, 2016.

[NW13] D. Needell and R. Ward. Two-subspace projection method for coherent overdetermined linear systems. *Journal of Fourier Analysis and Applications*, 19(2):256–269, 2013.

[QRZ15] Z. Qu, P. Richtarik, and T. Zhang. Quartz: Randomized dual coordinate ascent with arbitrary sampling. In *Advances in neural information processing systems*, volume 28, pages 865–873, 2015.

[RM51] H. Robbins and S. Monroe. A stochastic approximation method. *Ann. Math. Statist.*, 22:400–407, 1951.

[RSS12] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2012.

[RT15] P. Richtárik and M. Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, pages 1–11, 2015.

[SRB13] M. Schmidt, N. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

[SSS08] S. Shalev-Shwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In *Proceedings of the 25th international conference on Machine learning*, pages 928–935, 2008.

[SSSSC11] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[SV09] T. Strohmer and R. Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *J. Fourier Anal. Appl.*, 15(2):262–278, 2009.

[SZ12] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *arXiv preprint arXiv:1212.1824*, 2012.

[TBRS13] M. Takac, A. Bijral, P. Richtarik, and N. Srebro. Mini-batch primal and dual methods for SVMs. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 3, pages 1022–1030, 2013.

[XZ14] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[YCRM16] J. Yang, Y.-L. Chow, C. Ré, and M. W. Mahoney. Weighted sgd for ℓ_p regression with randomized preconditioning. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 558–569. SIAM, 2016.

[ZZ15] P. Zhao and T. Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015.