Stochastic L-BFGS Revisited: Improved Convergence Rates and Practical Acceleration Strategies

Renbo Zhao
Dept. ECE, ISEM & Math
National University of Singapore
elezren@nus.edu.sg

William B. Haskell
Dept. ISEM
National University of Singapore
isehwb@nus.edu.sg

Vincent Y. F. Tan
Dept. ECE & Math
National University of Singapore
vtan@nus.edu.sg

Abstract

We revisit the stochastic limited-memory BFGS (L-BFGS) algorithm. By proposing a new framework for analyzing convergence, we theoretically improve the (linear) convergence rates and computational complexities of the stochastic L-BFGS algorithms in previous works. In addition, we propose several practical acceleration strategies to speed up the empirical performance of such algorithms. We also provide theoretical analyses for most of the strategies. Experiments on large-scale logistic and ridge regression problems demonstrate that our proposed strategies yield significant improvements via-à-vis competing state-of-the-art algorithms.

1 INTRODUCTION

In this work, we are interested in the following (unconstrained) convex finite-sum minimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left[f(\mathbf{x}) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right], \tag{1}$$

where d and n denote the dimension of decision vector and the number of component functions respectively. Problems in the form of (1) play important roles in machine learning and signal processing. One important class of such problems is the *empirical risk minimization* (ERM) problem, where each f_i assumes the form

$$f_i(\mathbf{x}) \triangleq \ell(\mathbf{a}_i^T\mathbf{x}, b_i) + \lambda R(\mathbf{x}). \tag{2}$$

In (2), $\ell : \mathbb{R}\times\mathbb{R} \to \mathbb{R}_+$ denotes a smooth loss function, $R : \mathbb{R}^d \to \mathbb{R}_+$ a smooth convex regularizer (e.g., Tikhonov), $\lambda \geq 0$ the regularization weight and $\{(\mathbf{a}_i, b_i)\}_{i=1}^n \subseteq \mathbb{R}^{d+1}$ the set of feature-response pairs. Depending on the form of ℓ and R, many important machine learning problems are special cases of ERM, such as logistic regression, ridge regression and soft-margin support vector machine.

We focus on the case where both n and d are large, and f is ill-conditioned (i.e., the condition number of f is large).[1] In the context of ERM, this means the dataset $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ that defines (1) has large size and the feature vectors $\mathbf{a}_i$ have high ambient dimension. However, the points typically belong to a low-dimensional manifold. Such a setting is particularly relevant in the big-data era, due to unprecedented data acquisition abilities.

When n is large, the computational cost incurred by the batch optimization methods (both first- and second-order) are prohibitive. Therefore, stochastic (randomized) optimization methods have become very popular in this setting. At each iteration, only a subset of component functions, rather than all of them, are processed. In this way, much more progress can be made towards global optima in the time that only allows batch methods to take a single step. When d is large, Newton- or quasi-Newton-based methods (both batch and stochastic) incur both high computational and storage complexities. Consequently only first-order and limited-memory quasi-newton methods (e.g., L-BFGS (Liu and Nocedal, 1989)) are feasible in this setting.

When both n and d are large, as in our setting, most of research efforts have been devoted to *stochastic first-order methods*, which include stochastic gradient descent (SGD) (Bottou, 1998; Bottou and LeCun, 2004) and its variance-reduced modifications (Defazio et al., 2014; Harikandeh et al., 2015; Johnson and Zhang, 2013; Schmidt et al., 2017). However, these methods do not make use of the curvature information. This limits their abilities to find highly accurate solutions for ill-conditioned problems. In order to incorporate the curvature information in the limited-memory setting, recently much progress have been made toward developing *stochastic L-BFGS* algorithm. A partial list of such works includes Bordes et al. (2009); Byrd et al. (2016); Gower et al. (2016); Mokhtari and Ribeiro (2015); Moritz et al. (2016); Schraudolph et al. (2007); Sohl-Dickstein et al. (2014). Among them, the SQN method (Byrd et al., 2016) innovatively makes use of

[1] In this work, the condition number of a (strongly) convex function refers to that of its Hessian.

the subsampled Hessian-vector products to form the correction pairs (as opposed to using difference of stochastic gradients) and achieves better results than previous methods. However, the convergence rate is sublinear (in the strongly-convex case), similar to that of SGD. Later, Moritz et al. (2016) combines this method with *stochastic variance-reduced gradient* (SVRG) and proves linear convergence of the resulting algorithm. Gower et al. (2016) maintains the structure of this algorithm but incorporates the *block BFGS update* to collect more curvature information in the optimization process. Although the convergence rate of this new method remains similar to that in Moritz et al. (2016), experimental results have demonstrated practical speedup introduced by the block BFGS update.

1.1 MOTIVATIONS AND CONTRIBUTIONS

Our work can be motivated from both theory and practice. In terms of theory, although linear convergence has been shown for both algorithms in Moritz et al. (2016) and Gower et al. (2016), the convergence rates (and hence computational complexities) therein can be potentially further improved.[2] In terms of practice, in addition to block BFGS update, there may exist several other practical strategies that can potentially further accelerate[3] the algorithm in Moritz et al. (2016). Based on these two aspects, our work makes the following contributions.

1) We propose a *coordinate transformation framework* to analyze the stochastic L-BFGS-type algorithms in Moritz et al. (2016) and Gower et al. (2016). Our framework yields a much improved convergence rate and computational complexity. Indeed, our approach can be applied to many other *stochastic second-order algorithms* as well (e.g., SQN algorithm in Byrd et al. (2016)); thus it has great generalizability and wide potential impacts. The essential idea of our method is to unify the analysis of stochastic first-order and second-order method, as a result, it *opens new avenues of designing and analyzing* other variants of stochastic second-order algorithms based on their first-order counterparts. Among these variants, two important examples include proximal (quasi-)Newton methods and momentum-based accelerated (quasi-)Newton methods.

2) We derive *two novel technical lemmas* on variance bound of stochastic gradient with *nonuniform mini-batch sampling* and spectral bound of the sequence of metric matrices $\{\mathbf{H}_r\}_{r\geq 0}$ in stochastic L-BFGS algorithms. These two lemmas are of independent interest. For example, the lemma on the spectral bound is equally applicable to batch L-BFGS algorithms.

[2]The analysis method in Gower et al. (2016) mainly follows that in Moritz et al. (2016), so we treat the analyses in both works in a unified manner.

[3]In this work, we refer "acceleration" to general strategies that speed up the algorithm, not necessarily the ones based on momentum methods.

3) We conduct a *systematic computational complexity analysis* for the stochastic L-BFGS algorithms. To the best of our knowledge, such an analysis is among the first for stochastic L-BFGS methods. Similar to our convergence analysis framework, our complexity analysis method can be easily generalized to other stochastic (quasi-)Newton methods as well.

4) We propose *several practical acceleration strategies* to speed up the convergence of the stochastic L-BFGS algorithm in Moritz et al. (2016). We demonstrate the efficacy of our strategies through numerical experiments on logistic and ridge regression problems. Theoretically, we also prove linear convergence for most strategies.

2 PRELIMINARIES

Notations. We use lowercase, bold lowercase and bold uppercase letters to denote scalars, vectors and matrices respectively. For a matrix $\mathbf{U} \in \mathbb{R}^{m_1\times m_0}$, we denotes its (p,q)-th entry as u_{pq}. For a function $f : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$, define a function $f \circ \mathbf{U}$ such that $(f \circ \mathbf{U})(\mathbf{z}) \triangleq f(\mathbf{U}\mathbf{z})$, for any $\mathbf{z} \in \mathbb{R}^{m_0}$. We use $\mathbb{N}$ to denote the set of natural numbers excluding zero. For any $n \in \mathbb{N}$, we define $[n] \triangleq \{1,\dots,n\}$ and $(n] \triangleq \{0,1,\dots,n\}$. Accordingly, define $\mathcal{A}_{(n]} \triangleq \{\mathcal{A}_0,\dots,\mathcal{A}_n\}$. We use $\|\cdot\|$ to denote both the ℓ_2 norm of a vector and the spectral norm of a matrix. We use $\mathbf{B}$ and $\mathbf{H}$ (with subscripts and superscripts) to denote the approximate Hessian and approximate inverse Hessian in L-BFGS algorithms respectively, following the convention in Nocedal and Wright (2006). $\mathbf{H}$ is also known as the *metric matrix* (Goldfarb, 1970). Sections and lemmas with indices beginning with 'S' will appear in the supplemental material.

Assumptions. We make two blanket assumptions on the component functions $\{f_i\}_{i\in[n]}$.

Assumption 1. *For each $i \in [n]$, f_i is convex and twice differentiable on $\mathbb{R}^d$. For ERM problems* (2)*, we assume these two properties are satisfied by the loss function ℓ in its first argument on $\mathbb{R}$ and by the regularizer R on $\mathbb{R}^d$.*

Assumption 2. *For each $i \in [n]$, f_i is μ_i-strongly convex and L_i-smooth on $\mathbb{R}^d$, where $0 < \mu_i \leq L_i$.*

Remark 1. Assumptions 1 and 2 are standard in the analysis of second-order optimization methods, for both deterministic and stochastic cases. The strong convexity of f_i in Assumption 2 ensures *positive* curvature at any point in $\mathbb{R}^d$, which in turn guarantees the well-definedness of the BFGS update. As a common practice in the literature (Byrd et al., 2016; Moritz et al., 2016), this condition can typically be enforced by adding a strongly convex regularizer (e.g., Tikhonov) to f_i. Due to the strong convexity, (1) has a unique solution, denoted as $\mathbf{x}^*$.

3 ALGORITHM

We present the pseudo-code of the algorithm in Moritz et al. (2016) in Algorithm 1. (Our acceleration strategies for this algorithm will be shown and discussed in Section 6.) In this algorithm, s denotes the outer iteration index, t the inner iteration index and r the index of metric matrices $\{\mathbf{H}_r\}_{r\geq 0}$. We use $\mathbf{x}_{s,t}$ and $\mathbf{x}^s$ to denote an inner iterate and outer iterate respectively. Each outer iteration consists of m inner iterations. The only modification that we make on the original algorithm in Moritz et al. (2016) is sampling the index set $\mathcal{B}_{s,t} \subseteq [n]$ of size b with replacement *nonuniformly*. Specifically, the elements in $\mathcal{B}_{s,t}$ are sampled i.i.d. from a discrete distribution $P \triangleq (p_1, \ldots, p_n)$, such that for any $i \in [n]$, $p_i = L_i / \sum_{i=1}^n L_i$. As will be seen in Lemma 3, compared to uniform sampling, nonuniform sampling leads to a better variance bound on the stochastic gradient $\mathbf{v}_{s,t}$. Using $\mathcal{B}_{s,t}$ and $\nabla f(\mathbf{x}^s)$, we then compute $\mathbf{v}_{s,t}$ according to (5) in Algorithm 1, where for any $\mathbf{x} \in \mathbb{R}^d$,

$$\nabla f_{\mathcal{B}_{s,t}}(\mathbf{x}) \triangleq \frac{1}{b} \sum_{i \in \mathcal{B}_{s,t}} \frac{1}{n p_i} \nabla f_i(\mathbf{x}). \tag{3}$$

Next we compute the search direction formed by the product of the metric matrix $\mathbf{H}_r$ and $\mathbf{v}_{s,t}$. In the limited-memory setting, we do not store $\mathbf{H}_r$, but only a set $\mathcal{H}_r$, which consists of $M' \triangleq \min\{r, M\}$ recent *correction pairs* $\{(\mathbf{s}_j, \mathbf{y}_j)\}_{j=r-M'+1}^r$. The correction pairs are constructed from the averaged past iterates $\{\overline{\mathbf{x}}_r\}_{r\geq 0}$ (defined in line 13), and thus contains the "smoothed" curvature information in recent iterates. Following Byrd et al. (2016), in computing $\mathbf{y}_r$, we first sample an index set $\mathcal{T}_r \subseteq [n]$ of size b_{H} uniformly without replacement, and then let $\mathbf{y}_r = \nabla^2 f_{\mathcal{T}_r}(\overline{\mathbf{x}}_r)\mathbf{s}_r$, where

$$\nabla^2 f_{\mathcal{T}_r}(\overline{\mathbf{x}}_r) \triangleq \frac{1}{b_{\mathrm{H}}} \sum_{i \in \mathcal{T}_r} \nabla^2 f_i(\overline{\mathbf{x}}_r), \tag{4}$$

denotes the sub-sampled Hessian at $\overline{\mathbf{x}}_r$.

In computing $\mathbf{H}_r \mathbf{v}_{s,t}$, a direct approach would be to compute $\mathbf{H}_r$ first and form the product. Computing $\mathbf{H}_r$ involves applying M' BFGS updates to the matrix $\mathbf{H}_r^{(r-M')} \triangleq (\mathbf{s}_r^T \mathbf{y}_r) / \|\mathbf{y}_r\|^2 \mathbf{I}$ using $\{(\mathbf{s}_j, \mathbf{y}_j)\}_{j=r-M'+1}^r$. For each $k \in \{r - M' + 1, \ldots, r\}$, the update is

$$\mathbf{H}_r^{(k)} = \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k{}^T}{\mathbf{y}_k{}^T \mathbf{s}_k}\right) \mathbf{H}_r^{(k-1)} \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k{}^T}{\mathbf{y}_k{}^T \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k{}^T}{\mathbf{y}_k{}^T \mathbf{s}_k}.$$

Finally we set $\mathbf{H}_r = \mathbf{H}_r^{(r)}$. Instead of using this direct approach, we adopt the two-loop recursion algorithm (Nocedal and Wright, 2006, Algorithm 7.4) to compute $\mathbf{H}_r \mathbf{v}_{s,t}$ as a whole. This approach has the same effect as the direct one, but with much reduced computation.

At the end of each outer iteration, the starting point of next outer iteration, $\mathbf{x}^{s+1}$ is either uniformly sampled (option I) or averaged (option II) from all the past inner iterates $\{\mathbf{x}_{s,t}\}_{t\in[m]}$. As shown in Theorem 1, these two options can be analyzed in a unified manner.

Algorithm 1 Stochastic L-BFGS Algorithm with Nonuniform Mini-batch Sampling

1: **Input**: Initial decision vector $\mathbf{x}^0$, mini-batch sizes b and b_{H}, parameters m, M and Υ, step-size η, termination threshold ϵ
2: **Initialize** $s := 0$, $r := 0$, $\overline{\mathbf{x}}_0 = \mathbf{0}$, $\mathbf{H}_0 := \mathbf{I}$, $\mathcal{H}_0 := \emptyset$
3: **Repeat**
4: Compute a full gradient $\mathbf{g}_s \triangleq \nabla f(\mathbf{x}^s)$
5: $\mathbf{x}_{s,0} := \mathbf{x}^s$
6: **for** $t = 0, 1, \ldots, m-1$
7: Sample a set $\mathcal{B}_{s,t}$ with size b
8: Compute a variance-reduced gradient

$$\mathbf{v}_{s,t} := \nabla f_{\mathcal{B}_{s,t}}(\mathbf{x}_{s,t}) - \nabla f_{\mathcal{B}_{s,t}}(\mathbf{x}^s) + \mathbf{g}_s \tag{5}$$

9: Compute $\mathbf{H}_r \mathbf{v}_{s,t}$ from $\mathcal{H}_r$ and $\mathbf{v}_{s,t}$
10: $\mathbf{x}_{s,t+1} := \mathbf{x}_{s,t} - \eta \mathbf{H}_r \mathbf{v}_{s,t}$
11: **if** $sm + t > 0$ **and** $(sm + t) \equiv 0 \mod \Upsilon$
12: $r := r + 1$
13: $\overline{\mathbf{x}}_r := \frac{1}{\Upsilon} \sum_{l=t-\Upsilon+1}^{t} \mathbf{x}_{s,l}$
14: Sample a set $\mathcal{T}_r$ with size b_{H}
15: $\mathbf{s}_r := \overline{\mathbf{x}}_r - \overline{\mathbf{x}}_{r-1}$, $\mathbf{y}_r := \nabla^2 f_{\mathcal{T}_r}(\overline{\mathbf{x}}_r)\mathbf{s}_r$
16: Update $\mathcal{H}_r := \{(\mathbf{s}_j, \mathbf{y}_j)\}_{j=r-M'+1}$
17: **end if**
18: **end for**
19: **Option I**: Sample τ_s uniformly randomly from $[m]$ and set $\mathbf{x}^{s+1} := \mathbf{x}_{s,\tau_s}$
20: **Option II**: $\mathbf{x}^{s+1} := \frac{1}{m} \sum_{t=1}^{m} \mathbf{x}_{s,t}$
21: $s := s + 1$
22: **Until** $\left|f(\mathbf{x}^s) - f(\mathbf{x}^{s-1})\right| < \epsilon$
23: **Output**: $\mathbf{x}^s$

4 CONVERGENCE ANALYSIS

4.1 DEFINITIONS

Let $\{\underline{i}_j\}_{j\in[n]}$ and $\{\bar{i}_j\}_{j\in[n]}$ be permutations of $[n]$ such that $\mu_{\min} \triangleq \mu_{\underline{i}_1} \leq \cdots \leq \mu_{\underline{i}_n}$ and $L_{\bar{i}_1} \leq \cdots \leq L_{\bar{i}_n} \triangleq L_{\max}$. Given any $\widetilde{n} \in [n]$, define

$$\overline{\mu}_{\widetilde{n}} \triangleq \frac{1}{\widetilde{n}} \sum_{j=1}^{\widetilde{n}} \mu_{\underline{i}_j} \quad \text{and} \quad \overline{L}_{\widetilde{n}} \triangleq \frac{1}{\widetilde{n}} \sum_{j=n-\widetilde{n}+1}^{n} L_{\bar{i}_j}. \tag{6}$$

Accordingly, define $\kappa_{\max} \triangleq L_{\max}/\mu_{\min}$ and $\kappa_{\widetilde{n}} \triangleq \overline{L}_{\widetilde{n}}/\overline{\mu}_{\widetilde{n}}$. In particular, define $\overline{\mu} \triangleq \overline{\mu}_n$, $\overline{L} \triangleq \overline{L}_n$ and $\kappa \triangleq \overline{L}/\overline{\mu}$. We also define a filtration $\{\mathcal{F}_{s,t}\}_{s\geq 0, t\in(m-1]}$ such that $\mathcal{F}_{s,t}$ contains all the information up to the time (s,t). Formally, $\mathcal{F}_{s,t} \triangleq \sigma(\{\tau_j\}_{j=0}^{s-1} \cup \{\mathcal{B}_{i,j}\}_{i\in(s-1], j\in(m-1]} \cup \{\mathcal{B}_{s,j}\}_{j=0}^{t-1} \cup \{\mathcal{T}_j\}_{j=0}^{\lfloor (sm+t)/L \rfloor})$, where $\sigma\left(\{x_j\}_{j=1}^n\right)$ denotes

the σ-algebra generated by random variables $\{x_j\}_{j=1}^n$. We also define $\mathcal{F}_s \triangleq \mathcal{F}_{s,0}$.

To introduce our coordinate transformation framework, we define some transforms of variables appearing in Algorithm 1. Specifically, for any $s,t,r \geq 0$, define $\widetilde{\mathbf{x}}_{s,t,r} \triangleq \mathbf{H}_r^{-1/2}\mathbf{x}_{s,t}$, $\widetilde{\mathbf{x}}^{s,r} \triangleq \mathbf{H}_r^{-1/2}\mathbf{x}^s$, $\widetilde{\mathbf{x}}_r^* \triangleq \mathbf{H}_r^{-1/2}\mathbf{x}^*$ and $\widetilde{\mathbf{v}}_{s,t,r} \triangleq \mathbf{H}_r^{1/2}\mathbf{v}_{s,t}$.[4] We also define transformed functions $\widetilde{f}_{i,r} \triangleq f_i \circ \mathbf{H}_r^{1/2}$ and $\widetilde{f}_r \triangleq \frac{1}{n}\sum_{i=1}^n \widetilde{f}_{i,r}$, for any $i \in [n]$ and $r \geq 0$.

4.2 PRELIMINARY LEMMAS

From the definitions of transformed variables and functions in Section 4.1, we immediately have the following lemmas.

Lemma 1. *For any $s,t,r \geq 0$ and $i \in [n]$, we have*

$$\widetilde{f}_{i,r}(\widetilde{\mathbf{x}}_{s,t,r}) = f_i(\mathbf{x}_{s,t}), \tag{7}$$

$$\nabla \widetilde{f}_{i,r}(\widetilde{\mathbf{x}}_{s,t,r}) = \mathbf{H}_r^{1/2}\nabla f_i(\mathbf{x}_{s,t}), \tag{8}$$

$$\nabla^2 \widetilde{f}_{i,r}(\widetilde{\mathbf{x}}_{s,t,r}) = \mathbf{H}_r^{1/2}\nabla f_i(\mathbf{x}_{s,t})\mathbf{H}_r^{1/2}. \tag{9}$$

Lemma 2. *If there exist $0 < \gamma' \leq \Gamma'$ such that $\gamma'\mathbf{I} \preceq \mathbf{H}_r \preceq \Gamma'\mathbf{I}$ for all $r \geq 0$, then for any $i \in [n]$ and $r \geq 0$, $\widetilde{f}_{i,r}$ is twice differentiable, $(\mu_i\gamma')$-strongly convex and $(L_i\Gamma')$-smooth on $\mathbb{R}^d$. Consequently, $\widetilde{f}_r$ is twice differentiable, $(\gamma'\overline{\mu})$-strongly convex and $(\Gamma'\overline{L})$-smooth on $\mathbb{R}^d$.*

Next we derive two other lemmas that will not only be used in the analysis later, but have potential to be applied to more general problem settings. Specifically, Lemma 3 can be applied to any stochastic optimization algorithms based on SVRG and Lemma 4 can be applied to any finite-sum minimization algorithms based on L-BFGS methods (not necessarily stochastic in nature). The proofs for Lemmas 3 and 4 are deferred to Sections S-3 and S-4 respectively.

Lemma 3 (Variance bound of $\mathbf{v}_{s,t}$)**.** *In Algorithm 1, we have $\mathbb{E}_{\mathcal{B}_{s,t}}[\mathbf{v}_{s,t}|\mathcal{F}_{s,t}] = \nabla f(\mathbf{x}_{s,t})$ and*

$$\mathbb{E}_{\mathcal{B}_{s,t}}\left[\|\mathbf{v}_{s,t} - \nabla f(\mathbf{x}_{s,t})\|^2 |\mathcal{F}_{s,t}\right] \leq \frac{4\overline{L}}{b}\left(f(\mathbf{x}_{s,t}) - f(\mathbf{x}^*) + f(\mathbf{x}^s) - f(\mathbf{x}^*)\right). \tag{10}$$

Remark 2. In previous works (Gower et al., 2016; Moritz et al., 2016), a uniform mini-batch sampling of $\mathcal{B}_{s,t}$ was employed, and different variance bounds of $\mathbf{v}_{s,t}$ were derived. In Moritz et al. (2016, Lemma 6), the bound was

$$4L_{\max}\left(f(\mathbf{x}_{s,t}) - f(\mathbf{x}^*) + f(\mathbf{x}^s) - f(\mathbf{x}^*)\right). \tag{11}$$

In Gower et al. (2016, Lemma 2), this bound was slightly improved to be

$$4L_{\max}((f(\mathbf{x}_{s,t}) - f(\mathbf{x}^*)) + (1 - 1/\kappa_{\max})\left(f(\mathbf{x}^s) - f(\mathbf{x}^*)\right)). \tag{12}$$

However, both of these bounds fail to capture the dependence on the mini-batch size b. In contrast, in this work we consider a nonuniform mini-batch sampling (with replacement). Due to division by b and $\overline{L} \leq L_{\max}$ (indeed in many cases, $\overline{L} \ll L_{\max}$), our bound in (10) is much superior to (11) and (12). As will be seen in Theorem 1, our better bound (10) leads to a faster (linear) convergence rate of Algorithm 1.

Lemma 4 (Uniform Spectral Bound of $\{\mathbf{H}_r\}_{r\geq 0}$)**.** *The spectra of $\{\mathbf{H}_r\}_{r\geq 0}$ are uniformly bounded, i.e., for each $r \geq 0$, $\gamma\mathbf{I} \preceq \mathbf{H}_r \preceq \Gamma\mathbf{I}$, where*[5]

$$\gamma \triangleq \frac{1}{(M+1)\overline{L}_{b_{\mathrm{H}}}} \quad \text{and} \quad \Gamma \triangleq \frac{\kappa_{b_{\mathrm{H}}}^{M+1}}{\overline{\mu}_{b_{\mathrm{H}}}(\kappa_{b_{\mathrm{H}}} - 1)}. \tag{13}$$

Remark 3. In Byrd et al. (2016) and Moritz et al. (2016), the authors make use of a classical technique in Liu and Nocedal (1989) to derive a different uniform spectral bound of $\{\mathbf{H}_r\}_{r\geq 0}$. Their technique involves applying $\mathrm{tr}(\cdot)$ and $\det(\cdot)$ recursively to the BFGS update rule

$$\mathbf{B}_r^{(k)} = \mathbf{B}_r^{(k-1)} - \frac{\mathbf{B}_r^{(k-1)}\mathbf{s}_k\mathbf{s}_k{}^T\mathbf{B}_r^{(k-1)}}{\mathbf{s}_k{}^T\mathbf{B}_r^{(k-1)}\mathbf{s}_k} + \frac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{s}_k^T\mathbf{y}_k}, \tag{14}$$

where $\mathbf{B}_r^{(k)} \triangleq (\mathbf{H}_r^{(k)})^{-1}$ denotes the approximate Hessian matrix at step k in the reconstruction of $\mathbf{B}_r \triangleq (\mathbf{H}_r)^{-1}$. The lower and upper bounds derived by this technique are

$$\widetilde{\gamma} = \frac{1}{(d+M)L_{\max}} \text{ and } \widetilde{\Gamma} = (d+M)^{d+M-1}\frac{\kappa_{\max}^{d+M-1}}{\mu_{\min}}$$

respectively. As will be seen in Proposition 1, the overall computational complexity of Algorithm 1 heavily depends on the estimated (uniform) condition number of $\{\mathbf{H}_r\}_{r\geq 0}$. Therefore, it is instructive to compute this quantity for both (γ,Γ) and $(\widetilde{\gamma},\widetilde{\Gamma})$ as

$$\kappa_{\mathrm{H}} \triangleq \frac{\Gamma}{\gamma} = (M+1)\frac{\kappa_{b_{\mathrm{H}}}^{M+2}}{\kappa_{b_{\mathrm{H}}} - 1} \approx (M+1)\kappa_{b_{\mathrm{H}}}^{M+1}, \tag{15}$$

$$\widetilde{\kappa}_{\mathrm{H}} \triangleq \frac{\widetilde{\Gamma}}{\widetilde{\gamma}} = (M+d)^{M+d}\kappa_{\max}^{M+d}, \tag{16}$$

where the approximation in (15) follows from $\kappa_{b_{\mathrm{H}}} \gg 1$ (see Footnote 5). By comparing (15) and (16), we notice our estimate for the condition number of $\{\mathbf{H}_r\}_{r\geq 0}$, namely κ_{H}, is smaller than those in Byrd et al. (2016) and Moritz et al. (2016), namely $\widetilde{\kappa}_{\mathrm{H}}$, in several aspects. First, κ_{H} does not grow (exponentially) with the data dimension d. Second, κ_{H} depends on $\kappa_{b_{\mathrm{H}}}$, which is usually much smaller than $\kappa_{\max}$. Third, even if we set $d = 1$ in (16), the factor $M+1$ in (15) is much smaller than the factor $(M+1)^{M+1}$ in (16). As a result, our improved estimate of the condition number of $\{\mathbf{H}_r\}_{r\geq 0}$ will lead to a much better computational complexity estimate (see Proposition 1).

[4] As will be shown in Lemma 4, for any $r \geq 0$, $\mathbf{H}_r \succeq \gamma\mathbf{I}$ for some $\gamma > 0$. Therefore, $\mathbf{H}_r^{1/2}$ (and $\mathbf{H}_r^{-1/2}$) are well-defined.

[5] We assume $\kappa_{b_{\mathrm{H}}} > 1$ for any $b_{\mathrm{H}} \in [n]$ since we focus on the setting where f is ill-conditioned, i.e., $\kappa_{b_{\mathrm{H}}} \geq \kappa \gg 1$. If $\kappa_{b_{\mathrm{H}}} = 1$ for some $b_{\mathrm{H}} \in [n]$, then $\Gamma = (\kappa_{b_{\mathrm{H}}}^M + M)/\overline{\mu}_{b_{\mathrm{H}}}$ and γ remains the same. The proof for this case can be straightforwardly adapted from that in Section S-4.

4.3 MAIN RESULT

Theorem 1 presents our main convergence result. For the complete proof, we refer readers to Section S-2.

Theorem 1. *In Algorithm 1, if we choose* $\eta < \min\{b/12, 1\}/(\Gamma\overline{L})$ *and* m *sufficiently large, then with either option I or II, we have*

$$\mathbb{E}\left[f(\mathbf{x}^s) - f(\mathbf{x}^*)\right] \leq \rho^s \left(f(\mathbf{x}^0) - f(\mathbf{x}^*)\right), \text{ where} \quad (17)$$

$$\rho = \frac{b}{\gamma\overline{\mu}m\eta(b - 4\eta\Gamma\overline{L})} + \frac{4\eta\Gamma\overline{L}}{b - 4\eta\Gamma\overline{L}}\left(1 + \frac{1}{m}\right) < 1. \quad (18)$$

Remark 4. We compare our linear convergence rate ρ in (18) with those in Moritz et al. (2016) and Gower et al. (2016). Since the convergence rates in these two works are almost the same, we use the rate in Moritz et al. (2016) for comparison. The linear rate $\widetilde{\rho}$ in Moritz et al. (2016) equals

$$\frac{1}{2\widetilde{\gamma}\mu_{\min}m\eta(1 - \eta\widetilde{\Gamma}L_{\max}\kappa_{\max}\widetilde{\kappa}_{\mathrm{H}})} + \frac{\eta\widetilde{\Gamma}L_{\max}\kappa_{\max}\widetilde{\kappa}_{\mathrm{H}}}{1 - \eta\widetilde{\Gamma}L_{\max}\kappa_{\max}\widetilde{\kappa}_{\mathrm{H}}}.$$

For simplicity, if we let $b = 1$, $\mu_{\min} = \overline{\mu}$, $L_{\max} = \overline{L}$, $\widetilde{\gamma} = \gamma$, $\widetilde{\Gamma} = \Gamma$ and ignore other constant factors,[6] we notice that there is an additional multiplicative factor $\kappa\kappa_{\mathrm{H}}$ associated with $\eta\Gamma\overline{L}$ in $\widetilde{\rho}$. As a result $\widetilde{\rho} > \rho$. A more direct way to observe the detrimental effects of this additional $\kappa\kappa_{\mathrm{H}}$ is to compare the computational complexities resulting from ρ and $\widetilde{\rho}$. See Remark 6 for details. The reason that we manage to avoid this factor in our rate ρ is precisely because we adopt the *coordinate transformation framework* in our analysis (see proof sketch below). Specifically, by absorbing the sequence of metric matrices $\{\mathbf{H}_r\}_{r\geq 0}$ into decision vectors and functions, we are able to proceed through bounding the (expected squared Euclidean) distance between $\widetilde{\mathbf{x}}_{s,t+1,r}$ and $\widetilde{\mathbf{x}}_r^*$, instead of directly bounding $f(\mathbf{x}_{s,t+1})$ via the smoothness property of f (cf. proof of Theorem 7 in Moritz et al. (2016)). Thus in our analysis, we avoid the additional appearance of $\overline{L}$ and Γ (which leads to the additional factor $\kappa\kappa_{\mathrm{H}}$).

Proof Sketch. Fix an outer iteration s and consider an inner iteration t. Define $r \triangleq \lfloor (sm + t)/L \rfloor$,[7] then the iteration in (10) becomes

$$\widetilde{\mathbf{x}}_{s,t+1,r} = \widetilde{\mathbf{x}}_{s,t,r} - \eta\widetilde{\mathbf{v}}_{s,t,r}. \quad (19)$$

From Lemmas 1, 3 and 4, we have $\mathbb{E}_{\mathcal{B}_{s,t}}\left[\widetilde{\mathbf{v}}_{s,t,r} \middle| \mathcal{F}_{s,t}\right] = \nabla\widetilde{f}_r(\widetilde{\mathbf{x}}_{s,t,r})$ and

$$\mathbb{E}_{\mathcal{B}_{s,t}}\left[\|\widetilde{\mathbf{v}}_{s,t,r} - \nabla\widetilde{f}_r(\widetilde{\mathbf{x}}_{s,t,r})\|^2 \middle| \mathcal{F}_{s,t}\right] \leq \frac{4\Gamma\overline{L}}{b}(\widetilde{f}_r(\widetilde{\mathbf{x}}_{s,t,r}) - \widetilde{f}_r(\widetilde{\mathbf{x}}_r^*) + \widetilde{f}_{r'}(\widetilde{\mathbf{x}}^{s,r'}) - \widetilde{f}_{r'}(\widetilde{\mathbf{x}}_{r'}^*)), \quad (20)$$

where $r' \triangleq \lfloor sm/L \rfloor$. Define $\widetilde{\boldsymbol{\delta}}_{s,t,r} \triangleq \widetilde{\mathbf{v}}_{s,t,r} - \nabla\widetilde{f}_r(\widetilde{\mathbf{x}}_{s,t,r})$. Using $\eta \leq 1/(\Gamma\overline{L})$ and Lemma 2, we are able to show that

$$\|\widetilde{\mathbf{x}}_{s,t+1,r} - \widetilde{\mathbf{x}}_r^*\|^2 \leq \|\widetilde{\mathbf{x}}_{s,t,r} - \widetilde{\mathbf{x}}_r^*\|^2 - 2\eta(\widetilde{f}_r(\widetilde{\mathbf{x}}_{s,t+1,r}) - \widetilde{f}_r(\widetilde{\mathbf{x}}_r^*)) + 2\eta^2\|\widetilde{\boldsymbol{\delta}}_{s,t,r}\|^2 - 2\eta\left\langle\widetilde{\boldsymbol{\delta}}_{s,t,r}, \widetilde{\mathbf{x}}_{s,t,r} - \widetilde{\mathbf{x}}_r^*\right\rangle.$$

Taking expectation w.r.t. $\mathcal{B}_{s,t}$ and using (S-8), we have

$$\mathbb{E}_{\mathcal{B}_{s,t}}\left[\|\widetilde{\mathbf{x}}_{s,t+1,r} - \widetilde{\mathbf{x}}_r^*\|^2 + 2\eta(\widetilde{f}_r(\widetilde{\mathbf{x}}_{s,t+1,r}) - \widetilde{f}_r(\widetilde{\mathbf{x}}_r^*)) \middle| \mathcal{F}_{s,t}\right] \leq \|\widetilde{\mathbf{x}}_{s,t,r} - \widetilde{\mathbf{x}}_r^*\|^2 + \frac{8}{b}\Gamma\overline{L}\eta^2(\widetilde{f}_r(\widetilde{\mathbf{x}}_{s,t,r}) - \widetilde{f}_r(\widetilde{\mathbf{x}}_r^*) + \widetilde{f}_{r'}(\widetilde{\mathbf{x}}^{s,r'}) - \widetilde{f}_{r'}(\widetilde{\mathbf{x}}_{r'}^*)). \quad (21)$$

Telescoping (S-20) over $t \in (0, m-1]$ and noting that both options I and II in Algorithm 1 yield

$$\frac{1}{m}\sum_{t=1}^{m}\mathbb{E}_{\mathcal{B}_{s,(t-1]}}\left[\widetilde{f}_r(\widetilde{\mathbf{x}}_{s,t,r}) - \widetilde{f}_r(\widetilde{\mathbf{x}}_r^*) \middle| \mathcal{F}_{s,t-1}\right] \geq \mathbb{E}_{\mathcal{B}_{s,(m-1]}}\left[\widetilde{f}_{r''}(\widetilde{\mathbf{x}}^{s+1,r''}) - \widetilde{f}_{r''}(\widetilde{\mathbf{x}}_{r''}^*) \middle| \mathcal{F}_s\right], \quad (22)$$

where $r'' \triangleq \lfloor (s+1)m/L \rfloor$, we have

$$2m\eta(1 - \frac{4}{b}\Gamma\overline{L}\eta)\mathbb{E}_{\mathcal{B}_{s,(m-1]}}\left[\widetilde{f}_{r''}(\widetilde{\mathbf{x}}^{s+1,r''}) - \widetilde{f}_{r''}(\widetilde{\mathbf{x}}_{r''}^*) \middle| \mathcal{F}_s\right] \leq \|\widetilde{\mathbf{x}}^{s,r'} - \widetilde{\mathbf{x}}_{r'}^*\|^2 + \frac{8}{b}\Gamma\overline{L}\eta^2(1+m)(\widetilde{f}_{r'}(\widetilde{\mathbf{x}}^{s,r'}) - \widetilde{f}_{r'}(\widetilde{\mathbf{x}}_{r'}^*)).$$

Using $(\gamma\overline{\mu})$-strong convexity of $\widetilde{f}_{r'}$, we can derive the bound $\|\widetilde{\mathbf{x}}^{s,r'} - \widetilde{\mathbf{x}}_{r'}^*\|^2 \leq \frac{2}{\gamma\overline{\mu}}(\widetilde{f}_{r'}(\widetilde{\mathbf{x}}^{s,r'}) - \widetilde{f}_{r'}(\widetilde{\mathbf{x}}_{r'}^*))$. Finally, using Lemma 1 and rearranging, we have

$$\mathbb{E}\left[f(\mathbf{x}^{s+1}) - f(\mathbf{x}^*) \middle| \mathcal{F}_s\right] \leq \rho(f(\mathbf{x}^s) - f(\mathbf{x}^*)). \quad (23)$$

Taking expectation on both sides, we reach (17). □

5 COMPLEXITY ANALYSIS

In this section we provide a systematic framework for analyzing the computational complexity of the stochastic L-BFGS algorithm in Algorithm 1. Note that our framework can be easily generalized to other stochastic second-order algorithms, e.g., SQN algorithm in Byrd et al. (2016). To begin with, we make two additional assumptions.

Assumption 3. *For any* $\mathbf{x} \in \mathbb{R}^d$ *and* $i \in [n]$*, the gradient* $\nabla f_i(\mathbf{x})$ *can be computed in* $O(d)$ *operations.*[8]

Assumption 4. *For any* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ *and* $i \in [n]$*, the Hessian-vector product* $\nabla^2 f_i(\mathbf{x})\mathbf{y}$ *can be computed in* $O(d)$ *operations.*

Remark 5. These two assumptions are naturally satisfied for ERM problems (2) with Tikhonov regularization. For these problems, $R(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|^2$ and

$$\nabla\ell(\mathbf{a}_i^T\mathbf{x}, b_i) = \ell'(\mathbf{a}_i^T\mathbf{x}, b_i)\mathbf{a}_i + \lambda\mathbf{x}, \quad (24)$$

$$\nabla^2\ell(\mathbf{a}_i^T\mathbf{x}, b_i)\mathbf{y} = \ell''(\mathbf{a}_i^T\mathbf{x}, b_i)(\mathbf{a}_i^T\mathbf{y})\mathbf{a}_i + \lambda\mathbf{y}, \quad (25)$$

[6] However, bear in mind that by doing all these substitutions, $\widetilde{\rho}$ has already been much improved.

[7] To avoid cluttered notations, we omit showing the dependence of r on s and t.

[8] An operation refers to evaluation of an elementary function, such as addition, multiplication and logarithm.

where $\ell'(\cdot,\cdot)$ and $\ell''(\cdot,\cdot)$ are first and second derivatives of $\ell(\cdot,\cdot)$ w.r.t. the first argument. We easily see that the right-hand sides of both (24) and (25) can be computed in $O(d)$ operations.

From Algorithm 1, we observe that its total computational cost C can be split into three parts. The first part C_1 involves computing the variance-reduced gradient $\mathbf{v}_{s,t}$ in (5), the second part C_2 involves computing $\mathbf{H}_r\mathbf{v}_{s,t}$ (via two-loop recursion) in line 9, and the third part C_3 involves computing the correction pair $(\mathbf{s}_r, \mathbf{y}_r)$ in line 15.

Proposition 1. *Assume Assumptions 1 to 4. In Algorithm 1,*

$$C_1 = O\left((n+\kappa\kappa_{\mathrm{H}})d\log(1/\epsilon)\right), \tag{26}$$

$$C_2 = O\left(\kappa\kappa_{\mathrm{H}} d\log(1/\epsilon)\right), \tag{27}$$

$$C_3 = O\left(d\log(1/\epsilon)\right). \tag{28}$$

Thus the total computational cost $C \triangleq \sum_{i=1}^{3} C_i$ equals

$$O\left((n+\kappa\kappa_{\mathrm{H}})\, d\log(1/\epsilon)\right). \tag{29}$$

Proof. We leverage techniques that have become standard in the SVRG literature (e.g., Xiao and Zhang (2014)). In (18), if we choose $\eta = \theta b/(\Gamma\overline{L})$ for some $0 < \theta < 1/12$, $m = \theta'\kappa\kappa_{\mathrm{H}}/b$ for some sufficiently large positive constant θ', and use $1 + 1/m \leq 2$, then $\rho = \frac{1}{\theta'\theta(1-4\theta)} + \frac{8\theta}{1-4\theta} < 1$. As a result, the required number of outer iterations to achieve ϵ-suboptimality is $O(\log(1/\epsilon))$. Thus (26) follows from Assumption 3 and that $2mb$ gradients (of component functions) are computed in each inner iteration. If we further choose $M = \Theta(b)$, then (27) follows from the fact that two-loop recursion can be done in $O(Md)$ time (Nocedal and Wright, 2006, Chapter 7). Lastly, if we choose $b_{\mathrm{H}} = \Theta(\Upsilon)$, then we obtain (28) using Assumption 4. □

Remark 6. Following a similar argument, we can deduce the total complexity estimate $\widetilde{C}$ based on the linear rate $\widetilde{\rho}$ (see Remark 4) derived in Moritz et al. (2016) as

$$\widetilde{C} = O\left(\left(n + b(\kappa_{\max}\widetilde{\kappa}_{\mathrm{H}})^2\right) d\log(1/\epsilon)\right). \tag{30}$$

Compared with $\widetilde{C}$, we observe that our complexity estimate C in (29) is much better, in several aspects. First, the dependence of C on the condition number $\kappa\kappa_{\mathrm{H}}$ is linear, rather than quadratic. The quadratic dependence of $\kappa_{\max}\widetilde{\kappa}_{\mathrm{H}}$ in $\widetilde{C}$ is precisely caused by the additional $\kappa_{\max}\widetilde{\kappa}_{\mathrm{H}}$ in $\widetilde{\rho}$ (see Remark 4). Second, C is independent of the mini-batch size b. The appearance of b in $\widetilde{C}$ is a result of the loose bound on variance of $\mathbf{v}_{s,t}$ (cf. (11) and (12)). Third, the condition number $\kappa\kappa_{\mathrm{H}}$ in C is much more smaller than $\kappa_{\max}\widetilde{\kappa}_{\mathrm{H}}$ in $\widetilde{C}$ for ill-conditioned problems. This is a result of non-uniform sampling of $\mathcal{B}_{s,t}$ and our improved bound on the spectra of $\{\mathbf{H}_r\}_{r\geq 0}$ (see Lemma 4).

Remark 7. As our coordinate transformation framework unifies the design and analysis of stochastic first- and second-order algorithms, we believe that momentum-based acceleration techniques for stochastic first-order methods (Allen-Zhu, 2016; Lin et al., 2015) can be applied to Algorithm 1 as well. (Details are left to future work.) In this case, the dependence on $\kappa\kappa_{\mathrm{H}}$ in C may be further improved to $\sqrt{\kappa\kappa_{\mathrm{H}}}$ (Allen-Zhu, 2016).

6 ACCELERATION STRATEGIES

In this section, we propose three practical acceleration strategies. We follow the notations in Section 3 and Algorithm 1. As will be shown in Section 7.1, all of these strategies result in faster convergence in practice. For first and second strategies, we also provide their theoretical analyses in Propositions 2 and 3 respectively. See Sections S-5 and S-6 for the proofs of these two propositions.

6.1 GEOMETRIC SAMPLING AND AVERAGING

Instead of choosing $\mathbf{x}^{s+1}$ according to option I or II in Algorithm 1, inspired by Konečný and Richtárik (2013), we can introduce a "forgetting" effect by considering two other sampling or averaging schemes:

option III: Sample τ_s randomly from $[m]$ according to the distribution $Q \triangleq (\beta^{m-1}/c, \beta^{m-2}/c, \ldots, 1/c)$ and set $\mathbf{x}^{s+1} := \mathbf{x}_{s,\tau_s}$,

option IV: $\mathbf{x}^{s+1} := \frac{1}{c}\sum_{t=1}^{m}\beta^{m-t}\mathbf{x}_{s,t}$,

where $0 < \beta \leq 1 - \eta\gamma\overline{\mu} < 1$ and the normalization constant $c \triangleq \sum_{t=1}^{m}\beta^{m-t}$. Since $\beta \in (0,1)$, we observe that in both options III and IV, more recent iterates (i.e., iterates $\mathbf{x}_{s,t}$ with larger indices t) will have larger contributions to $\mathbf{x}^{s+1}$. Theoretically, these two schemes can be analyzed in a unified manner, as shown in the following proposition.

Proposition 2. *In Algorithm 1, if we choose $\eta < \min\{b/12, 1\}/(\Gamma\overline{L})$ and m sufficiently large, then with either option III or IV, we have*

$$\mathbb{E}\left[f(\mathbf{x}^s) - f(\mathbf{x}^*)\right] \leq \overline{\rho}^{\,s}\left(f(\mathbf{x}^0) - f(\mathbf{x}^*)\right), \text{ where} \tag{31}$$

$$\overline{\rho} \triangleq \frac{b}{\gamma\overline{\mu}c'\eta\left(b - 4\eta\Gamma\overline{L}/(1-\eta\gamma\overline{\mu})\right)} + \frac{4\eta\Gamma\overline{L}}{b - 4\eta\Gamma\overline{L}/(1-\eta\gamma\overline{\mu})}\left(1 + \frac{1}{c'}\right) < 1 \tag{32}$$

and $c' \triangleq c/(1-\eta\gamma\overline{\mu})^m$.

Remark 8. In the literature (Gower et al., 2016; Moritz et al., 2016), usually option I or II (in Algorithm 1) is analyzed theoretically to prove that the stochastic L-BFGS algorithms therein converge linearly. However, for faster convergence in practice, $\mathbf{x}^{s+1}$ is chosen to be the last inner iterate $\mathbf{x}_{s,m}$. However, the latter is not amenable to linear convergence analysis. This *gap between theory and practice* is filled in by our geometric sampling or averaging scheme, i.e., option III or IV. Specifically, as shown in

Figure 2, our scheme not only yields linear convergence in theory, but also performs as well as the "last inner iterate" scheme in practice.

6.2 SUBSAMPLED GRADIENT STABILIZATION

In Algorithm 1, at the beginning of each outer iteration s, we compute a full gradient $\mathbf{g}_s$ to stabilize the subsequent (inner) iterations. Inspired by Harikandeh et al. (2015), we propose a strategy that only computes a subsampled gradient $\widetilde{\mathbf{g}}_s$ at the start of each outer iteration s. Specifically, we uniformly sample a subset $\widetilde{\mathcal{B}}_s$ of $[n]$ with size $\widetilde{b}_s$ *without replacement* and then form $\widetilde{\mathbf{g}}_s \triangleq (1/\widetilde{b}_s)\sum_{i\in\widetilde{\mathcal{B}}_s}\nabla f_i(\mathbf{x}^s)$. The size of $\widetilde{\mathcal{B}}_s$, namely $\widetilde{b}_s$, increases with the index s until it reaches n. By judiciously choosing $\widetilde{b}_s$, we can show that the resulting algorithm still enjoys linear convergence with rate $\overline{\rho}$, when integrated with the geometric sampling/averaging scheme in Section 6.1. Before we formally state this result in Proposition 3, we first make an assumption in additional to Assumptions 1 and 2.

Assumption 5. *The inner iterates $\{\mathbf{x}_{s,t}\}_{s\geq 0, t\in[m]}$ generated by the modified algorithm in Section 6.2 are bounded almost surely, i.e., there exists $B<\infty$ such that for any $s\geq 0$ and $t\in[m]$, $\|\mathbf{x}_{s,t}-\mathbf{x}^*\|\leq B$.*

Proposition 3. *Let Assumptions 1, 2 and 5 hold. For any $\xi>0$ and $S\in\mathbb{N}$, and for any $s\in(S]$, if we choose $\widetilde{b}_s \geq \underline{\widetilde{b}}_s \triangleq nS^2\alpha_s/(S^2\alpha_s+(n-1)\xi^2\overline{\rho}^{2s})$, where $\alpha_s \triangleq 1/n\sum_{i=1}^n\|\nabla f_i(\mathbf{x}^s)\|^2$, we have*

$$\mathbb{E}\left[f(\mathbf{x}^s)-f(\mathbf{x}^*)\right] \leq \overline{\rho}^s\Bigg\{\left(f(\mathbf{x}^0)-f(\mathbf{x}^*)\right) + \left(1+\frac{1}{c'}\right)\frac{\xi b}{b-4\Gamma\overline{L}\eta/(1-\eta\gamma\overline{\mu})}\left(\kappa_{\mathrm{H}}^{1/2}B+\eta\Gamma\xi\right)\Bigg\}.$$

Remark 9. Several remarks are in order. First, we remark that assumptions involving almost sure boundedness of iterates (e.g., Assumption 5) have appeared many times in the literature of stochastic optimization (Borkar, 2008; Harikandeh et al., 2015; Hu et al., 2009) and are always observed to hold in our experiments. Second, under this assumption, we can show that $\{\alpha_s\}_{s\geq 0}$ are bounded almost surely using Lipschitz continuity of ∇f_i, for any $i\in[n]$ in Assumption 2. Consequently, there exists $B'<\infty$ such that $\alpha_s\leq B'$ for any $s\geq 0$ and hence

$$\underline{\widetilde{b}}_s \leq \frac{nS^2B'}{S^2B'+(n-1)\xi^2\overline{\rho}^{2s}} \tag{33}$$

$$\leq \frac{nS^2B'}{(n-1)\xi^2}\left(\overline{\rho}^{-2}\right)^s. \tag{34}$$

As a sanity check, we observe that (33) increases to n as $s\to\infty$. By further upper bounding (33) by (34), we obtain a practical rule to select $\widetilde{b}_s$. Namely, it suffices to choose $\widetilde{b}_s=\min\{\zeta\upsilon^s,n\}$, for some constants $\zeta>0$ and $\upsilon>1$. As shown in Section 7, this rule works well in practice. Third, for any $\epsilon>0$, we can choose $S\in\mathbb{N}$ such that our algorithm achieves ϵ-suboptimality, i.e., $\mathbb{E}\left[f(\mathbf{x}^S)-f(\mathbf{x}^*)\right]<\epsilon$.

6.3 SMALL APPROXIMATE HESSIANS

In additional to high dimensionality and large size, *sparsity* is also a typical attribute for modern data, i.e., many feature vectors only have a few nonzero entries. For ERM problems (2) (with Tikhonov regularization), this implies that the Hessian of f in (1),

$$\nabla^2 f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^n \ell''(\mathbf{a}_i^T\mathbf{x},b_i)(\mathbf{a}_i\mathbf{a}_i^T)+\lambda\mathbf{I}, \tag{35}$$

tends to be sparse. Based on this observation, we propose a strategy that aims to approximate $\nabla^2 f(\mathbf{x})$ by several smaller Hessian matrices and update them efficiently. For sparse data, collecting curvature information via smaller dense Hessians can be more effective than directly manipulating the high-dimensional sparse Hessian. As a result, the algorithm converges faster in practice (see Figure 4).

Before describing our strategy, we first introduce some notations. We partition $[n]$ into K groups, and denote the set of partitions as $\mathcal{P}\triangleq\{\mathcal{P}_1,\ldots,\mathcal{P}_K\}$. For any $i\in[K]$, we define $\mathcal{S}_i\triangleq\cup_{j\in\mathcal{P}_i}\mathrm{supp}(\mathbf{a}_j)$, where $\mathrm{supp}(\mathbf{a}_j)$ denotes the support of the vector $\mathbf{a}_j$. We define $d_i\triangleq|\mathcal{S}_i|$ and denote the elements in $\mathcal{S}_i$ as $\{s_{i,1},\ldots,s_{i,d_i}\}$. We also define $F_i=\sum_{j\in\mathcal{P}_i}f_j$ so that $f=\frac{1}{n}\sum_{i=1}^K F_i$. We define a *projection matrix* $\mathbf{U}_i\in\mathbb{R}^{d_i\times d}$ such that for any $p\in[d_i]$ and $q\in[d]$, $u_{pq}=1$ if $q=s_{i,p}$ and 0 otherwise. Accordingly, for any $l\in\mathcal{P}_i$, define a function $\phi_{i,l}:\mathbb{R}^{d_i}\to\mathbb{R}$ such that $f_l\triangleq\phi_{i,l}\circ\mathbf{U}_i$. Note that $\phi_{i,l}$ is uniquely defined by the definition $\mathbf{U}_i$. Also define $\phi_i\triangleq\sum_{l\in\mathcal{P}_i}\phi_{i,l}$. Therefore,

$$\nabla^2 f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^K \mathbf{U}_i^T\nabla^2\phi_i(\mathbf{U}_i\mathbf{x})\mathbf{U}_i,\ \forall\,\mathbf{x}\in\mathbb{R}^d. \tag{36}$$

We now describe our strategy. In Algorithm 1, for any $i\in[K]$ and any $j\in\{r-M'+1,\ldots,r\}$, define correction pairs $\mathbf{s}_{j,i}\triangleq\mathbf{U}_i(\overline{\mathbf{x}}_j-\overline{\mathbf{x}}_{j-1})$ and $\mathbf{y}_{j,i}\triangleq\sum_{l\in\mathcal{T}_{r,i}}\nabla^2\phi_{i,l}(\mathbf{U}_i\overline{\mathbf{x}}_j)\mathbf{s}_{j,i}$, where $\mathcal{T}_{r,i}$ with size b_{H}/K is uniformly sampled from $\mathcal{P}_i$. Accordingly, define $\mathbf{S}_{r,i}\triangleq[\mathbf{s}_{r-M'+1,i},\ldots,\mathbf{s}_{r,i}]$, $\mathbf{Y}_{r,i}\triangleq[\mathbf{y}_{r-M'+1,i},\ldots,\mathbf{y}_{r,i}]$. Instead of storing $\mathcal{H}_r$, we only store matrices $\{(\mathbf{S}_{r,i},\mathbf{Y}_{r,i})\}_{i=1}^K$. To reconstruct approximation $\mathbf{B}_{r,i}$ for each $\nabla^2\phi_i$ at $\mathbf{U}_i\overline{\mathbf{x}}_r$, as usual, we apply M' BFGS updates (S-51) (using the correction pairs stored in $\mathbf{S}_{r,i}$ and $\mathbf{Y}_{r,i}$) to $\mathbf{B}_{r,i}^{(0)}\triangleq\delta_{r,i}\mathbf{I}$, where $\delta_{r,i}\triangleq\|\mathbf{y}_{r,i}\|^2/\mathbf{s}_{r,i}^T\mathbf{y}_{r,i}$. This procedure can be implemented *efficiently* via a *compact representation* (Nocedal and Wright, 2006), i.e.,

$$\mathbf{B}_{r,i}=\delta_{r,i}\mathbf{I}-\mathbf{W}_{r,i}\mathbf{M}_{r,i}^{-1}\mathbf{W}_{r,i}^T, \tag{37}$$

$$\mathbf{M}_{r,i}\triangleq\begin{bmatrix}\delta_{r,i}{\mathbf{S}_{r,i}}^T\mathbf{S}_{r,i} & \mathbf{L}_{r,i}\\ {\mathbf{L}_{r,i}}^T & -\mathbf{D}_{r,i}\end{bmatrix}, \tag{38}$$

where $\mathbf{W}_{r,i} \triangleq [\delta_r^{(i)}\mathbf{S}_r^{(i)}, \mathbf{Y}_r^{(i)}]$ and $\mathbf{L}_{r,i}$ and $\mathbf{D}_{r,i}$ are the lower triangular matrix (excluding diagonal) and diagonal matrix of $\mathbf{S}_{r,i}^T\mathbf{Y}_{r,i}$ respectively. Analogous to (36), we define the approximation of $\nabla^2 f$ at $\mathbf{U}_i\bar{\mathbf{x}}_r$, denoted as $\mathbf{B}_r$, as

$$\mathbf{B}_r \triangleq \frac{1}{n}\sum_{i=1}^{K}\mathbf{U}_i^T\mathbf{B}_{r,i}\mathbf{U}_i. \tag{39}$$

We remark that the strong convexity of each function f_i (see Assumption 2), together with the full-row-rank property of $\mathbf{U}_i$, ensures $\nabla^2\phi_i \succ 0$ on $\mathbb{R}^{d_i}$. This implies *positive curvature* $\mathbf{s}_{r,i}^T\mathbf{y}_{r,i} > 0$ and hence the positive definiteness of $\mathbf{B}_{r,i}$, for any $r \geq 0$ and $i \in [K]$. As a result, $\mathbf{B}_r \succ 0$ on $\mathbb{R}^d$. This suggests the usage of the *conjugate gradient* (CG) method to compute the search direction at time (s,t), namely $\mathbf{p}_{s,t} \triangleq -\mathbf{B}_r^{-1}\mathbf{v}_{s,t}$, via solving the positive definite system $\mathbf{B}_r\mathbf{p}_{s,t} = -\mathbf{v}_{s,t}$. In particular, for any $\mathbf{z} \in \mathbb{R}^d$, $\mathbf{B}_r\mathbf{z}$ and $\mathbf{z}^T\mathbf{B}_r\mathbf{z}$ in CG can be computed very efficiently using (37) and (39). For example, $\mathbf{z}^T\mathbf{B}_r\mathbf{z}$ equals

$$\frac{1}{n}\sum_{i=1}^{K}\delta_{r,i}\left\|\mathbf{z}_i\right\|^2 - (\mathbf{W}_{r,i}^T\mathbf{z}_i)^T\mathbf{M}_{r,i}^{-1}(\mathbf{W}_{r,i}^T\mathbf{z}_i), \tag{40}$$

where $\mathbf{z}_i \triangleq \mathbf{U}_i\mathbf{z}$. We observe that the total computational cost in (40) is $O(M'(M'^2+d'))$, where $d' \triangleq \sum_{i=1}^{K} d_i$. For sparse data, usually $d' = O(d)$, so this cost is still linear in d. In addition, we can compute (40) *in parallel* across $i \in [K]$. (Intuitively, this amounts to *collecting curvature* from each function ϕ_i in parallel.) In this case, the computational time will be greatly reduced to $O(M'(M'^2 + \max_i d_i))$. Since typically $\max_i d_i \ll d$, the computational savings from *parallel curvature collection* can be significant.[9]

7 NUMERICAL EXPERIMENTS

We consider two ERM problems, including logistic regression (with Tikhonov regularization) and ridge regression. For logistic regression, $b_i \in \{-1, 1\}$ and

$$f_i(\mathbf{x}) \triangleq \log\left(1 + e^{-b_i(\mathbf{a}_i^T\mathbf{x})}\right) + \frac{\lambda}{2}\left\|\mathbf{x}\right\|^2, \ \forall\, i \in [n]. \tag{41}$$

For ridge regression, $b_i \in \mathbb{R}$ and

$$f_i(\mathbf{x}) \triangleq \left(\mathbf{a}_i^T\mathbf{x} - b_i\right)^2 + \frac{\lambda}{2}\left\|\mathbf{x}\right\|^2, \ \forall\, i \in [n]. \tag{42}$$

Simple calculations show the smoothness parameters L_i for (41) and (42) are $\left\|\mathbf{a}_i\right\|^2/4 + \lambda$ and $2\left\|\mathbf{a}_i\right\|^2 + \lambda$ respectively. In both (41) and (42), we choose $\lambda = 1/n$, following the convention in the literature (e.g., Konečný et al. (2016)).

We test logistic and ridge regression on `rcv1.binary` and `E2006-tfidf` datasets respectively (Chang and Lin, 2011). (In the sequel we abbreviate them as `rcv1` and `E2006`.) The size-dimension statistics (n, d) for these two datasets are $(20242, 47236)$ and $(16087, 15036)$ respectively. From (35), we define a sparsity estimate of $\nabla^2 f$ at any $\mathbf{x} \in \mathbb{R}^d$ as $\sigma \triangleq \left|\mathrm{supp}(\mathbf{A}\mathbf{A}^T + \mathbf{I})\right|/d^2$, where $\mathbf{A} \triangleq [\mathbf{a}_1 \ldots, \mathbf{a}_n]$ is the data matrix. For `rcv1` and `E2006`, σ equals 0.0154 and 0.0404 respectively. For both datasets, the norms of all feature vectors $\{\mathbf{a}_i\}_{i=1}^n$ have been normalized to unit. Since the the smoothness parameters L_i for both ERM problems only depend on $\|\mathbf{a}_i\|$ and λ, we have $L_i = L_j$ for any $i, j \in [n]$. Thus the nonuniform distribution P in Section 3 becomes uniform, and the advantage of nonuniform sampling of $\mathcal{B}_{s,t}$ cannot be observed.

To estimate the global optimum $\mathbf{x}^*$ as ground truth, we use batch L-BFGS-B algorithm (Zhu et al., 1997). We randomly initialize $\mathbf{x}^0$ according to a scaled standard normal distribution.[10] We use the number of data passes (i.e., number of data points accessed divided by n), rather than time, to measure the efficiency of algorithms. This has been a well-established convention in the literature since execution time highly depends on implementation of algorithms.

Finally we describe the parameter setting. We set the mini-batch size $b = \sqrt{n}$, Hessian update period $\Upsilon = 10$ and the memory parameter $M = 10$. We set $b_{\mathrm{H}} = b\Upsilon$ so that the computation for $\mathbf{y}_r$ can be amortized to each inner iteration. We set the number of inner iterations $m = n/b$, so that each outer iteration will access $2n$ data points. Lastly, we set $\eta = 1 \times 10^{-2}$. From Figure 1, we observe that when η is too large, e.g., $\eta = 0.1$, Algorithm 1 only converges sublinearly; whereas when η is too small, e.g., $\eta = 1 \times 10^{-3}$, Algorithm 1 converges linearly but slowly. This corresponds well to our theoretical analysis in Theorem 1, which indicates that when η falls below a threshold, ρ increases as η decreases. For both ERM problems, we find $\eta = 1 \times 10^{-2}$ yields fast linear convergence.

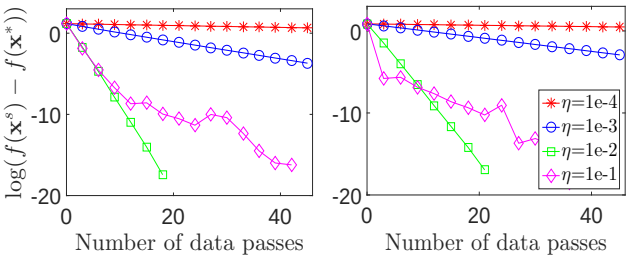
7.1 PERFORMANCE OF ACCELERATION STRATEGIES

We first examine the performance of Algorithm 1 with different schemes of choosing $\mathbf{x}^{s+1}$. We consider five schemes in total, including (a) uniform sampling (option I), (b) uniform averaging (option II), (c) geometric sampling (option III), (d) geometric averaging (option IV) and (e) last inner iterate (in Remark 8). For options III and IV, we set $\beta = 1/2$. From Figure 2, we observe that options III and IV perform as well as the "last inner iterate" scheme, on both ERM problems, and outperform the schemes based on uniform sampling/averaging significantly. For all the subsequent experiments, we use option IV to select $\mathbf{x}^{s+1}$.

We next compare the performance of Algorithm 1 with and without using the subsampled gradient stabilization strategy in Section 6.2. As suggested by Remark 9, we choose

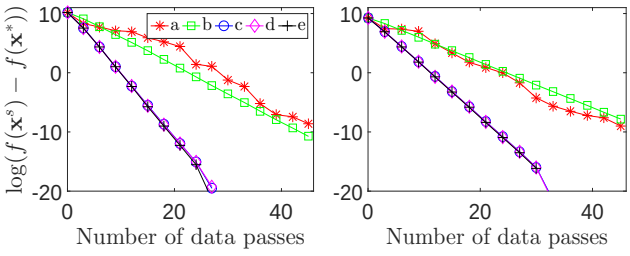
[9] The memory parameter M (note that $M' \leq M$) is usually set to a small constant, e.g., 5 or 10. Thus it has much less effects on the computational complexity compared to d or $\max_i d_i$.

[10] The performance of our algorithms were observed to be insensitive to the initialization of $\mathbf{x}^0$.



(a) logistic regression (`rcv1`) (b) ridge regression (`E2006`)

Figure 1: Log suboptimality versus number of passes through data of Algorithm 1 with different step sizes η.

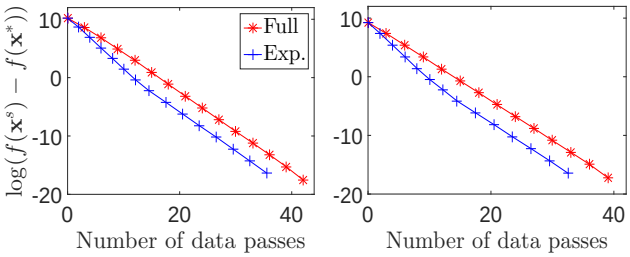


(a) logistic regression (`rcv1`) (b) ridge regression (`E2006`)

Figure 2: Comparison of Algorithm 1 with different selection schemes for $\mathbf{x}^{s+1}$.

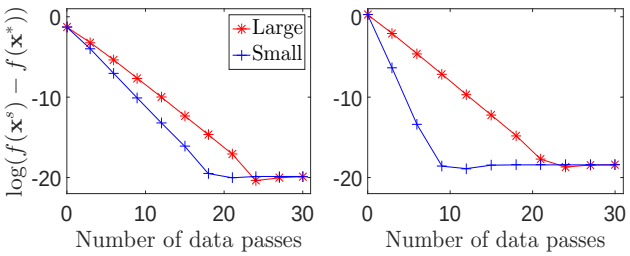
$\widetilde{b}_s = \min\{\zeta\upsilon^s, n\}$, where $\zeta = n/\upsilon^q$, $\upsilon = 3$ and $q = 8$. That is, the number of component gradients in $\widetilde{\mathbf{g}}_s$ exponentially increases in the first $p = 8$ outer iterations and then remains at n. From Figure 3, we observe that this simple parameter selection method works well on both ERM problems, especially in the initial phase (when s is small). In addition, we also observe when s is large, both algorithms have almost the same (linear) convergence rates. This corroborates our analysis in Proposition 3.

We finally compare the performance of Algorithm 1 with and without using the low-dimensional approximate Hessian strategy in Section 6.3. We set the number of partitions $K = 5$ and partition $[n]$ evenly and randomly. From Figure 4, we observe that our strategy leads to improvements of convergence on both logistic and ridge regression problems, and the improvement on the latter is *very significant*. Moreover, our strategy *preserves the linear convergence* of Algorithm 1 on both problems. As discussed in Section 6.3, our strategy can be even more efficient in scenarios where parallel computational resources are available.



(a) logistic regression (`rcv1`) (b) ridge regression (`E2006`)

Figure 3: Comparison of Algorithm 1 without (Full) and with (Exp.) using the strategy in Section 6.2.

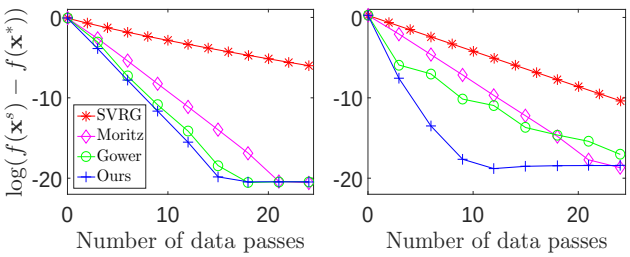


(a) logistic regression (`rcv1`) (b) ridge regression (`E2006`)

Figure 4: Comparison of Algorithm 1 without (Large) and with (Small) using the strategy in Section 6.3.

7.2 COMPARISON TO OTHER ALGORITHMS

We combine all of our acceleration strategies in Section 6 and compare the resulting algorithm with three benchmarking algorithms, including SVRG in Johnson and Zhang (2013) and two state-of-the-art stochastic L-BFGS algorithms in Moritz et al. (2016) and Gower et al. (2016). For the algorithm in Gower et al. (2016), we focus on its variant (b), since it consistently outperformed other variants in experiments. We tested all the three benchmarking algorithms on both ERM problems with different step sizes $\eta \in \{10^{-4}, 10^{-3}, \ldots, 1\}$ and selected the best η for each algorithm. The outer iterate $\mathbf{x}^{s+1}$ in all these algorithms are selected via "the last iterate" scheme. From Figure 5, we observe that on both ERM problems, our algorithm has faster convergence compared to all the benchmarking algorithms. The improvement of convergence is particularly significant on the ridge regression problem. This observation is consistent with our observations in Section 7.1.
Future work and open problems: see Section S-7.



(a) logistic regression (`rcv1`) (b) ridge regression (`E2006`)

Figure 5: Comparison of our algorithm (Ours) with benchmarking algorithms (SVRG, Moritz and Gower).

References

Z. Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. arXiv:1603.05953, 2016.

A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-newton stochastic gradient descent. *J. Mach. Learn. Res.*, 10:1737–1754, Dec. 2009.

V. S. Borkar. *Stochastic approximation: a dynamical systems viewpoint*. Cambridge, 2008.

L. Bottou. Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*. Cambridge University Press, 1998.

L. Bottou and Y. LeCun. Large scale online learning. In *Proc. NIPS*, pages 1361–1368. Vancouver, BC, Canada, Dec. 2004.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM J. Optim.*, 26(2):1008–1031, 2016.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2: 27:1–27:27, 2011.

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proc. NIPS*, pages 1646–1654, Montréal, Québec, Canada, 2014.

D. Goldfarb. A family of variable metric updates derived by variational means. *Math. Comput.*, 24(109):23–26, 1970.

R. M. Gower, D. Goldfarb, and P. Richtárik. Stochastic block BFGS: squeezing more curvature out of data. In *Proc. ICML*, pages 1869–1878, New York City, NY, USA, June 2016.

R. Harikandeh, M. O. Ahmed, A. Virani, M. Schmidt, J. Konečný, and S. Sallinen. Stopwasting my gradients: Practical svrg. In *Proc. NIPS*, pages 2251–2259. Montréal, Quebec, Canada, 2015.

C. Hu, W. Pan, and J. T. Kwok. Accelerated gradient methods for stochastic optimization and online learning. In *Porc. NIPS*, pages 781–789. Vancouver, B.C., Canada, 2009.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proc. NIPS*, pages 315–323, Lake Tahoe, Nevada, USA, 2013.

J. Konečný and P. Richtárik. Semi-stochastic gradient descent methods. arXiv:1312.1666, 2013.

J. Konečný, J. Liu, P. Richtárik, and M. Takáč. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE J. Sel. Top. Signal Process.*, 10(2):242–255, 2016.

H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Proc. NIPS*, pages 3384–3392, 2015.

D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45 (3), Dec. 1989.

A. Mokhtari and A. Ribeiro. Global convergence of online limited memory BFGS. *J. Mach. Learn. Res.*, 16(1): 3151–3181, Jan 2015.

P. Moritz, R. Nishihara, and M. I. Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *Proc. AISTATS*, pages 249–258, Cadiz, Spain, May 2016.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Math. Program.*, 162(1):83–112, 2017.

N. N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-Newton method for online convex optimization. In *Proc. ATSTATS*, San Juan, Puerto Rico, March 2007.

J. Sohl-Dickstein, B. Poole, and S. Ganguli. Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods. In *Proc. ICML*, pages 604–612, Beijing, China, June 2014.

D. Williams. *Probability with Martingales*. Cambridge University Press, 1991.

L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM J. Optim.*, 24(4):2057–2075, 2014.

C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23 (4), Dec 1997.