Stochastic Block BFGS: Squeezing More Curvature out of Data

Robert M. Gower GOWERROBERT@GMAIL.COM
Donald Goldfarb GOLDFARB@COLUMBIA.EDU
Peter Richtárik PETER.RICHTARIK@ED.AC.UK

Abstract

We propose a novel limited-memory stochastic block BFGS update for incorporating enriched curvature information in stochastic approximation methods. In our method, the estimate of the inverse Hessian matrix that is maintained by it, is updated at each iteration using a sketch of the Hessian, i.e., a randomly generated compressed form of the Hessian. We propose several sketching strategies, present a new quasi-Newton method that uses stochastic block BFGS updates combined with the variance reduction approach SVRG to compute batch stochastic gradients, and prove linear convergence of the resulting method. Numerical tests on large-scale logistic regression problems reveal that our method is more robust and substantially outperforms current state-of-the-art methods.

1. Introduction

We design a new stochastic variable-metric (quasi-Newton) method—the stochastic block BFGS method—for solving the Empirical Risk Minimization (ERM) problem:

$$\min_{x\in\mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n f_i(x). \tag{1}$$

We assume the loss functions $f_i : \mathbb{R}^d \to \mathbb{R}$ to be convex and twice differentiable and focus on the setting where the number of *data points (examples)* (n) is very large.

To solve (1), we employ iterative methods of the form

$$x_{t+1} = x_t - \eta H_t g_t, \tag{2}$$

where $\eta > 0$ is a stepsize, $g_t \in \mathbb{R}^d$ is an estimate of the gradient $\nabla f(x_t)$ and $H_t \in \mathbb{R}^{d\times d}$ is a positive definite estimate of the inverse Hessian matrix, that is $H_t \approx \nabla^2 f(x_t)^{-1}$. We refer to H_t as the *metric matrix* [1].

The most successful classical optimization methods fit the format (2), such as gradient descent ($H_t = I$), Newton's method $\left(H_t = \nabla^2 f(x_t)^{-1}\right)$, and the quasi-Newton methods $\left(H_t \approx \nabla^2 f(x_t)^{-1}\right)$; all with $g_t = \nabla f(x_t)$. The difficulty in our setting is that the large number of data points makes the computational costs of a single iteration of these classical methods prohibitively expensive.

To amortize these costs, the current state-of-the-art methods use subsampling, where g_t and H_t are calculated using only derivatives of the subsampled function

$$f_S(x) \stackrel{\text{def}}{=} \frac{1}{|S|}\sum_{i\in S} f_i(x),$$

where $S \subseteq [n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$ is a subset selected uniformly at random. Using the subsampled gradient $\nabla f_S(x)$ as a proxy for the gradient is the basis for the stochastic gradient descent (SGD) method, but also for many successful variance reduced methods (Schmidt et al., 2013; Shalev-Shwartz & Zhang, 2013; Johnson & Zhang, 2013; Konečný & Richtárik, 2014; Defazio et al., 2014; Shalev-Shwartz & Zhang, 2016; Konečný et al., 2016) that make use of the subsampled gradient in calculating g_t.

Recently, there has been an effort to calculate H_t using subsampled Hessian matrices $\nabla^2 f_T(x_t)$, where $T \subseteq [n]$ is sampled uniformly at random and independently of S (Erdogdu & Montanari, 2015; Roosta-Khorasani & Mahoney, 2016).

One fairly successful solution to these issues (Byrd et al., 2015; Moritz et al., 2016) is to use a single Hessian-vector product $\nabla^2 f_T(x_t)v$, where $v \in \mathbb{R}^d$ is a suitably selected vector, to update H_t by the limited-memory (L-BFGS) (Nocedal, 1980) version of the classical BFGS (Broyden, 1967; Fletcher, 1970; Goldfarb, 1970; Shanno, 1971) method. Calculating this Hessian-vector product can be done inexpensively using the directional

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

[1] Methods of the form (2) can be seen as estimates of a gradient descent method under the metric defined by H_t. Indeed, let $\langle x, y\rangle_{H_t} \stackrel{\text{def}}{=} \langle H_t^{-1}x, y\rangle$ for any $x, y \in \mathbb{R}^d$ denote an inner product, then the gradient of $f(x_t)$ in this metric is $H_t\nabla f(x_t)$.

derivative

$$\nabla^2 f_T(x_t)v = \frac{d}{d\alpha}\nabla f_T(x_t + \alpha v)\Big|_{\alpha=0}. \quad (3)$$

In particular, when using automatic differentiation techniques (Christianson, 1992; Griewank & Walther, 2008) or backpropagation on a neural network (Pearlmutter, 1994), evaluating the above costs at most five times as much as the cost of evaluating the subsampled gradient $\nabla f_T(x_t)$.

Using only a single Hessian-vector product to update H_t yields only a very limited amount of curvature information, and thus may result in an ineffective metric matrix. The block BFGS method addresses this issue.

The starting point for the development of the block BFGS method is the simple observation that, ideally, we would like the metric matrix H_t to satisfy the inverse equation

$$H_t \nabla^2 f_T(x_t) = I,$$

since then H_t would be the inverse of an unbiased estimate of the Hessian. But solving the inverse equation is computationally expensive. So instead, we propose that H_t satisfy a *sketched* version of this equation, namely

$$H_t \nabla^2 f_T(x_t) D_t = D_t, \quad (4)$$

where $D_t \in \mathbb{R}^{d\times q}$ is a randomly generated matrix which has relatively few columns ($q \ll d$). The *sketched subsampled Hessian* $\nabla^2 f_T(x_t) D_t$ can be calculated efficiently through q directional derivatives of the form (3).

Note that (4) has possibly an infinite number of solutions, including the inverse of $\nabla^2 f_T(x_t)$. To determine H_t uniquely, we maintain a previous estimate $H_{t-1} \in \mathbb{R}^{d\times d}$ and project H_{t-1} onto the space of symmetric matrices that satisfy (4). The resulting update, applied to H_{t-1} to arrive at H_t, is the block BFGS update.

In the remainder of the paper we detail the block BFGS update, present a new limited-memory block BFGS update and introduce several new sketching strategies. We conclude by presenting the results of numerical tests of a method that combines the limited-memory block BFGS update with the SVRG method of Johnson & Zhang (2013), and demonstrate that our new method yields dramatically better results when compared to SVRG, or the SVRG method coupled with the classical L-BFGS update as proposed by Moritz et al. (2016).

1.1. Contributions

This paper makes five main contributions.

(a) New metric learning framework. We develop a *stochastic block BFGS update*[2] for approximately tracking the inverse of the Hessian of f. This technique is novel in two ways: the update is more flexible than the traditional BFGS update as it works by employing the actions of a subsampled Hessian on a *set* of *random vectors* rather than just on a single deterministic vector, as is the case with standard BFGS. That is, we use *block sketches* of the subsampled Hessian.

(b) Stochastic block BFGS method. Our block BFGS update is capable of incorporating *enriched* second-order information into gradient based stochastic approximation methods for solving problem (1). In this paper we illustrate the power of this strategy in conjunction with the strategy employed in the SVRG method of (Johnson & Zhang, 2013) for computing variance-reduced stochastic gradients. We prove that the resulting combined method is linearly convergent and empirically demonstrate its ability to substantially outperform current state-of-the-art methods.

(c) Limited-memory method. To make the stochastic block BFGS method applicable to large-scale problems, we devise a new limited-memory variant of it. As is the case for L-BFGS (Nocedal, 1980), our limited-memory approach allows for a user-defined amount of memory to be set aside. But unlike L-BFGS, our limited-memory approach allows one to use the available memory to store more recent curvature information, encoded in sketches of previous Hessian matrices. This development of a new limited block BFGS method should also be of general interest to the optimization community.

(d) Factored form. We develop a limited-memory factored form of the block BFGS update. No such method existed before. Factored forms are important as they can be used to enforce positive definiteness of the metric even in the presence numerical imprecision. Furthermore, we use the factored form in calculating new sketching matrices.

(e) Adaptive sketching. Not only can sketching be used to tackle the large dimensions of the Hessian, but it can also simultaneously precondition the inverse equation (4). We present a self-conditioning (i.e., adaptive) sketching that also makes use of the efficient factored form of the block BFGS method developed earlier. We also present a sketch based on using previous search directions. Adaptive sketching can in practice lead to significant speedup in comparison with sketching from a fixed distribution.

[2] In this paper we use the word "update" to denote metric learning, i.e., an algorithm for updating one positive definite matrix into another.

1.2. Background and Related Work

The first stochastic variable-metric method developed that makes use of subsampling was the online L-BFGS method (Schraudolph et al., 2007). In this work the authors adapt the L-BFGS method to make use of subsampled gradients, among other empirically verified improvements. The regularized BFGS method (Mokhtari & Ribeiro, 2014; 2015) also makes use of stochastic gradients, and further modifies the BFGS update by adding a regularizer to the metric matrix.

The first method to use subsampled Hessian-vector products in the BFGS update, as opposed to using differences of stochastic gradients, was the SQN method (Byrd et al., 2015). Recently, (Moritz et al., 2016) propose combining SQN with SVRG. The resulting method performs very well in numerical tests. In our work we combine a novel *stochastic block BFGS* update with SVRG, and prove linear convergence. The resulting method is more versatile and superior in practice to SQN as it can capture more useful curvature information.

The update formula that we refer to as the block BFGS update has a rather interesting background. The formula first appeared in 1983 in unpublished work (Schnabel, 1983) on designing quasi-Newton methods that make use of multiple secant equations. Schnabel's method requires several modifications that stem from the lack of symmetry and positive definiteness of the resulting update. Later, and completely independently, the block BFGS update appears in the domain decomposition literature (Mandel & Brezina, 1993) as a preconditioner, where it is referred to as the balancing preconditioner. In that work, the motivation and deduction are very different from those used in the quasi-Newton literature; for instance, no variational interpretation is given for the method. The balancing preconditioner was subsequently taken out of the PDE context and tested as a general purpose preconditioner for solving a single linear system and systems with changing right hand side (Gratton et al., 2011). Furthermore, Gratton et al. (2011) present a factored form of the update in a different context, which we adapt for limited-memory implementation. Finally, and again independently, a family of block quasi-Newton methods that includes the block BFGS is presented by Gower & Gondzio (2014) through a variational formulation and by Hennig (2015) using Bayesian inference.

2. Stochastic Block BFGS Update

The stochastic block BFGS update, applied to H_{t-1} to obtain H_t, is defined by the weighted projection

$$H_t = \arg\min_{H\in\mathbb{R}^{n\times n}} \|H - H_{t-1}\|_t^2 \quad \text{subject to} \quad H\nabla^2 f_T(x_t)D_t = D_t,\ H = H^T, \quad (5)$$

where $\|H\|_t^2 \overset{\text{def}}{=} \mathbf{Tr}\left(H\nabla^2 f_T(x_t)H^T\nabla^2 f_T(x_t)\right)$, and $\mathbf{Tr}(\cdot)$ denotes the trace. The method is stochastic since D_t is a random matrix. The constraint in (5) serves as a *fidelity* term, enforcing that H_t satisfies a sketch of the inverse equation (4). The objective in (5) acts as a *regularizer*, and ensures that the difference between H_t and H_{t-1} is a low rank update. The choice of the objective is special yet in another sense: recent results show that if the iteration determined by (5) is applied to a fixed invertible matrix A in place of the subsampled Hessian, then the matrices H_t would converge to the inverse A^{-1} at a linear rate. This is yet one more reason to expect that in our setting the matrices H_t approximately track the inverse Hessian.

The solution to (5) is

$$H_t = D_t\Delta_t D_t^T + \left(I - D_t\Delta_t Y_t^T\right)H_{t-1}\left(I - Y_t\Delta_t D_t\right), \quad (6)$$

where $\Delta_t \overset{\text{def}}{=} (D_t^T Y_t)^{-1}$ and $Y_t \overset{\text{def}}{=} \nabla^2 f_T(x_t)D_t$. This solution was given in (Gower & Gondzio, 2014; Hennig, 2015) and in (Schnabel, 1983) for multiple secant equations.

Note that the stochastic block BFGS (6) yields the same matrix H_t if D_t is replaced by any other matrix $\tilde{D}_t \in \mathbb{R}^{d\times q}$ such that $\mathbf{span}(\tilde{D}_t) = \mathbf{span}(D_t)$. It should also be pointed out that the matrix H_t produced by (6) is not what would be generated by a sequence of q rank-two BFGS updates based on the q columns of D_t. That is, unless the columns of D_t are $\nabla^2 f_T(x_t)$-conjugate, as would be the case if they were generated by the BFGS method applied to a the minimization of a strictly convex quadratic function $x^T\nabla^2 f_T(x_t)x$ using exact line-search.

We take this opportunity to point out that stochastic block BFGS can generate the metric used in the Stochastic Dual Newton Ascent (SDNA) method (Qu et al., 2016) as a special case. Indeed, when $H_{t-1} = 0$, then H_t is given by

$$H_t = D_t(D_t^T\nabla^2 f_T(x_t)D_t)^{-1}D_t^T. \quad (7)$$

When the sketching matrix D_t is a random column submatrix of the identity, then (7) is the positive semidefinite matrix used in calculating the iterates of the SDNA method (Qu et al., 2016). However, SDNA operates in the dual of (1).

3. Stochastic Block BFGS Method

The goal of this paper is to design a method that uses a low-variance estimate of the gradient, but also gradually incorporates *enriched* curvature information. To this end, we propose to combine the stochastic variance reduced gradient (SVRG) approach (Johnson & Zhang, 2013) with our *novel stochastic block BFGS update,* described in the previous section. The resulting method is Algorithm 1.

Algorithm 1 has an outer loop in k and an inner loop in t. In the outer loop, the *outer* iterate $w_k \in \mathbb{R}^d$ and the full gradient $\nabla f(w_k)$ are computed. In the inner loop, both the estimate of the gradient g_t and our metric H_t are updated using the SVRG update and the block BFGS update, respectively.

To form the sketching matrix D_t we employ one of the following three strategies:

a) Gaussian sketch. D_t has standard Gaussian entries sampled i.i.d at each iteration. **b) Previous search directions delayed.** Let us write $d_t = -H_t g_t$ for the search direction used in step t of the method. We store L search directions $D_t = [d_{t+1-L}, \ldots, d_t]$ and then update H_t only once every L inner iterations.

c) Self-conditioning. We sample $C_t \subseteq [d]$ uniformly at random and set $D_t = L_{t-1} I_{:C_t} = [L_{t-1}]_{:C_t}$, where $L_{t-1} L_{t-1}^T = H_{t-1}$ and $I_{:C_t}$ denotes the concatenation of the columns of the identity matrix indexed by a set $C_t \subset [d]$. Thus the sketching matrix is formed with a random subset of columns of a factored form of H_t. The idea behind this strategy is that the ideal sketching matrix should be $D_t = (\nabla^2 f_T(x_t))^{-1/2}$ so that the sketch compresses *and* acts as a preconditioner on the inverse equation (4). It was also shown (Gower & Richtárik, 2015) that this choice of sketching matrix can accelerate the convergence of H_t to the inverse of a fixed matrix. In Section 3.2 we detail how to efficiently maintain and update the factored form.

3.1. Block Limited-Memory BFGS

When d is large, we cannot store the $d \times d$ matrix H_t. Instead, we store M block triples, consisting of previous block curvature pairs and the inverse of their products

$$(D_{t+1-M}, Y_{t+1-M}, \Delta_{t+1-M}), \ldots, (D_t, Y_t, \Delta_t). \quad (8)$$

With these triples we can form the H_t operator implicitly by using a block limited-memory two loop recurrence. To describe this two loop recurrence, let $V_t \overset{\text{def}}{=} I - D_t \Delta_t Y_t^T$.

The block BFGS update (6) with memory parameter M can be expanded as a function of the M curvature triples (8) and of H_{t-M} as

$$\begin{aligned} H_t &= V_t H_{t-1} V_t^T + D_t \Delta_t D_t^T \\ &= V_t \cdots V_{t+1-M} H_{t-M} V_{t+1-M}^T \cdots V_t^T \\ &+ \sum_{i=t}^{t+1-M} V_t \cdots V_{i+1} D_i \Delta_i D_i^T V_{i+1}^T \cdots V_t^T, \end{aligned}$$

Since we do not store H_t for any t, we do not have access to H_{t-M}. In our experiments we simply set $H_{t-M} = I$ to the identity matrix (other, more sophisticated, choices are possible, but we do not explore them further here). Using the above expansion, the action of the operator H_t on a vector v can be efficiently calculated using Algorithm 2.

Algorithm 1 Stochastic Block BFGS Method

inputs: $w_0 \in \mathbb{R}^d$, stepsize $\eta > 0$, $s =$ subsample size, $q =$ sample action size, and variance reduction increment m.
initiate: $H_{-1} = I$
for $k = 0, 1, 2, \ldots$ **do**
  Compute the full gradient $\mu = \nabla f(w_k)$
  Set $x_0 = w_k$
  for $t = 0, \ldots, m-1$ **do**
    Sample $S_t, T_t \subseteq [n]$, independently
    Compute a variance-reduced stochastic gradient
      $g_t = \nabla f_{S_t}(x_t) - \nabla f_{S_t}(w_k) + \mu$
    Form $D_t \in \mathbb{R}^{d \times q}$ so that $\text{rank}(D_t) = q$
    Compute $Y_t = \nabla^2 f_{T_t}(x_t) D_t$
    Compute ${D_t}^T Y_t$ and its Cholesky factorization
      (this implicitly forms $\Delta_t = (D_t^T Y_t)^{-1}$)
    Option I: Use (6) to obtain H_t and set $d_t = -H_t g_t$
    Option II: Compute $d_t = -H_t g_t$ via Algorithm 2
    Set $x_{t+1} = x_t + \eta d_t$
  end for
  Option I: Set $w_{k+1} = x_m$
  Option II: Set $w_{k+1} = x_i$, where i is selected uniformly at random from $[m] = \{1, 2, \ldots, m\}$
end for
output w_{k+1}

Algorithm 2 Block L-BFGS Update (Two-loop Recursion)

inputs: $g_t \in \mathbb{R}^d$, $D_i, Y_i \in \mathbb{R}^{d \times q}$ and $\Delta_i \in \mathbb{R}^{q \times q}$ for $i \in \{t+1-M, \ldots, t\}$.
initiate: $v \leftarrow g_t$
for $i = t, \ldots, t-M+1$ **do**
  $\alpha_i \leftarrow \Delta_i D_i^T v, \quad v \leftarrow v - Y_i \alpha_i$
end for
for $i = t-M+1, \ldots, t$ **do**
  $\beta_i \leftarrow \Delta_i Y_i^T v, \quad v \leftarrow v + D_i(\alpha_i - \beta_i)$
end for
output $H_t g_t \leftarrow v$

The total cost in floating point operations of executing Algorithm 2 is $Mq(4d + 2q)$. In our experiments $M = 5$ and q will be orders of magnitude less than d, typically $q \leq \sqrt{d}$. Thus the cost of applying Algorithm 2 is approximately $O(d^{3/2})$. This does not include the cost of computing the product ${D_t}^T Y_t$ ($O(q^2 d)$ operations) and its Cholesky factorization ($O(q^3)$ operations), which is done outside of Algorithm 2. The two places in Algorithm 2 where multiplication by Δ_i is indicated is in practice performed by solving two triangular systems using the Cholesky factor of ${D_i}^T Y_i$. We do this because it is more numerically stable than explicitly calculating the inverse matrix Δ_i.

3.2. Factored Form

Here we develop a new efficient method for maintaining and updating a *factored form of the metric matrix*. This facilitates the development of a novel idea which we call *self-conditioning sketch.*

Let $L_{t-1} \in \mathbb{R}^{d\times d}$ be invertible such that $L_{t-1}L_{t-1}^T = H_{t-1}$. Further, let $G_t = (D_t^T L_{t-1}^{-T} L_{t-1}^{-1} D_t)^{1/2}$ and $R_t = \Delta_t^{1/2}$. An update formula for the factored form of H_t, i.e., for L_t for which $H_t = L_t L_t^T$, was recently given (in a different context) by Gratton et al. (2011):

$$L_t = V_t L_{t-1} + D_t R_t G_t^{-1} D_t^T L_{t-1}^{-T}. \tag{9}$$

This factored form of H_t is too costly to compute because it requires inverting L_{t-1}. However, if we let $D_t = L_{t-1}I_{:C_t}$, where $C_t \subset [d]$, then (9) reduces to

$$L_t = V_t L_{t-1} + D_t R_t I_{C_t:} \tag{10}$$

which can be computed efficiently. Furthermore, this update of the factored form (10) can be expanded as

$$\begin{aligned} L_t &= V_t\left(V_{t-1}L_{t-2} + D_{t-1}R_{t-1}I_{C_{t-1}:}\right) + D_t R_t I_{C_t:} \\ &= V_t \cdots V_{t+2-M} V_{t+1-M} L_{t-M} \\ &+ V_t \cdots V_{t+2-M} D_{t+1-M} R_{t+1-M} I_{C_{t+1-M}:} \\ &+ D_t R_t I_{C_t:} \end{aligned} \tag{11}$$

By storing M previous curvature triples (8) and additionally the sets $C_{t+1-M}, \ldots, C_t$, we can calculate the action of L_t on a matrix $V \in \mathbb{R}^{d\times q}$ by using (11), see Algorithm 3. To the best of our knowledge, this is the first limited-memory factored form in the literature. Since we do not store L_t explicitly for any t we do not have access to L_{t-M}, required in computing (11). Thus we simply use $L_{t-M} = I$. Again, we can implement a more numerically stable version of Algorithm 3 by storing the Cholesky factor of $D_i^T Y_i$ and using triangular solves, as opposed to calculating the inverse matrix $\Delta_i = (D_i^T Y_i)^{-1}$.

Algorithm 3 Block L-BFGS Update (Factored loop recursion)

inputs: $V, D_i, Y_i, \Delta_i \in \mathbb{R}^{d\times q}$ and $C_i \subset [d]$, for $i \in \{t+1-M, \ldots, t\}$.
initiate: $W \leftarrow V$
for $i = t+1-M, \ldots, t$ **do**
  $W = W - D_i\Delta_i Y_i^T W + D_i R_i W_{C_i:}$
end for
output $L_t V \leftarrow W$

4. Convergence

In this section we prove that Algorithm 1 converges linearly. Our analysis relies on the following assumption, and is a combination of novel insights and techniques from (Konečný & Richtárik, 2014) and (Moritz et al., 2016).

Assumption 1. *There exist constants $0 < \lambda \leq \Lambda$ such that*

$$\lambda I \preceq \nabla^2 f_T(x) \preceq \Lambda I \tag{12}$$

for all $x \in \mathbb{R}^d$ and all $T \subseteq [n]$.

We first need two technical lemmas.

Lemma 1. *Let H_t be the result of applying the limited-memory Block BFGS update with memory M, as implicitly defined by Algorithm 2. Then there exists positive constants $0 < \gamma \leq \Gamma$ such that for all t we have*

$$\gamma I \preceq H_t \preceq \Gamma I, \tag{13}$$

where

$$\frac{1}{1+M\Lambda} \leq \gamma \leq \Gamma \leq (1+\sqrt{\kappa})^{2M}\left(1+\frac{1}{\lambda(2\sqrt{\kappa}+\kappa)}\right),$$

and $\kappa \overset{def}{=} \Lambda/\lambda$.

A proof of this lemma is given in Section 7.

The spectral bound provided by Lemma 1 is many orders of magnitude tighter than the bound provided in (Moritz et al., 2016). As a result, our convergence rate is many orders of magnitude better than the one provided in (Moritz et al., 2016).

We now state a bound on the norm of the SVRG variance-reduced gradient for minibatches.

Lemma 2. *Suppose Assumption 1 holds, let w_* be the unique minimizer of f and let $w, x \in \mathbb{R}^d$. Let $\mu = \nabla f(w)$ and $g = \nabla f_S(x) - \nabla f_S(w) + \mu$. Taking expectation with respect to S, we have*

$$\begin{aligned} \mathbb{E}\left[\|g\|_2^2\right] &\leq 4\Lambda(f(x) - f(w_*)) \\ &\quad + 4(\Lambda - \lambda)(f(w) - f(w_*)). \end{aligned}$$

The following theorem guarantees the linear convergence of Algorithm 1.

Theorem 1. *Suppose that Assumption 1 holds. Let w_* be the unique minimizer of f. When Option II is used in Algorithm 1, we have for all $k \geq 0$ that*

$$\mathbb{E}\left[f(w_k) - f(w_*)\right] \leq \rho^k \mathbb{E}\left[f(w_0) - f(w_*)\right],$$

where the convergence rate is given by

$$\rho = \frac{1/2m\eta + \eta\Gamma^2\Lambda(\Lambda-\lambda)}{\gamma\lambda - \eta\Gamma^2\Lambda^2} < 1,$$

assuming we have chosen $\eta < \gamma\lambda/(2\Gamma^2\Lambda^2)$ and that we choose m large enough to satisfy

$$m \geq \frac{1}{2\eta\left(\gamma\lambda - \eta\Gamma^2\Lambda(2\Lambda-\lambda)\right)},$$

which is a positive lower bound given our restriction on η.

Proof. Since $g_t = \nabla f_{S_t}(x_t) - \nabla f_{S_t}(w_k) + \mu$ and $x_{t+1} =$

$x_t - \eta H_t g_t$ in Algorithm 1, from (12) we have that

$$f(x_{t+1}) \leq f(x_t) + \eta \nabla f(x_t)^T d_t + \frac{\eta^2 \Lambda}{2} \|d_t\|_2^2$$
$$= f(x_t) - \eta \nabla f(x_t)^T H_t g_t + \frac{\eta^2 \Lambda}{2} \|H_t g_t\|_2^2 .$$

Taking expectation conditioned on x_t (i.e., with respect to S_t, T_t and D_t) and using Lemma 1 we have

$$\mathbb{E}\left[f(x_{t+1}) \,|\, x_t\right] \leq f(x_t) - \eta \mathbb{E}\left[\nabla f(x_t)^T H_t \nabla f(x_t) \,|\, x_t\right] + \frac{\eta^2 \Lambda}{2} \mathbb{E}\left[\|H_t g_t\|_2^2 \,|\, x_t\right]$$
$$\leq f(x_t) - \eta\gamma \|\nabla f(x_t)\|_2^2 + \frac{\eta^2 \Gamma^2 \Lambda}{2} \mathbb{E}\left[\|g_t\|_2^2 \,|\, x_t\right]$$

Introducing the notation $\delta_f(x) \overset{\text{def}}{=} f(x) - f(w_*)$ and applying Lemma 2 and the fact that strongly convex functions satisfy the inequality $\|\nabla f(x)\|_2^2 \geq 2\lambda \delta_f(x)$ for all $x \in \mathbb{R}^d$, gives

$$\mathbb{E}\left[f(x_{t+1}) \,|\, x_t\right] \leq f(x_t) - 2\eta\gamma\lambda\delta_f(x_t) + 2\eta^2\Gamma^2\Lambda\left(\Lambda\delta_f(x_t)) + (\Lambda - \lambda)\delta_f(w_k)\right))$$
$$= f(x_t) - \alpha\delta_f(x_t) + \beta\delta_f(w_k).$$

where $\alpha = 2\eta\left(\gamma\lambda - \eta\Gamma^2\Lambda^2\right)$ and $\beta = 2\eta^2\Gamma^2\Lambda(\Lambda - \lambda)$. Taking expectation, summing over $t = 0, \ldots, m-1$ and using telescopic cancellation gives

$$\mathbb{E}\left[f(x_m)\right] = \mathbb{E}\left[f(x_0)\right] - \alpha\left(\sum_{t=1}^{m-1} \mathbb{E}\left[\delta_f(x_t)\right]\right) + m\beta\mathbb{E}\left[\delta_f(w_k)\right]$$
$$= \mathbb{E}\left[f(w_k)\right] - m\alpha\mathbb{E}\left[\delta_f(w_{k+1})\right] + m\beta\mathbb{E}\left[\delta_f(w_k)\right],$$

where we used that $w_k = x_0$ and $\sum_{t=1}^{m} \mathbb{E}\left[x_t\right] = m\mathbb{E}\left[w_{k+1}\right]$ which is a consequence of using Option II in Algorithm 1. Rearranging the above gives

$$0 \leq \mathbb{E}\left[f(w_k) - f(x_m)\right] - m\alpha\mathbb{E}\left[\delta_f(w_{k+1})\right] + m\beta\mathbb{E}\left[\delta_f(w_k)\right]$$
$$\leq \mathbb{E}\left[\delta_f(w_k)\right] - m\alpha\mathbb{E}\left[\delta_f(w_{k+1})\right] + m\beta\mathbb{E}\left[\delta_f(w_k)\right]$$
$$= -m\alpha\mathbb{E}\left[\delta_f(w_{k+1})\right] + (1 + m\beta)\mathbb{E}\left[\delta_f(w_k)\right]$$

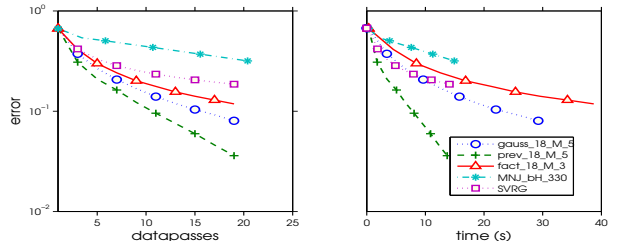
where we used that $f(w_*) \leq f(x_m)$. Using that $\eta < \gamma\lambda/(2\Gamma^2\Lambda^2)$, it follows that

$$\mathbb{E}\left[\delta_f(w_{k+1})\right] \leq \frac{1 + 2m\eta^2\Gamma^2\Lambda(\Lambda - \lambda)}{2m\eta(\gamma\lambda - \eta\Gamma^2\Lambda^2)} \mathbb{E}\left[\delta_f(w_k)\right]. \qquad \square$$

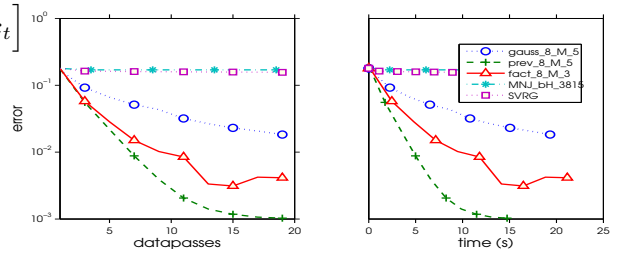
5. Numerical Experiments

To validate our approach, we compare our Algorithm to the SVRG and the variable-metric adaption of the SVRG presented in (Moritz et al., 2016), which we refer to as the MNJ method. We tested seven empirical risk minimization problems with a logistic loss and L2 regularizer using data from LIBSVM (Chang & Lin, 2011). We set the regularization parameter $\lambda = 1/n$ for all experiments. All the methods were implemented in MATLAB.[3]

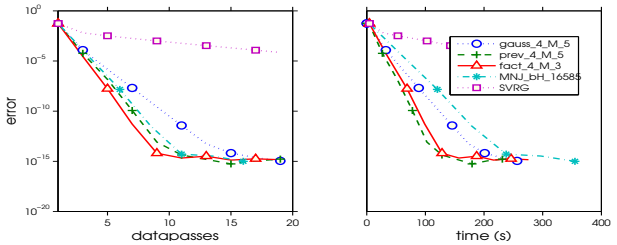
[3]All the code for the experiments can be downloaded from www.maths.ed.ac.uk/~prichtar/i_software.html.



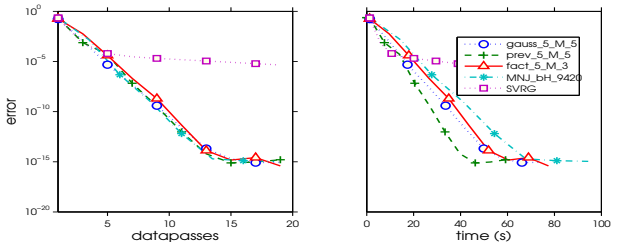
(a) gissette_scale



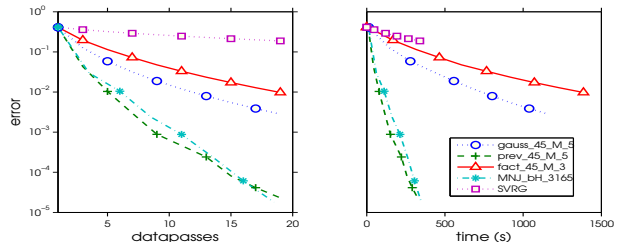
(b) covtype-libsvm-binary



(c) HIGGS



(d) SUSY

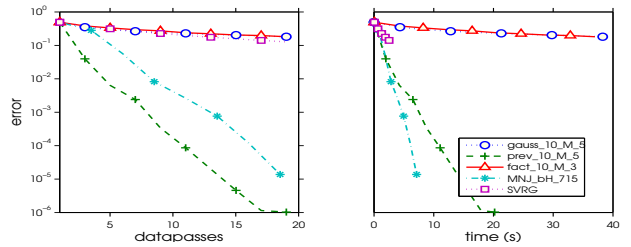


(e) epsilon_normalized

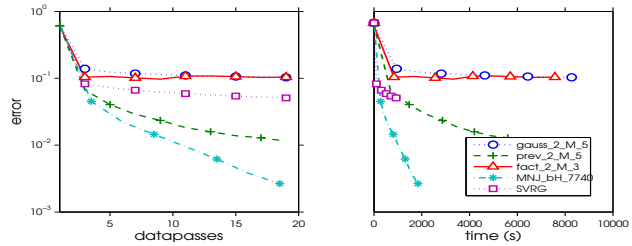
Figure 1. (a) gissette_scale $(d; n) = (5,000, 6,000)$ (b) covtype-libsvm-binary $(d; n) = (54, 581, 012)$ (c) HIGGS $(d; n) = (28; 11,000,000)$ (d) SUSY $(d; n) = (18; 3,548,466)$ (e) epsilon_normalized $(d; n) = (2,000; 400,000)$

method	description
`guass_q_M`	$D_t \in \mathbb{R}^{d\times q}$ with i.i.d Gaussian entries
`prev_L_M`	$D_t = [d_t, \ldots, d_{t-L+1}]$. Updated every L inner iterations
`fact_q_M`	$D_t = I_{t-1} I_{:C}$ where $C \subset [n]$ sampled uniformly at random and $\lvert C\rvert = q$
`MNJ_|T|`	Algorithm 1 in (Moritz et al., 2016) where $\lvert T\rvert$ = size of Hessian subsampling

Table 1. A key to the abbreviation of each method



(a) `rcv1-train.binary`



(b) `url-combined`

Figure 2. (a) `rcv1-train.binary` $(d; n) = (47,236; 20,242)$ (b) `url-combined` $(d; n) = (3,231,961; 2,396,130)$.

We tested three variants of Algorithm 1 with three different sketching matrices, each specified by the use of a different sketching matrix. In Table 1 we present a key to the abbreviations used in all our figures. The first three methods, `gauss`, `prev` and `fact`, are implementations of the three variants (a), (b) and (c), respectively, of Algorithm 1 using the three different sketching methods discussed at the start of Section 3. In these three methods the M stands for the number of stored curvature triples (8) used. In all our variants, for simplicity, we use the same subsampling for the gradient and Hessian, that is $S_t = T_t$.

We set the subsampling size $|S_t| = \sqrt{n}$ throughout our tests. We tested each method with a stepsize

$$\eta \in \{10^0,\ 5\cdot 10^{-1},\ 10^{-1},\ \ldots,\ 10^{-6},\ 5\cdot 10^{-7},\ 10^{-7}\}$$

for the best outcome, and used the resulting η. Finally, we used $m = \lfloor n/|S_t| \rfloor$ for the number of inner iterations, so that the SVRG method performs an entire pass over the data before recalculating the gradient.

For the `MNJ` method, we used the suggestion of the authors of both (Byrd et al., 2015) and (Moritz et al., 2016), and choose $|T_t| \approx L|S_t|$ so that the computational workload of performing L inner iterations of the SVRG method was approximately equal to that of applying the L-BFGS metric once. The exact rule we used was

$$|T_t| = \left\lfloor \min\left\{ \tfrac{L|S_t|}{2}, n^{2/3} \right\} \right\rfloor .$$

We set the memory to 10 for the `MNJ` method in all tests, which is a standard choice for the L-BFGS method.

In Figures 1 and 2 we plot the error $f(x_t) - f(w_*)$ against datapasses and time for each method[4]. While measuring time is implementation dependent, we decided to include these time plots in order to offer further insight into the methods performance. Note that we did not use any sophisticated implementation tricks such as "lazy" gradient updates (Konečný & Richtárik, 2014), but instead implemented each method as originally designed so that the methods could be compared on a equal footing.

On the problems with d significantly smaller then n, such as those in Figures 1(c) and 1(d), all the methods that make use of curvature information performed similarly and significantly better than the SVRG method. The `prev` method proved to be the best overall and the most robust, performing comparably well on problems with $d \ll n$ including Figures 1(c) and 1(d), but was also the most efficient method in Figures 1(a), 1(b), 1(e) (shared being most efficient with `MNJ`) and Figure 2(a). The only problem on which the `prev` method was not the most efficient method was on the `url-combined` problem in Figure 2(b), where the `MNJ` method proved to be the most efficient.

In general, Algorithm 1 and its variants perform very well in terms of time taken, but better still in terms of data passes. This is because our methods are able to better utilize the available stochastic information in each iteration than competing methods. We do this by performing more expensive computations aimed at collecting more curvature information. This approach pays its dividends in regimes where data access is costly, relative to the time spent on processing the data.

6. Proof of Lemma 2

Let $h_S(w) = f_S(w) - f_S(w_*) - \nabla f_S(w_*)^T(w - w_*)$. Note that $h_S(w)$ achieves its minimum at w_* and $h_S(w_*) = 0$. Furthermore, $\nabla^2 h_S(w) \preceq \Lambda I$ from Assumption 1. Consequently, for every $b \in \mathbb{R}^d$ we have

$$0 = h_S(w_*) \le h_S(w+b) \le h_S(w) + \nabla h_S(w)^T b + \tfrac{\Lambda}{2}\|b\|_2^2 .$$

[4] Thanks to Mark Schmidt, whose code `prettyPlot` was used to generate all figures: `https://www.cs.ubc.ca/~schmidtm/`

Minimizing the right hand side of the above in b gives

$$0 = h_S(w_*) \leq h_S(w) - \tfrac{1}{2\Lambda}\left\|\nabla h_S(w)\right\|_2^2.$$

Re-arranging the above and switching back to f, we have

$$\begin{aligned}&\left\|\nabla f_S(w) - \nabla f_S(w_*)\right\|_2^2\\ &\leq 2\Lambda\left(f_S(w) - f_S(w_*) - \nabla f_S(w_*)^T(w - w_*)\right).\end{aligned}$$

Recalling the notation $\delta_f(x) \overset{\text{def}}{=} f(x) - f(w_*)$ and taking expectation with respect to S gives

$$\mathbb{E}\left[\left\|\nabla f_S(w) - \nabla f_S(w_*)\right\|_2^2\right] \leq 2\Lambda\delta_f(w). \tag{14}$$

Now we apply the above to obtain

$$\begin{aligned}\mathbb{E}\left[\left\|g\right\|_2^2\right] &\leq 2\mathbb{E}\left[\left\|\nabla f_S(x) - \nabla f_S(w_*)\right\|_2^2\right]\\ &+ 2\mathbb{E}\left[\left\|\nabla f_S(w) - \nabla f_S(w_*) - \mu\right\|_2^2\right]\\ &\leq 2\mathbb{E}\left[\left\|\nabla f_S(x) - \nabla f_S(w_*)\right\|_2^2\right]\\ &+ 2\mathbb{E}\left[\left\|\nabla f_S(w) - \nabla f_S(w_*)\right\|_2^2\right]\\ &- 2\left\|\nabla f(w)\right\|_2^2.\end{aligned}$$

where we used $\left\|a+b\right\|_2^2 \leq 2\left\|a\right\|_2^2 + 2\left\|b\right\|_2^2$ in the first inequality and $\mu = \mathbb{E}\left[\nabla f_S(w_k) - \nabla f_S(w_*)\right]$. Finally,

$$\begin{aligned}\mathbb{E}\left[\left\|g\right\|_2^2\right] &\overset{(14)}{\leq} 4\Lambda\left(\delta_f(x) + \delta_f(w)\right) - 2\left\|\nabla f(w)\right\|_2^2\\ &\leq 4\Lambda\delta_f(x) + 4(\Lambda - \lambda)(\delta_f(w)).\end{aligned}$$

In the last inequality we used the fact that strongly convex functions satisfy $\left\|\nabla f(x)\right\|_2^2 \geq 2\lambda\,\delta_f(x)$ for all $x \in \mathbb{R}^d$.

7. Proof of Lemma 1

To simplify notation, we define $G \overset{\text{def}}{=} \nabla^2 f_T(x_t)$, $\Delta = \Delta_t$, $Y = Y_t$, $H = H_{t-1}$, $H^+ = H_t$, $B = H^{-1}$, $B^+ = (H^+)^{-1}$ and $V = Y\Delta D^T$. Thus, the block BFGS update (6) can be written as

$$H^+ = H - V^T H - HV + V^T HV + D\Delta D^T.$$

Proposition 2.2 in (Gower & Gondzio, 2014) proves that so long as G and H (and hence B) are positive definite and D has full rank, then H^+ is positive definite and non-singular, and consequently, B^+ is well defined and positive definite. Using the Sherman-Morrison-Woodbury identity the update formula for B^+, as shown in the Appendix in (Gower & Gondzio, 2014), is given by

$$B^+ = B + Y\Delta Y^{^T} - BD(D^{^T}BD)^{-1}D^{^T}B. \tag{15}$$

We will now bound $\lambda_{\max}(H^+) = \left\|H^+\right\|_2$ from above and $\lambda_{\min}(H^+) = 1/\left\|B^+\right\|_2$ from below.

Let $C = BD(D^{^T}BD)^{-1}D^{^T}B$. Then since $C \succeq 0$, $B - C \preceq B$ and hence, $\left\|B - C\right\|_2 \leq \left\|B\right\|$ and

$$\left\|B^+\right\|_2 \leq \left\|B\right\|_2 + \left\|Y\Delta Y^{^T}\right\|_2.$$

Now, letting $G^{\frac{1}{2}}$ and $G^{-\frac{1}{2}}$ denote the unique square root of G and its inverse, and defining $U = G^{\frac{1}{2}}D$, we have

$$D\Delta D^{^T} = G^{-\frac{1}{2}}U(U^{^T}U)^{-1}U^{^T}G^{-\frac{1}{2}} = G^{-\frac{1}{2}}PG^{-\frac{1}{2}},$$

where $P = U(U^{^T}U)^{-1}U^{^T}$ is an orthogonal projection matrix. Moreover, it is easy to see that

$$Y\Delta Y^{^T} = G^{\frac{1}{2}}PG^{\frac{1}{2}} \text{ and } V = Y\Delta D^{^T} = G^{\frac{1}{2}}PG^{-\frac{1}{2}}.$$

Since $\left\|MN\right\|_2 \leq \left\|M\right\|_2\left\|N\right\|_2$ and $\left\|P\right\|_2 = 1$, we have $\left\|D\Delta D^{^T}\right\|_2 \leq \left\|G^{-1}\right\|_2$, $\left\|Y\Delta Y^{^T}\right\|_2 \leq \left\|G\right\|_2$ and $\left\|Y\Delta D^{^T}\right\|_2 \leq \left\|G^{-\frac{1}{2}}\right\|_2\left\|G^{\frac{1}{2}}\right\|_2$.

Hence, $\left\|B^+\right\|_2 \leq \left\|B\right\|_2 + \left\|G\right\|_2 \overset{(12)}{\leq} \left\|B\right\|_2 + \Lambda$. Furthermore,

$$\begin{aligned}\left\|H^+\right\|_2 &\leq \left\|H\right\|_2 + 2\left\|H\right\|_2\left\|G^{-\frac{1}{2}}\right\|_2\left\|G^{\frac{1}{2}}\right\|_2\\ &+ \left\|H\right\|_2\left\|G^{-1}\right\|_2\left\|G\right\|_2 + \left\|G^{-1}\right\|_2,\\ &\leq (1 + 2\sqrt{\kappa} + \kappa)\left\|H\right\|_2 + \tfrac{1}{\lambda} = \alpha\left\|H\right\|_2 + \tfrac{1}{\lambda},\end{aligned}$$

where $\kappa = \Lambda/\lambda$ and $\alpha = (1 + \sqrt{\kappa})^2$.

Since we use a memory of M block triples (D_i, Y_i, Δ_i), and the metric matrix H_t is the result of applying, at most, M block updates BFGS (6) to H_0, we have that

$$\lambda_{\max}(B_t) = \left\|B_t\right\| \leq \left\|B_{t-M}\right\| + M\Lambda, \tag{16}$$

and hence that

$$\gamma = \lambda_{\min}(H_t) \geq \tfrac{1}{\left\|B_{t-M}\right\| + M\Lambda}.$$

Finally, letting $\alpha = (1 + \sqrt{\kappa})^2$, we have

$$\begin{aligned}\Gamma &= \lambda_{\max}(H_t) = \left\|H_t\right\| \leq \alpha^M\left\|H_{t-M}\right\| + \tfrac{1}{\lambda}\textstyle\sum_{i=0}^{M-1}\alpha^i\\ &= \alpha^M\left\|H_{t-M}\right\| + \tfrac{1}{\lambda}\tfrac{\alpha^M - 1}{\alpha - 1}\\ &\leq (1 + \sqrt{\kappa})^{2M}(\left\|H_{t-M}\right\| + \tfrac{1}{\lambda(2\sqrt{\kappa} + \kappa)}).\end{aligned}$$

The bound now follows by observing that $H_{t-M} = I$.

8. Extensions

This work opens up many new research avenues. For instance, sketching techniques are increasingly successful tools in large scale machine learning, numerical linear algebra, optimization and computer science, and thus one could employ a number of new sketching methods in the block BFGS method, such as the Walsh-Hadamard matrix (Pilanci & Wainwright, 2016; Lu et al., 2013). Using new sophisticated sketching methods, combined with the block BFGS update, could result in even more efficient and accurate estimates of the underlying curvature.

While in this work we have for simplicity focused on utilizing our metric learning techniques in conjunction with SVRG, they can be used with other optimization algorithms as well, including SGD, SAG, SAGA, SDCA and more.

Acknowledgements

Peter Richtárik would like to acknowledge support from EPSRC Grant EP/K02325X/1 and EPSRC Fellowship EP/N005538/1. Donald Goldfarb would like to acknowledge support from NSF Grant CCF-1527809.

References

Broyden, C. G. Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381, 1967.

Byrd, R H, Hansen, S L, Nocedal, Jorge, and Singer, Y. A stochastic quasi-Newton method for large-scale optimization. *arXiv:1401.7020v2*, 2015.

Chang, Chih Chung and Lin, Chih Jen. LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, April 2011.

Christianson, Bruce. Automatic Hessians by reverse accumulation. *IMA Journal of Numerical Analysis*, 12(2): 135–150, 1992.

Defazio, Aaron, Bach, Francis, and Lacoste-Julien, Simon. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *arXiv:1407.0202*, 2014.

Erdogdu, Murat A. and Montanari, Andrea. Convergence rates of sub-sampled Newton methods. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 3052–3060. Curran Associates, Inc., 2015.

Fletcher, Rodger. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–323, 1970.

Goldfarb, Donald. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

Gower, Robert M. and Gondzio, Jacek. Action constrained quasi-Newton methods. *arXiv:1412.8045v1*, 2014.

Gower, Robert M. and Richtárik, Peter. Stochastic dual ascent for solving linear systems. *arXiv:1512.06890*, 2015.

Gratton, Serge, Sartenaer, Annick, and Ilunga, Jean Tshimanga. On a class of limited memory preconditioners for large-scale nonlinear least-squares problems. *SIAM Journal on Optimization*, 21(3):912–935, 2011.

Griewank, Andreas and Walther, Andrea. *Evaluating Derivatives*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edi edition, 2008.

Hennig, Philipp. Probabilistic interpretation of linear solvers. *SIAM Journal on Optimization*, 25(1):234–260, 2015.

Johnson, Rie and Zhang, Tong. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pp. 315–323. Curran Associates, Inc., 2013.

Konečný, Jakub and Richtárik, Peter. S2GD: Semi-stochastic gradient descent methods. *arXiv:1312.1666*, 2014.

Konečný, Jakub, Liu, Jie, Richtárik, Peter, and Takáč, Martin. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255, 2016.

Lu, Yichao, Dhillon, Paramveer, Foster, Dean P, and Ungar, Lyle. Faster ridge regression via the subsampled randomized Hadamard transform. In *Advances in Neural Information Processing Systems 26*, pp. 369–377. 2013.

Mandel, Jan and Brezina, Marian. Balancing domain decomposition: Theory and performance in two and three dimensions. *Communications in Numerical Methods in Engineering*, 9(3):233–241, 1993.

Mokhtari, Aryan and Ribeiro, Alejandro. Regularized stochastic BFGS algorithm. *IEEE Transactions on Signal Processing*, 62:1109–1112, 2014.

Mokhtari, Aryan and Ribeiro, Alejandro. Global convergence of online limited memory BFGS. *The Journal of Machine Learning Research*, 16:3151–3181, 2015.

Moritz, Philipp, Nishihara, Robert, and Jordan, Michael I. A linearly-convergent stochastic L-BFGS algorithm. In *International Conference on Artificial Intelligence and Statistics*, pp. 249–258, 2016.

Nocedal, Jorge. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773, 1980.

Pearlmutter, Barak A. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

Pilanci, Mert and Wainwright, Martin J. Iterative Hessian sketch : Fast and accurate solution approximation for constrained least-squares. *Journal of Machine Learning Research*, 17:1–33, 2016.

Qu, Zheng, Richtárik, Peter, Takáč, Martin, and Fercoq, Olivier. SDNA: Stochastic dual Newton ascent for empirical risk minimization. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Roosta-Khorasani, Farbod and Mahoney, Michael W. Sub-sampled Newton methods I: globally convergent algorithms. *arXiv:1601.04737*, 2016.

Schmidt, Mark, Le Roux, Nicolas, and Bach, Francis. Minimizing finite sums with the stochastic average gradient. *arXiv:1309.2388*, 2013.

Schnabel, Robert B. Quasi-Newton methods using multiple secant equations ; cu-cs-247-83. Technical report, Computer Science Technical Reports. Paper 244, 1983.

Schraudolph, Nicol N, Simon, Gunter, and Yu, Jin. A stochastic quasi-Newton method for online convex optimization. *In Proceedings of 11th International Conference on Artificial Intelligence and Statistics*, 2007.

Shalev-Shwartz, Shai and Zhang, Tong. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14(1):567–599, February 2013.

Shalev-Shwartz, Shai and Zhang, Tong. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155(1): 105–145, 2016.

Shanno, D F. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24 (111):647–656, 1971.