SONIA: A Symmetric Blockwise Truncated Optimization Algorithm

Majid Jahani
Lehigh University
Bethlehem, PA 18015
majidjahani89@gmail.com

Mohammadreza Nazari
SAS Institute
Cary, NC 27513
mrza.nazari@gmail.com

Rachael Tappenden
University of Canterbury
Christchurch 8041, New Zealand
rachael.tappenden@canterbury.ac.nz

Albert S. Berahas
University of Michigan
Ann Arbor, MI 48109
albertberahas@gmail.com

Martin Takáč
Lehigh University
Bethlehem, PA 18015
Takac.MT@gmail.com

Abstract

This work presents a new optimization algorithm for empirical risk minimization. The algorithm bridges the gap between first- and second-order methods by computing a search direction that uses a second-order-type update in one subspace, coupled with a scaled steepest descent step in the orthogonal complement. To this end, partial curvature information is incorporated to help with ill-conditioning, while simultaneously allowing the algorithm to scale to the large problem dimensions often encountered in machine learning applications. Theoretical results are presented to confirm that the algorithm converges to a stationary point in both the strongly convex and nonconvex cases. A stochastic variant of the algorithm is also presented, along with corresponding theoretical guarantees. Numerical results confirm the strengths of the new approach on standard machine learning problems.

1 Introduction

This paper presents a novel optimization algorithm for empirical risk minimization:

$$\min_{w\in\mathbb{R}^d} F(w) := \frac{1}{n}\sum_{i=1}^{n} f(w; x^i, y^i) = \frac{1}{n}\sum_{i=1}^{n} f_i(w), \quad (1.1)$$

where $\{(x^i, y^i)\}_{i=1}^n$ are training examples (observations), and $f_i : \mathbb{R}^d \to \mathbb{R}$ is the composition of a prediction function (parameterized by $w \in \mathbb{R}^d$) and a loss function associated with the ith training observation (sample). Problems of the form (1.1) arise in a wide variety of machine learning applications [Bishop, 2006, Chang and Lin, 2011, Friedman et al., 2001, LeCun et al., 2015]. The main challenge of solving these problems stems from the fact that they are often high-dimensional and nonlinear, and may be nonconvex.

For many machine learning applications, a common approach is to employ first-order methods such as the Stochastic Gradient method (SGD). SGD and its variance-reduced, adaptive and distributed variants [Robbins and Monro, 1951, Duchi et al., 2011, Schmidt et al., 2017, Johnson and Zhang, 2013, Nguyen et al., 2017, Kingma and Ba, 2014, Recht et al., 2011] are popular because they are simple to implement and have low per-iteration cost. However, these methods often require significant tuning efforts to ensure practical performance, and they struggle on ill-conditioned problems.

One avenue for mitigating the aforementioned issues is the use of second-order or quasi-Newton methods [Nocedal and Wright, 2006, Fletcher, 1987, Dennis and Moré, 1977]. These methods, in the deterministic setting, are relatively insensitive to their associated hyper-parameters and are able to alleviate the effects of ill-conditioning. Unfortunately, a drawback of these methods is that they often do not scale sufficiently well with the high dimensionality (both n and d) typical in machine/deep learning problems. Thus, the computational burden of using deterministic higher-order methods is often deemed to be too high.

Recently, attention has shifted towards stochastic second-order [Roosta-Khorasani and Mahoney, 2018, Byrd et al., 2011, Martens, 2010, Gower et al., 2019, Bollapragada et al., 2016] and quasi-Newton

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

[Curtis, 2016, Byrd et al., 2016, Berahas et al., 2016, Mokhtari and Ribeiro, 2015, Schraudolph et al., 2007, Berahas and Takáč, 2020, Keskar and Berahas, 2016] methods. These methods attempt to combine the speed of Newton's method and the scalability of first-order methods by incorporating curvature information in a judicious manner, and have proven to work well for several machine learning tasks [Berahas et al., 2020, Xu et al., 2020]. However, the question of how to balance the accuracy in the gradient and Hessian approximation is yet unresolved, and as such these methods often perform on par with their first-order variants.

Several other attempts have been made to balance the first- versus second-order trade-off, in order to find ways of incorporating partial curvature information to help with ill-conditioning at an acceptable cost. For example, variants of coordinate descent methods that perform second-order-type updates restricted to a low dimensional subspace, are prototypical methods in this niche [Fountoulakis and Tappenden, 2018, Richtárik and Takáč, 2014, Tappenden et al., 2016]. However, while progress has been made, the gap between first- and second-order methods remains.

In this paper, we propose the **S**ymmetric bl**O**ckwise tru**N**cated optim**I**zation **A**lgorithm (`SONIA`). `SONIA` aims to bridge the gap between first- and second-order methods, but is different in nature to coordinate descent methods because at every iteration a step in the *full dimensional space* is generated. The search direction consists of two components. The first component lies in an m-dimensional subspace (where $m \ll d$ is referred to as the 'memory' and is user defined), and is generated using a second-order approach. The second component of the update lies in the orthogonal complement, and is an inexpensive scaled steepest descent update. The combination of the two components allows for the overall search direction to explore the full-dimensional space at every iteration.

Contributions

- ***Novel Optimization Algorithm***. We propose `SONIA` for solving empirical risk minimization problems; the method attempts to bridge the gap between first- and second-order methods. `SONIA` judiciously incorporates curvature information in one subspace (dimension is determined by the user) and takes a gradient descent step in the complement of that subspace. As such, at every iteration, `SONIA` takes a step in full dimensional space while retaining a low per-iteration cost and storage, similar to that of limited memory quasi-Newton methods.
- ***Theoretical Analysis***. We derive convergence guarantees for `SONIA` (deterministic and stochastic regimes) for strongly convex and nonconvex optimization problems. These guarantees match those of popular quasi-Newton methods such as L-BFGS.
- ***Stochastic Variant***. We develop and analyze a stochastic variant of `SONIA` that uses stochastic gradient and Hessian approximations in lieu of the true gradient and Hessian, respectively.
- ***Competitive Numerical Results***. We investigate the performance of `SONIA` (deterministic and stochastic) on strongly convex (logistic regression) and nonconvex (nonlinear least squares) problems that arise in machine learning. Our proposed methods are competitive with the algorithms of choice in both the deterministic and stochastic settings.

Organization Related works are described in Section 2. Section 3 presents our proposed algorithm, `SONIA`, and its stochastic variant. We show the theoretical properties of our proposed method in Section 4. Numerical results on deterministic and stochastic problems are reported in Section 5. Finally, in Section 6 we provide some final remarks and discuss possible avenues for future research.

2 Related Work

The following works employ iterate updates of the form:

$$w_{k+1} = w_k + \alpha_k p_k, \tag{2.1}$$

where $p_k \in \mathbb{R}^d$ is the search direction and $\alpha_k > 0$ is the step length (or learning rate).

The work in [Paternain et al., 2019] proposes a Newton-type algorithm for nonconvex optimization. At each iteration the construction and eigenvalue decomposition (full dimensional) of the Hessian is required, small (in modulus) eigenvalues are truncated, and a Newton-like search direction is generated using the truncated inverse Hessian instead of the true inverse Hessian. The method works well in practice and is guaranteed to converge to local minima, but is expensive.

Quasi-Newton methods–methods that compute search directions using (inverse) Hessian approximations constructed using past iterate and gradient information–represent some of the most effective algorithms for minimizing nonlinear objective functions. This class of nonlinear optimization algorithms includes BFGS, DFP and SR1; see [Fletcher, 1987, Nocedal and Wright, 2006, Dennis and Moré, 1977] and the references therein.

The Symmetric Rank One (SR1) update is a special case of a rank one quasi-Newton method [Dennis and Moré, 1977]. It is the unique symmetric rank-1 update that satisfies the secant condition

$B_{k+1}s_k = y_k$, where $s_k = w_k - w_{k-1}$ and $y_k = \nabla F(w_k) - \nabla F(w_{k-1})$ are the curvature pairs and the Hessian approximation is updated at every iteration via:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{s_k^T (y_k - B_k s_k)}.$$

Hessian approximations using SR1 updates are not guaranteed to be positive definite, and while this was originally seen as a drawback, it is arguably viewed as an advantage in the context of nonconvex optimization. Limited memory variants exist where there is a fixed memory size m, and only the last m curvature pairs are kept and used to construct the Hessian approximation [Brust et al., 2017, Lu, 1996]. Let $S_k = [s_{k-m+1}, \dots, s_k] \in \mathbb{R}^{d\times m}$ and $Y_k = [y_{k-m+1}, \dots, y_k] \in \mathbb{R}^{d\times m}$ denote matrices consisting of the m most recent curvature pairs. As studied in [Byrd et al., 1994], the compact form of L-SR1 is as follows:

$$B_k = B_0 + (Y_k - B_0 S_k) M_k^{-1} (Y_k - B_0 S_k)^T, \quad (2.2)$$

where $M_k = L_k + D_k + L_k^T - S_k^T B_0 S_k$ and $S_k^T Y_k = L_k + D_k + U_k$, L_k denotes the strictly lower triangular part, D_k is the diagonal and U_k denotes the strictly upper triangular part of $S_k^T Y_k$, respectively, and B_0 is an initial approximation (usually set as $B_0 = \eta I$, $\eta > 0$). A key observation is that while B_{k+1} is a full dimensional $d \times d$ matrix, the inverse in (2.2) is a small $m \times m$ matrix. Recent works that employ the compact L-SR1 update include [Erway et al., 2019, Brust et al., 2017, Berahas et al., 2019], where (2.2) defines the quadratic model within a trust region algorithm.

Rather than maintaining a history of the m most recent curvature pairs, the work [Berahas et al., 2019] proposed a *sampled* variant of the L-SR1 update. In that work, at each iteration $k \geq 0$, m directions $\{s_1, \dots, s_m\}$ are sampled around the current iterate w_k and stored as $S_k = [s_1, \dots, s_m] \in \mathbb{R}^{d\times m}$. Next, the gradient displacement vectors are computed via

$$Y_k = \nabla^2 F(w_k) S_k, \quad (2.3)$$

and the matrix B_0 is set to zero. In this way, previous curvature information is *forgotten*, and local curvature information is *sampled* around the current iterate. Moreover, depending on the way the vectors $\{s_1, \dots, s_m\}$ are sampled, one can view (2.3) as a sketch of the Hessian [Woodruff, 2014].

The approach proposed here combines quasi-Newton updates for indefinite Hessians [Erway et al., 2019] with sampled curvature pairs [Berahas et al., 2019]. Moreover, a truncation step (as in [Paternain et al., 2019]) allows us to avoid checking conditions on the curvature pairs, and ensures that the Hessian approximations constructed are positive definite. The subspace generation is based on the user-defined hyper-parameter m (user has full control over the computational cost of each iteration), and an eigenvalue decomposition step (performed in reduced dimension and so is cheap).

3 Symmetric blOckwise truNcated optimIzation Algorithm (SONIA)

In this section, we present our proposed algorithm. We begin by motivating and describing the deterministic variant of the method and then discuss its stochastic counterpart. We end this section by discussing the per iteration complexity of SONIA.

3.1 Deterministic SONIA

The SONIA algorithm updates iterates via (2.1). The search direction p_k consists of two components; the first component lies in one subspace and is a second-order based update, while the second component lies in the orthogonal complement and is a scaled steepest descent direction. We now describe how the subspaces are built at each iteration as well as how to compute the second-order component of the search direction.

The algorithm is initialized with a user defined parameter $m \ll d$ (*memory* size). At each iteration $k \geq 0$ of SONIA, m directions $\{s_1, \dots, s_m\}$ are randomly sampled, and curvature pair matrices S_k and Y_k are constructed via (2.3). Setting $B_0 = 0$, and substituting into (2.2) gives the compact form of the Hessian approximation used in this work[1]:

$$B_k = Y_k (Y_k^T S_k)^\dagger Y_k^T. \quad (3.1)$$

Similar to the strategy in [Erway et al., 2019], using the *thin* QR factorization of $Y_k = Q_k R_k$, where $Q_k \in \mathbb{R}^{d\times m}$ has orthonormal columns and $R_k \in \mathbb{R}^{m\times m}$ is an upper triangular matrix, (3.1) gives

$$B_k = Q_k R_k (Y_k^T S_k)^\dagger R_k^T Q_k^T. \quad (3.2)$$

Note that the matrix B_k is symmetric since, by (2.3) $(Y_k^T S_k)^\dagger = (S_k^T \nabla^2 F(w_k) S_k)^\dagger \in \mathbb{R}^{m\times m}$ is symmetric. Thus, by the spectral decomposition, $R_k (Y_k^T S_k)^\dagger R_k^T = V_k \Lambda_k V_k^T$, where the columns of $V_k \in \mathbb{R}^{m\times m}$ form an orthonormal basis (of eigenvectors), and $\Lambda_k \in \mathbb{R}^{m\times m}$ is a diagonal matrix containing the corresponding eigenvalues. Substituting this into (3.2) gives

$$B_k = Q_k V_k \Lambda_k V_k^T Q_k^T. \quad (3.3)$$

Since Q_k has orthonormal columns and V_k is an orthogonal matrix, it is clear that

$$\tilde{V}_k := Q_k V_k \in \mathbb{R}^{d\times m} \quad (3.4)$$

[1] If $S_k^T Y_k$ is full rank, then the pseudo-inverse in (3.1) is simply the inverse.

has orthonormal columns. Finally, the low rank decomposition of the Hessian approximation B_k is

$$B_k = \tilde{V}_k \Lambda_k \tilde{V}_k^T. \tag{3.5}$$

The following definition is motivated by [Paternain et al., 2019, Definition 2.1].

Definition 3.1. *Let B_k, $\tilde{V}_k$ and Λ_k be the matrices in (3.5), and let $0 < \omega \leq \Omega$. The truncated inverse Hessian approximation of B_k is $A_k := \tilde{V}_k(|\Lambda_k|_\omega^\Omega)^{-1}\tilde{V}_k^T$, where $(|\Lambda_k|_\omega^\Omega)_{ii} = \min\{\max\{|\Lambda_k|_{ii}, \omega\}, \Omega\}$.*

Definition 3.1 explains that any eigenvalue of Λ_k below (resp. above) the threshold ω (resp. Ω) is truncated and set to ω (resp. Ω). This is useful for several reasons. For one, it ensures that the search direction consists only of directions with non-negligible curvature. Moreover, unlike classical quasi-Newton methods that enforce conditions on the curvature pairs to guarantee that the (inverse) Hessian approximations are well-defined and that the updates are stable [Nocedal and Wright, 2006], `SONIA` utilizes a truncation step (Definition 3.1) and as such avoids the need for any such safe-guards. The reason for this is that even if $Y_k^T S_k$ is rank deficient, the truncation step ensures that A_k has full rank. This is especially important with SR1-type methods that require matrix-vector products in the checks [Nocedal and Wright, 2006].

Before we proceed, we make a few more observations about our algorithmic choice of constructing the gradient differencing curvature pairs via (2.3). As mentioned above, this ensures that the matrix $Y_k^T S_k$ is symmetric which is a fundamental component of our approach for three reasons. First, it ensures that the Hessian approximations constructed are symmetric. Second, it guarantees that the spectral decomposition exists (see (3.3)). Third, unlike the SR1 method that utilizes only the lower triangular part of the $Y_k^T S_k$ (non-symmetric) matrix to construct Hessian approximations (see (2.2)) and discards possibly useful curvature information, `SONIA` incorporates information from the full matrix. Furthermore, one can show that constructing curvature pairs in this manner guarantees that the secant equations hold for all curvature pairs, and that the Hessian approximations are scale invariant.

Next, we discuss the subspace decomposition. The gradient is orthogonally decomposed as:

$$\begin{aligned} \nabla F(w_k) &= g_k + g_k^\perp, \\ \text{where} \quad g_k &= \tilde{V}_k \tilde{V}_k^T \nabla F(w_k) \\ \text{and} \quad g_k^\perp &= (I - \tilde{V}_k \tilde{V}_k^T)\nabla F(w_k). \end{aligned} \tag{3.6}$$

Clearly, $g_k \in \text{range}(\tilde{V}_k\tilde{V}_k^T)$ and $g_k^\perp \in \ker(\tilde{V}_k\tilde{V}_k^T) \equiv \text{range}(I - \tilde{V}_k\tilde{V}_k^T)$. Vectors g_k and $g_k^\perp$ are orthogonal because the subspaces $\text{range}(\tilde{V}_k\tilde{V}_k^T)$ and $\text{range}(I - \tilde{V}_k\tilde{V}_k^T)$ are orthogonal complements (i.e., $g_k^T g_k^\perp = \nabla F(w_k)^T \tilde{V}_k\tilde{V}_k^T(I - \tilde{V}_k\tilde{V}_k^T)\nabla F(w_k) = 0$).

The `SONIA` search direction is

$$\begin{aligned} p_k &:= -\tilde{V}_k(|\Lambda_k|_\omega^\Omega)^{-1}\tilde{V}_k^T\nabla F(w_k) - \rho_k(I - \tilde{V}_k\tilde{V}_k^T)\nabla F(w_k) \\ &= -\tilde{V}_k(|\Lambda_k|_\omega^\Omega)^{-1}\tilde{V}_k^T g_k - \rho_k g_k^\perp, \end{aligned} \tag{3.7}$$

where for all $k \geq 0$,

$$\rho_k \in \left[\frac{1}{\Omega}, \hat{\lambda}_k\right], \tag{3.8}$$

$$\text{and} \quad \hat{\lambda}_k := \max_i\{[(|\Lambda_k|_\omega^\Omega)^{-1}]_{ii}\}. \tag{3.9}$$

Note, $\frac{1}{\Omega} \leq \hat{\lambda}_k \leq \frac{1}{\omega}$. The first component of the search direction lies in the subspace $\text{range}(\tilde{V}_k\tilde{V}_k^T)$, while the second component lies in the orthogonal complement.

Lemma 3.2. *The search direction p_k in (3.7) is equivalent to $p_k = -\mathcal{A}_k\nabla F(w_k)$, where*

$$\mathcal{A}_k := \tilde{V}_k(|\Lambda_k|_\omega^\Omega)^{-1}\tilde{V}_k^T + \rho_k(I - \tilde{V}_k\tilde{V}_k^T). \tag{3.10}$$

The search direction p_k in (3.7) can be interpreted as follows. If the memory is chosen as $m = 0$, then p_k is simply a scaled steepest descent direction (in this setting, ρ_k can be any positive number). On the other hand, if $m = d$, then p_k incorporates curvature information in the full dimensional space. If $0 < m < d$, then the algorithm is a hybrid of a second-order method in $\text{range}(\tilde{V}_k\tilde{V}_k^T)$ and steepest descent in the orthogonal complement $\text{null}(\tilde{V}_k\tilde{V}_k^T)$. Thus, this algorithm bridges the gap between first- and second-order methods.

Remark 3.3. *The following remarks are made regarding the search direction p_k.*

- *The first component of the search direction vanishes only if (i) the memory size is $m = 0$, or if (ii) $\nabla F(w) \in \text{null}(\tilde{V}\tilde{V}^T)$.*
- *The second component of the search direction vanishes only if (i) the memory size is $m = d$, or if (ii) $\text{range}(\tilde{V}\tilde{V}^T) \equiv \mathbb{R}^d$.*

The `SONIA` algorithm is presented in Algorithm 1. We should note, that the `SONIA` algorithm does not require the construction of the Hessian matrix or of the Hessian approximation matrix. Rather, `SONIA` is a *matrix-free* algorithm that utilizes Hessian-vector (or matrix) products to construct the curvature pairs and compute the search direction.

3.2 Stochastic `SONIA`

The `SONIA` algorithm presented in Section 3.1, requires a gradient evaluation and a Hessian-matrix product (to construct Y_k) at every iteration. For many machine learning applications, these computations can be

Algorithm 1 SONIA

Input: w_0 (initial iterate), m (memory), $0 < \omega \leq \Omega$ (truncation parameters).

for $k = 0, 1, 2, \ldots$ **do**
  Compute gradient $\nabla F(w_k)$
  Construct random $S_k \in \mathbb{R}^{d\times m}$ and set Y_k via (2.3)
  Compute QR factorization of $Y_k (= Q_k R_k)$
  Compute spectral decomposition of $R_k (Y_k^T S_k)^\dagger R_k^T (= V_k \Lambda_k V_k^T)$
  Construct $\tilde{V}_k (= Q_k V_k)$ via (3.4)
  Truncate eigenvalues of Λ_k to form $|\Lambda_k|_\omega^\Omega$
  Set ρ_k via (3.8)-(3.9)
  Decompose gradient $\nabla F(w_k) (= g_k + g_k^\perp)$ via (3.6)
  Compute search direction p_k via (3.7)
  Select steplength $\alpha_k > 0$, and set $w_{k+1} = w_k + \alpha_k p_k$
end for

prohibitively expensive. To overcome these difficulties, we present a stochastic variant of the SONIA algorithm that employs a *mini-batch* approach.

Stochastic SONIA chooses a set $\mathcal{I}_k \subset [n]$, and the new iterate is computed as follows:

$$w_{k+1} = w_k - \alpha_k \mathcal{A}_k \nabla F_{\mathcal{I}_k}(w_k),$$
$$\text{where} \quad \nabla F_{\mathcal{I}_k}(w_k) = \tfrac{1}{|\mathcal{I}_k|}\textstyle\sum_{i\in\mathcal{I}_k} \nabla F_i(w_k) \quad (3.11)$$

and $\mathcal{A}_k$ is the stochastic inverse truncated Hessian approximation. Stochastic SONIA uses stochastic Hessian-matrix products to construct Y_k, i.e., $Y_k = \nabla^2 F_{\mathcal{J}_k}(w_k) S_k$, where $\mathcal{J}_k \subset [n]$. It is important to note that for the theory (Section 4) the sample sets $\mathcal{I}_k$ and $\mathcal{J}_k$ need to be chosen independently.

3.3 Complexity of SONIA

The per iteration complexity of SONIA consists of: (1) a Hessian-matrix product ($\mathcal{O}(mnd)$); (2) a QR factorization of an $d \times m$ matrix ($\mathcal{O}(dm^2)$); (3) a pseudo-inverse of an $m \times m$ matrix ($\mathcal{O}(m^3)$); and, (4) a spectral decomposition of an $m \times m$ matrix ($\mathcal{O}(m^3)$). The computational cost and storage requirement for the SONIA algorithm are presented in Table 1[2], where we compare the cost and storage to popular limited-memory quasi-Newton methods and the NCN method [Paternain et al., 2019]. Note that the SONIA algorithm was developed for the regime where $m \ll d, n$. As is clear from Table 1, SONIA has similar cost and storage to LBFGS and LSR1, and is significantly more efficient, in both regards, to the NCN method. We should note that the computational cost and storage requirements for stochastic SONIA are $\mathcal{O}(d)$.

[2]The computations reported in Table 1 are on top of the function/gradient evaluations that are common to all methods. We highlight that the reported costs are in big $\mathcal{O}$ notation and in the regime $m \ll d, n$; in our experiments, we used the precise costs per iteration of each method in order to present meaningful results; see Section 5.

Table 1: Summary of Computational Cost and Storage (per iteration) for $m \ll n$, d.

method	computational cost	storage
NCN [Paternain et al., 2019]	$\mathcal{O}(nd^2 + d^3)$	$\mathcal{O}(d^2)$
LBFGS [Liu and Nocedal, 1989]	$\mathcal{O}(nd)$	$\mathcal{O}(d)$
LSR1 [Lu, 1996]	$\mathcal{O}(nd)$	$\mathcal{O}(d)$
SONIA [this paper]	$\mathcal{O}(nd)$	$\mathcal{O}(d)$

4 Theoretical Analysis

In this section, we present theoretical results for SONIA (deterministic and stochastic settings) for both strongly convex and nonconvex objective functions. Before we present the main theorems, we state two preliminary Lemmas that are used throughout this section. Proofs can be found in Appendix A.

Assumption 4.1. *The function F is twice continuously differentiable.*

Lemma 4.2. *The matrix $\mathcal{A}_k$ in (3.10) is positive definite for all $k \geq 0$.*

Lemma 4.3. *If Assumption 4.1 holds, there exist constants $0 < \mu_1 \leq \mu_2$ such that the inverse truncated Hessian approximations $\{\mathcal{A}_k\}$ generated by Algorithm 1 satisfy, $\mu_1 I \preceq \mathcal{A}_k \preceq \mu_2 I$, for all $k \geq 0$.*

4.1 Deterministic Setting

Strongly Convex Functions The following assumption is standard for strongly convex functions.

Assumption 4.4. *There exist positive constants $0 < \mu \leq L$ such that $\mu I \preceq \nabla^2 F(w) \preceq LI$, for all $w \in \mathbb{R}^d$.*

Theorem 4.5. *Suppose that Assumptions 4.1 and 4.4 hold, and let $F^\star = F(w^\star)$, where $w^\star$ is the minimizer of F. Let $\{w_k\}$ be the iterates generated by Algorithm 1, where $0 < \alpha_k = \alpha \leq \frac{\mu_1}{\mu_2^2 L}$, and w_0 is the starting point. Then, for all $k \geq 0$,*

$$F(w_k) - F^\star \leq (1 - \alpha\mu\mu_1)^k [F(w_0) - F^\star].$$

Theorem 4.5 shows that SONIA converges at a linear rate to the optimal solution of (1.1). The step length range prescribed by SONIA depends on μ_1 and μ_2, as does the rate. This is typical for limited memory quasi-Newton methods [Liu and Nocedal, 1989, Berahas et al., 2019]. In the worst-case, the matrix $\mathcal{A}_k$ can make the limit in Theorem 4.5 significantly worse than that of the first-order variant if the update has been unfortunate and generates ill-conditioned matrices. However, this is rarely observed in practice.

Nonconvex Functions The following assumptions are needed for the nonconvex case.

Assumption 4.6. *The function F is bounded below by a scalar $\widehat{F}$.*

Assumption 4.7. *The gradients of F are L-Lipschitz continuous for all $w \in \mathbb{R}^d$.*

Theorem 4.8. *Suppose that Assumptions 4.1, 4.6 and 4.7 hold. Let $\{w_k\}$ be the iterates generated by Algorithm 1, where $0 < \alpha_k = \alpha \leq \frac{\mu_1}{\mu_2^2 L}$, and w_0 is the starting point. Then, for any $T > 1$,*

$$\frac{1}{T}\sum_{k=0}^{T-1} \|\nabla F(w_k)\|^2 \leq \frac{2[F(w_0)-\hat{F}]}{\alpha\mu_1 T} \xrightarrow{T\to\infty} 0.$$

Theorem 4.8 bounds the average norm squared of the gradient and shows that the iterates spend increasingly more time in regions where the objective function has small gradient. By this result one can show that in the limit the iterates converge to a stationary point of F.

4.2 Stochastic Setting

Here, we present theoretical convergence results for the stochastic variant of SONIA. Note that, in this section $\mathbb{E}_{\mathcal{I}_k}[\cdot]$ denotes the conditional expectation given w_k, whereas $\mathbb{E}[\cdot]$ denotes the total expectation over the full history. We make the following standard assumptions.

Assumption 4.9. *There exist a constant γ such that $\mathbb{E}_{\mathcal{I}}[\|\nabla F_{\mathcal{I}}(w) - \nabla F(w)\|^2] \leq \gamma^2$.*

Assumption 4.10. *$\nabla F_{\mathcal{I}}(w)$ is an unbiased estimator of the gradient, i.e., $\mathbb{E}_{\mathcal{I}}[\nabla F_{\mathcal{I}}(w)] = \nabla F(w)$, where the samples $\mathcal{I}$ are drawn independently.*

Strongly Convex Functions

Theorem 4.11. *Suppose that Assumptions 4.1, 4.4, 4.9 and 4.10 hold, and let $F^\star = F(w^\star)$, where $w^\star$ is the minimizer of F. Let $\{w_k\}$ be the iterates generated by Algorithm 1, where $0 < \alpha_k = \alpha \leq \frac{\mu_1}{\mu_2^2 L}$, and w_0 is the starting point. Then, for all $k \geq 0$,*

$$\mathbb{E}[F(w_k) - F^\star] \leq (1 - \alpha\mu_1\mu)^k (F(w_0) - F^\star - \tfrac{\alpha\mu_2^2\gamma^2 L}{2\mu_1\mu}) + \tfrac{\alpha\mu_2^2\gamma^2 L}{2\mu_1\mu}.$$

The bound in Theorem 4.11 has two components: (1) a term decaying linearly to zero; and, (2) a term identifying the neighborhood of convergence. Notice that a larger step length yields a more favorable constant in the linearly decaying term, at the cost of an increase in the size of the neighborhood of convergence. As in the deterministic case, the step length range and rate of SONIA depends on μ_1 and μ_2. Thus, this result is weaker than that of its first-order variant if the update has been unfortunate and generates ill-conditioned matrices. Again, this is seldom observed in practice.

One can establish convergence of stochastic SONIA to the optimal solution $w^\star$ by employing a sequence of step lengths that converge to zero [Robbins and Monro, 1951], but at the slower, sublinear rate. Another way to achieve exact convergence is to employ variance reduced gradient approximations [Johnson and Zhang, 2013, Schmidt et al., 2017], and achieve linear convergence, at the cost of computing the full gradient every so often, or increased storage.

Non-convex Functions

Theorem 4.12. *Suppose that Assumptions 4.1, 4.6, 4.7, 4.9 and 4.10 hold. Let $\{w_k\}$ be the iterates generated by Algorithm 1, where $0 < \alpha_k = \alpha \leq \frac{\mu_1}{\mu_2^2 L}$, and w_0 is the starting point. Then, for all $k \geq 0$,*

$$\mathbb{E}[\tfrac{1}{T}\sum_{k=0}^{T-1} \|\nabla F(w_k)\|^2] \leq \tfrac{2[F(w_0)-\hat{F}]}{\alpha\mu_1 T} + \tfrac{\alpha\mu_2^2\gamma^2 L}{\mu_1} \xrightarrow{T\to\infty} \tfrac{\alpha\mu_2^2\gamma^2 L}{\mu_1}.$$

Theorem 4.12 bounds the average norm squared of the gradient of F, in expectation. The result states that, in expectation, the iterates spend increasingly more time in regions where the objective function has small gradient. The difference with the deterministic setting is that one cannot show convergence to a stationary point; this is due to the variance in the gradient approximation employed. One can establish such convergence under appropriately diminishing step length schedules.

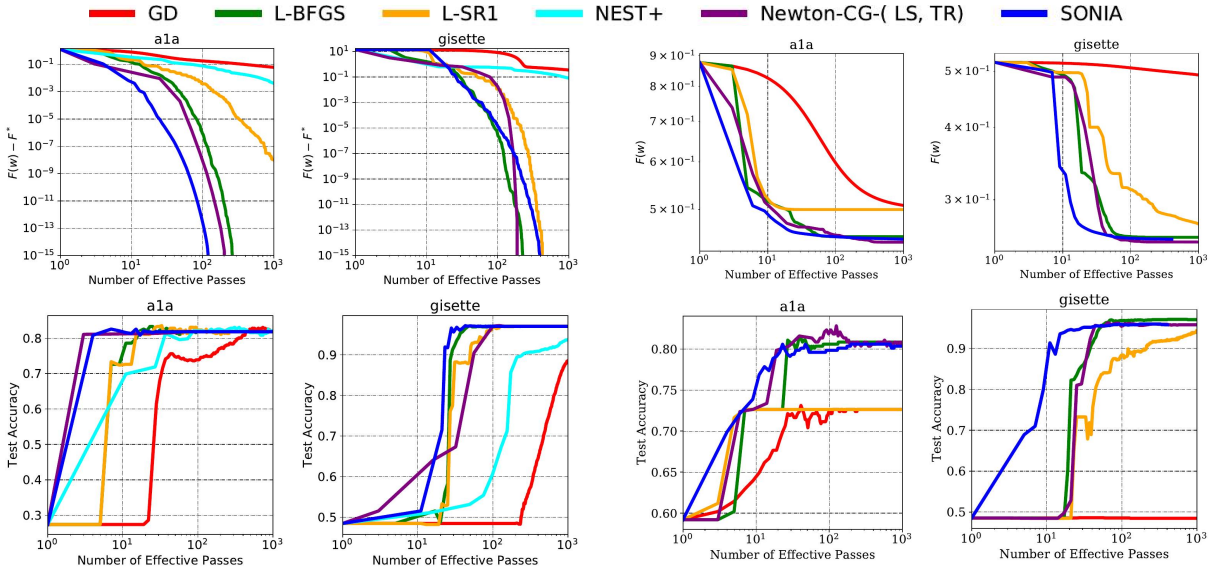
5 Numerical Experiments

In this section, we present numerical experiments[3] on two standard machine learning problems, and compare the empirical performance of SONIA with that of state-of-the-art first- and second-order methods[4], in both the stochastic and deterministic settings. We considered four different classes of problems: (1) deterministic and stochastic logistic regression (stronlgy convex); and, (2) deterministic and stochastic nonlinear least squares (nonconvex), and report results on two standard machine learning datasets[5]. (For brevity we report only a subset of the results here and defer the rest to Appendix D.) We compared the performance of SONIA to algorithms with computational cost and storage requirements linear in both n and d. As such, we did not compare against NCN [Paternain et al., 2019] and full-memory quasi-Newton methods. Our metric of comparison was the number of effective passes (or epochs), which we calculated as the number of function,

[3] All codes to reproduce numerical results are available at https://github.com/OptMLGroup/SONIA

[4] See Appendix C.1 for algorithm details.

[5] a1a and gistte. Available at: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.



(a) Comparison of optimality gap $(F(w)-F^\star)$ and Test Accuracy for different algorithms on Logistic Regression Problems.

(b) Comparison of objective function $(F(w))$ and Test Accuracy for different algorithms on Non-Linear Least Squares Problems.

Figure 1: Deterministic Problems. Datasets: `a1a`, and `gistte`.

gradient and Hessian-vector (or matrix) evaluations; see Appendix B for details. We tuned the hyper-parameters of each method individually for every instance; see Appendix C.3 for a complete description of the tuning efforts. Where applicable, the regularization parameter was chosen from the set $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. The memory size was set to $m = \min\{d, 64\}$, and the truncation parameters were set to $\omega = 10^{-5}$, $\Omega = 10^{8}$ and $\rho_k = \hat{\lambda}_k := \max_i\{[(|\Lambda_k|_\omega^\Omega)^{-1}]_{ii}\}$; we found that these choices gave the best performance. We also performed sensitivity analysis for `SONIA`; see Appendices D.1.1 and D.1.2.

5.1 Deterministic Setting

In the deterministic setting, we compared the performance of `SONIA` to that of Gradient Descent (GD), L-BFGS [Liu and Nocedal, 1989], L-SR1 [Lu, 1996], NEST+ [Nesterov, 2004] and Newton-CG [Nocedal and Wright, 2006]. We implemented the algorithms with adaptive procedures for selecting the step length (e.g., Armijo backtracking) and/or computing the step (e.g., trust-region subroutine) [Nocedal and Wright, 2006]. (Newton-CG was implemented with a line search for strongly convex problems and with a trust region for nonconvex problems.)

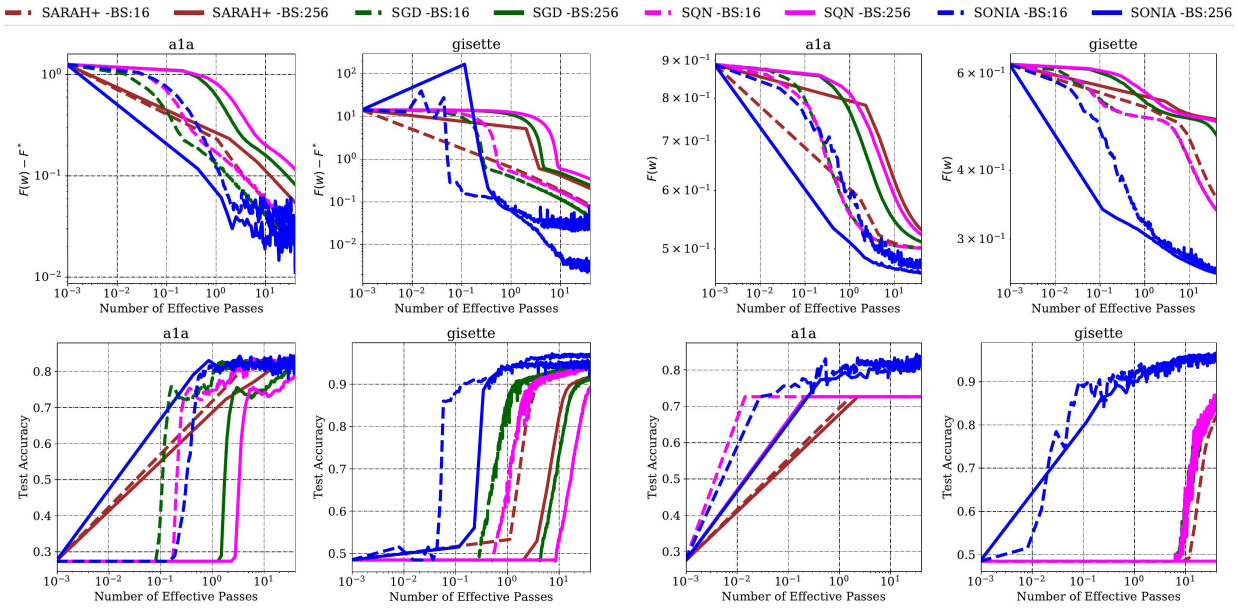
Deterministic Logistic Regression We considered ℓ_2 regularized logistic regression problems, $F(w) = \frac{1}{n}\sum_{i=1}^n \log(1+e^{-y_i x_i^T w}) + \frac{\lambda}{2}\|w\|^2$. Figure 1a shows the performance of the methods in terms of optimality gap $(F(w) - F^\star$[6]$)$ and testing accuracy versus number of effective passes. As is clear, the performance of `SONIA` is on par or better than that of the other methods. Similar behavior was observed on other datasets; see Appendix D.1.

Deterministic Non-Linear Least Squares We considered non-linear least squares problems, $F(w) = \frac{1}{n}\sum_{i=1}^n (y_i - \frac{1}{1+e^{-x_i^T w}})^2$, described in [Xu et al., 2020]. Figure 1b shows the performance of the methods in terms of objective function and testing accuracy versus number of effective passes. As is clear, the performance of `SONIA` is always better than the other methods in the initial stages of training, and the final objective and testing accuracy is comparable to the best method for each problem. We should note that in Figure 1b we report results for a single starting point (as is done in [Xu et al., 2020]); the performance of `SONIA` was robust for all starting points.

5.2 Stochastic Setting

In the stochastic setting, we compared the performance of `SONIA` to that of SGD [Bottou et al., 2018], SARAH [Nguyen et al., 2017] and SQN [Byrd et al., 2016]. We implemented the algorithms with fixed steplength rules, and tuned this parameter as well as the batch size for every problem; see Section C.3 for more details.

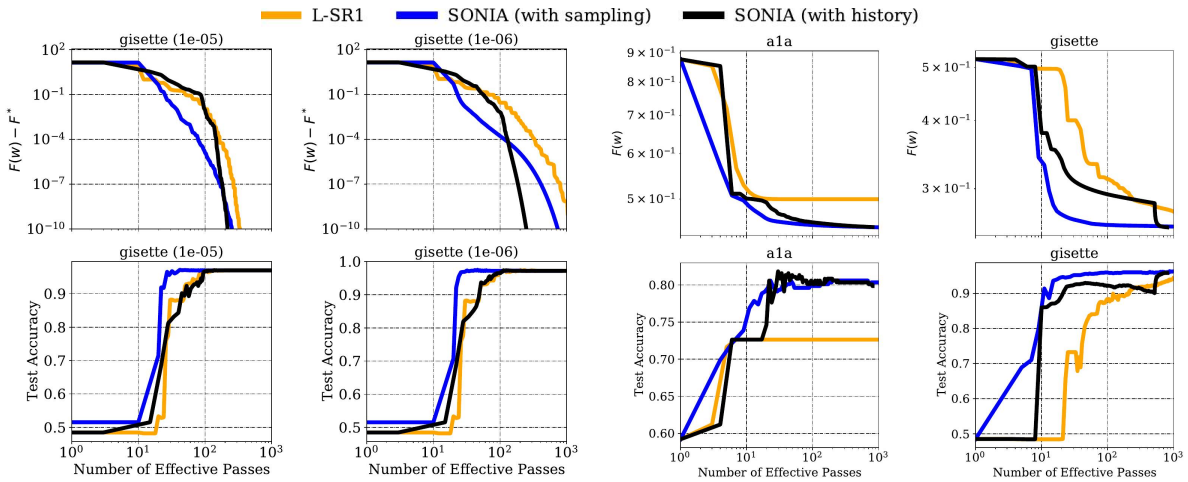
[6] To find $w^\star$ we ran the ASUESA algorithm [Ma et al., 2017]; see Section C.1.



(a) Comparison of optimality gap ($F(w) - F^\star$) and Test Accuracy for different algorithms on Logistic Regression Problems.

(b) Comparison of objective function ($F(w)$) and Test Accuracy for different algorithms on Non-Linear Least Squares Problems.

Figure 2: Stochastic Problems. Datasets: `a1a` and `gistte`.



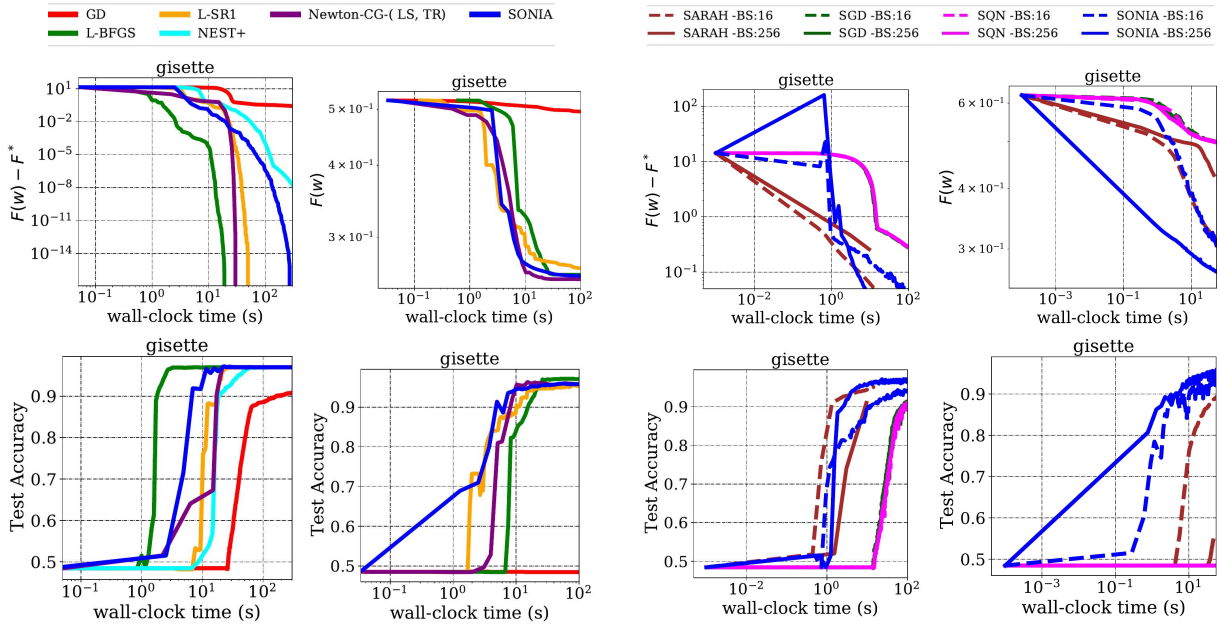
(a) Comparison of L-SR1, `SONIA` (with sampling) and `SONIA` (with history) on Logistic Regression Problems. ($\lambda = 10^{-5}, 10^{-6}$)

(b) Comparison of L-SR1, `SONIA` (with sampling) and `SONIA` (with history) on Non-Linear Least Squares Problems.

Figure 3: Comparison of `SONIA` variants. Datasets: `a1a` and `gistte`.

Stochastic Logistic Regression Figure 2a shows the performance of the stochastic methods on logistic regression problems. We show results for every method in the small batch regime (16) and in the large batch regime (256). As is clear, the stochastic variant of `SONIA` is competitive with the other methods. We should also mention that as predicted by the theory, using a larger batch size (lower variance in the stochastic gradient approximation) allows for `SONIA` to convergence to a smaller neighborhood around the optimal solution. For more results see Section D.3.

Stochastic Non-Linear Least Squares Figure 2b shows the performance of the stochastic methods on (nonconvex) nonlinear least squares problems. As is clear, the stochastic variant of `SONIA` outperforms the other methods for all problems reported. Within a very small number of epochs, `SONIA` is able to achieve high testing accuracy. We attribute the success of `SONIA` in the stochastic nonconvex regime to the fact that useful curvature information is incorporated in the search direction. For more results see Section D.4.



(a) Comparison of objective function and Test Accuracy for different algorithms on Deterministic Logistic Regression (first column) and Non-Linear Least Squares (second column) Problems.

(b) Comparison of objective function and Test Accuracy for different algorithms on Stochastic Logistic Regression (first column) and Non-Linear Least Squares (second column) Problems.

Figure 4: Comparison with respect to wall-clock time time. Dataset: `gistte`.

5.3 Sampling versus History

A fair question to ask is whether the good empirical performance of SONIA is due to sampling curvature pairs at every iteration. In Figure 3 we compare L-SR1, SONIA with sampling at every iteration to construct curvature pairs, and SONIA with limited-memory history. As is clear, SONIA with sampling appears to outperform the other variants suggesting that there is value to sampling and using new local information in the Hessian approximations even though the per iteration cost (in terms of effective passes) of SONIA with sampling is higher than both L-SR1 and SONIA with history.

5.4 Wall-clock Time Comparison

Finally, we investigated the performance of SONIA in terms of wall-clock time. To this end, Figure 4 shows the performance of the methods on the `gisette` dataset for all aforementioned problem classes. As is clear, despite the increased per iteration cost of SONIA, the method is competitive on these machine learning tasks. Similar performance is observed for other datasets; see Appendix D.5.

6 Final Remarks and Future Works

This paper describes a deterministic and stochastic variant of a novel optimization method, SONIA, for empirical risk minimization. The method attempts to bridge the gap between first- and second-order methods by computing a search direction that uses a second-order-type update in one subspace, coupled with a scaled steepest descent step in the orthogonal complement. Numerical results show that the method is efficient in both the deterministic and stochastic settings, and theoretical guarantees confirm that SONIA converges to a stationary point for both strongly convex and nonconvex functions.

Future research directions include: (1) developing adaptive memory variants of SONIA, (2) exploring stochastic SONIA variants that use an adaptive number of samples for gradient/Hessian approximations (leveraging ideas from [Byrd et al., 2012, Mokhtari et al., 2016, Jahani et al., 2020, Friedlander and Schmidt, 2012, Bollapragada et al., 2018]), or that employ variance reduced gradients, and (3) a thorough numerical investigation for deep learning problems to test the limits of the methods.

Acknowledgements

This work was partially supported by the U.S. National Science Foundation, under award numbers NSF:CCF:1618717 and NSF:CCF:1740796.

References

[Berahas et al., 2020] Berahas, A. S., Bollapragada, R., and Nocedal, J. (2020). An investigation of Newton-Sketch and subsampled Newton methods. *Optimization Methods and Software*, 35(4):661–680.

[Berahas et al., 2019] Berahas, A. S., Jahani, M., and Takáč, M. (2019). Quasi-newton methods for deep learning: Forget the past, just sample. *arXiv preprint arXiv:1901.09997*.

[Berahas et al., 2016] Berahas, A. S., Nocedal, J., and Takác, M. (2016). A multi-batch l-bfgs method for machine learning. In *Advances in Neural Information Processing Systems*, pages 1055–1063.

[Berahas and Takáč, 2020] Berahas, A. S. and Takáč, M. (2020). A robust multi-batch L-BFGS method for machine learning. *Optimization Methods and Software*, 35(1):191–219.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[Bollapragada et al., 2018] Bollapragada, R., Byrd, R., and Nocedal, J. (2018). Adaptive sampling strategies for stochastic optimization. *SIAM Journal on Optimization*, 28(4):3312–3343.

[Bollapragada et al., 2016] Bollapragada, R., Byrd, R. H., and Nocedal, J. (2016). Exact and inexact subsampled newton methods for optimization. *IMA Journal of Numerical Analysis*.

[Bottou et al., 2018] Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311.

[Brust et al., 2017] Brust, J., Erway, J. B., and Marci, R. F. (2017). On solving L-SR1 trust-region subproblems. *Computational Optimization and Applications*, 66:245—-266.

[Byrd et al., 2011] Byrd, R. H., Chin, G. M., Neveitt, W., and Nocedal, J. (2011). On the use of stochastic hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3):977–995.

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

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

[Byrd et al., 1994] Byrd, R. H., Nocedal, J., and Schnabel, R. B. (1994). Representations of quasi-newton matrices and their use in limited memory methods. *Math. Program.*, 63:129–156.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27.

[Curtis, 2016] Curtis, F. (2016). A self-correcting variable-metric algorithm for stochastic optimization. In *International Conference on Machine Learning*, pages 632–641.

[Dennis and Moré, 1977] Dennis, Jr, J. E. and Moré, J. J. (1977). Quasi-newton methods, motivation and theory. *SIAM review*, 19(1):46–89.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159.

[Erway et al., 2019] Erway, J. B., Griffin, J., Marcia, R. F., and Omheni, R. (2019). Trust-region algorithms for training responses: machine learning methods using indefinite hessian approximations. *Optimization Methods and Software*, pages 1–28.

[Fletcher, 1987] Fletcher, R. (1987). *Practical Methods of Optimization*. John Wiley & Sons, New York, 2 edition.

[Fountoulakis and Tappenden, 2018] Fountoulakis, K. and Tappenden, R. (2018). A flexible coordinate descent method. *Computational Optimization and Applications*, 70:351—-394.

[Friedlander and Schmidt, 2012] Friedlander, M. P. and Schmidt, M. (2012). Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405.

[Friedman et al., 2001] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

[Gower et al., 2019] Gower, R., Koralev, D., Lieder, F., and Richtárik, P. (2019). Rsn: Randomized subspace newton. In *Advances in Neural Information Processing Systems*, pages 616–625.

[Jahani et al., 2020] Jahani, M., He, X., Ma, C., Mokhtari, A., Mudigere, D., Ribeiro, A., and Takác, M. (2020). Efficient distributed hessian free algorithm for large-scale empirical risk minimization via accumulating sample strategy. In *International Conference on Artificial Intelligence and Statistics*, pages 2634–2644. PMLR.

[Jahani et al., 2019] Jahani, M., Nazari, M., Rusakov, S., Berahas, A. S., and Takáč, M. (2019). Scaling up quasi-newton algorithms: Communication efficient distributed sr1. *arXiv preprint arXiv:1905.13096*.

[Johnson and Zhang, 2013] Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323.

[Keskar and Berahas, 2016] Keskar, N. S. and Berahas, A. S. (2016). adaQN: An adaptive quasi-newton algorithm for training rnns. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 1–16. Springer.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[Liu and Nocedal, 1989] Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

[Lu, 1996] Lu, X. (1996). *A study of the limited memory SR1 method in practice*. University of Colorado at Boulder.

[Ma et al., 2017] Ma, C., Gudapati, N. V. C., Jahani, M., Tappenden, R., and Takáč, M. (2017). Underestimate sequences via quadratic averaging. *arXiv preprint arXiv:1710.03695*.

[Martens, 2010] Martens, J. (2010). Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742.

[Mokhtari et al., 2016] Mokhtari, A., Daneshmand, H., Lucchi, A., Hofmann, T., and Ribeiro, A. (2016). Adaptive newton method for empirical risk minimization to statistical accuracy. In *Advances in Neural Information Processing Systems*, pages 4062–4070.

[Mokhtari and Ribeiro, 2015] Mokhtari, A. and Ribeiro, A. (2015). Global convergence of online limited memory bfgs. *The Journal of Machine Learning Research*, 16(1):3151–3181.

[Nesterov, 2004] Nesterov, Y. (2004). *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87 of *Applied Optimization*. Springer (Originally published by Kluwer Academic Publishers). doi:10.1007/978-1-4419-8853-9.

[Nesterov, 2013a] Nesterov, Y. (2013a). Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161.

[Nesterov, 2013b] Nesterov, Y. (2013b). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

[Nguyen et al., 2017] Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. (2017). SARAH: a novel method for machine learning problems using stochastic recursive gradient. In *Advances in neural information processing systems*, volume 70, page 2613–2621.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer Series in Operations Research. Springer, second edition.

[Paternain et al., 2019] Paternain, S., Mokhtari, A., and Ribeiro, A. (2019). A newton-based method for nonconvex optimization with fast evasion of saddle points. *SIAM Journal on Optimization*, 29(1):343–368.

[Recht et al., 2011] Recht, B., Re, C., Wright, S., and Niu, F. (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701.

[Richtárik and Takáč, 2014] Richtárik, P. and Takáč, M. (2014). Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[Roosta-Khorasani and Mahoney, 2018] Roosta-Khorasani, F. and Mahoney, M. W. (2018). Sub-sampled newton methods. *Mathematical Programming*.

[Schmidt et al., 2017] Schmidt, M., Le Roux, N., and Bach, F. (2017). Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112.

[Schraudolph et al., 2007] Schraudolph, N. N., Yu, J., and Günter, S. (2007). A stochastic quasi-newton method for online convex optimization. In *Artificial Intelligence and Statistics*, pages 436–443.

[Tappenden et al., 2016] Tappenden, R., Richtárik, P., and Gondzio, J. (2016). Inexact coordinate descent: Complexity and preconditioning. *Journal of Optimization Theory and Applications*, 170:144—-176.

[Woodruff, 2014] Woodruff, D. P. (2014). Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157.

[Xu et al., 2020] Xu, P., Roosta, F., and Mahoney, M. W. (2020). Second-order optimization for non-convex machine learning: An empirical study. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 199–207. SIAM.