Global Convergence of Online Limited Memory BFGS

Aryan Mokhtari ARYANM@SEAS.UPENN.EDU
Alejandro Ribeiro ARIBEIRO@SEAS.UPENN.EDU
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104, USA

Editor:

Abstract

Global convergence of an online (stochastic) limited memory version of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method for solving optimization problems with stochastic objectives that arise in large scale machine learning is established. Lower and upper bounds on the Hessian eigenvalues of the sample functions are shown to suffice to guarantee that the curvature approximation matrices have bounded determinants and traces, which, in turn, permits establishing convergence to optimal arguments with probability 1. Numerical experiments on support vector machines with synthetic data showcase reductions in convergence time relative to stochastic gradient descent algorithms as well as reductions in storage and computation relative to other online quasi-Newton methods. Experimental evaluation on a search engine advertising problem corroborates that these advantages also manifest in practical applications.

Keywords: Quasi-Newton methods, large-scale optimization, stochastic optimization, support vector machines.

1. Introduction

Many problems in Machine Learning can be reduced to the minimization of a stochastic objective defined as an expectation over a set of random functions (Bottou and Cun (2005); Bottou (2010); Shalev-Shwartz and Srebro (2008); Mokhtari and Ribeiro (2014b)). Specifically, consider an optimization variable $\mathbf{w} \in \mathbb{R}^n$ and a random variable $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p$ that determines the choice of a function $f(\mathbf{w}, \boldsymbol{\theta}) : \mathbb{R}^{n \times p} \to \mathbb{R}$. Stochastic optimization problems entail determination of the argument $\mathbf{w}^*$ that minimizes the expected value $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$,

$$\mathbf{w}^* \;:=\; \operatorname*{argmin}_{\mathbf{w}} \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})] \;:=\; \operatorname*{argmin}_{\mathbf{w}} F(\mathbf{w}). \tag{1}$$

We refer to $f(\mathbf{w}, \boldsymbol{\theta})$ as the random or instantaneous functions and to $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$ as the average function. A canonical class of problems having this form are support vector machines (SVMs) that reduce binary classification to the determination of a hyperplane that separates points in a given training set; see, e.g., (Vapnik (1999); Bottou (2010); Boser et al. (ACM, 1992)). In that case $\boldsymbol{\theta}$ denotes individual training samples, $f(\mathbf{w}, \boldsymbol{\theta})$ the loss of choosing the hyperplane defined by $\mathbf{w}$, and $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$ the mean loss across all elements of the training set. The optimal argument $\mathbf{w}^*$ is the optimal linear classifier.

Numerical evaluation of objective function gradients $\nabla_{\mathbf{w}} F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\nabla_{\mathbf{w}} f(\mathbf{w}, \boldsymbol{\theta})]$ is intractable when the cardinality of Θ is large, as is the case, e.g., when SVMs are trained on large sets. This motivates the use of algorithms relying on stochastic gradients that provide gradient estimates based on small data subsamples. For the purpose of this paper stochastic optimization algorithms can be divided into three categories: Stochastic gradient descent (SGD) and related first order methods, stochastic Newton methods, and stochastic quasi-Newton methods.

SGD is the most popular method used to solve stochastic optimization problems (Bottou (2010); Shalev-Shwartz et al. (2007); Zhang (2004); LeRoux et al. (2012)). However, as we consider problems of ever larger dimension their slow convergence times have limited their practical appeal and fostered the search for alternatives. In this regard, it has to be noted that SGD is slow because of both, the use of gradients as descent directions and their replacement by random estimates. Several alternatives have been proposed to deal with randomness in an effort to render the convergence times of SGD closer to the faster convergence times of gradient descent (Syski (1983); Konecny and Richtarik (2013); Zhang et al. (2013a)). These SGD variants succeed in reducing randomness and end up exhibiting the asymptotic convergence rate of gradient descent. Although they improve asymptotic convergence rates, the latter methods are still often slow in practice. This is not unexpected. Reducing randomness is of no use when the function $F(\mathbf{w})$ has a challenging curvature profile. In these ill-conditioned functions SGD is limited by the already slow convergence times of deterministic gradient descent. The golden standard to deal with ill-conditioned functions in a deterministic setting is Newton's method. However, unbiased stochastic estimates of Newton steps can't be computed in general. This fact limits the application of stochastic Newton methods to problems with specific structure (Birge et al. (1995); Zargham et al. (2013)).

If SGD is slow to converge and stochastic Newton can't be used in general, the remaining alternative is to modify deterministic quasi-Newton methods that speed up convergence times relative to gradient descent without using Hessian evaluations (J. E. Dennis and More (1974); Powell (1971); Byrd et al. (1987); Nocedal and Wright (1999)). This has resulted in the development of the stochastic quasi-Newton methods known as online (o) Broyden-Fletcher-Goldfarb-Shanno (BFGS) (Schraudolph et al. (2007)), regularized stochastic BFGS (RES) (Mokhtari and Ribeiro (2014a)), and online limited memory (oL)BFGS (Schraudolph et al. (2007)) which occupy the middle ground of broad applicability irrespective of problem structure and conditioning. All three of these algorithms extend BFGS by using stochastic gradients both as descent directions and constituents of Hessian estimates. The oBFGS algorithm is a direct generalization of BFGS that uses stochastic gradients in lieu of deterministic gradients. RES differs in that it further modifies BFGS to yield an algorithm that retains its convergence advantages while improving theoretical convergence guarantees and numerical behavior. The oLBFGS method uses a modification of BFGS to reduce the computational cost of each iteration.

An important observation here is that in trying to adapt to the changing curvature of the objective, stochastic quasi-Newton methods may end up exacerbating the problem. Indeed, since Hessian estimates are stochastic, it is possible to end up with almost singular Hessian estimates. The corresponding small eigenvalues then result in a catastrophic amplification of the noise which nullifies progress made towards convergence. This is not a minor problem. In oBFGS this possibility precludes convergence analyses (Bordes et al. (2009); Schraudolph et al. (2007)) and may result in erratic numerical behavior (Mokhtari and Ribeiro (2014a)). As a matter of fact, the main motivation for the introduction of RES is to avoid this catastrophic noise amplification so as to retain smaller convergence times while ensuring that optimal arguments are found with probability 1 (Mokhtari and Ribeiro (2014a)). However valuable, the convergence guarantees of RES and the convergence time advantages of oBFGS and RES are tainted by an iteration cost of order $O(n^2)$ and $O(n^3)$, respectively, which precludes their use in problems where n is very large. In deterministic settings this problem is addressed by limited memory (L)BFGS (Dong C. and Nocedal (1989)) which can be easily generalized to develop the oLBFGS algorithm (Schraudolph et al. (2007)). Numerical tests of oLBFGS are promising but theoretical convergence characterizations are still lacking. The main contribution of this paper is to show that oLBFGS converges with probability 1 to optimal arguments across realizations of the random variables θ. This is the same convergence guarantee provided for RES and is in marked contrast with oBFGS, which fails to converge if not properly regularized. Convergence guarantees for oLBFGS do not require such measures.

We begin the paper with brief discussions of deterministic BFGS (Section 2) and LBFGS (Section 2.1) and the introduction of oLBFGS (Section 2.2). The fundamental idea in BFGS and oLBFGS

is to continuously satisfy a secant condition while staying close to previous curvature estimates. They differ in that BFGS uses all past gradients to estimate curvature while oLBFGS uses a fixed moving window of past gradients. The use of this window reduces memory and computational cost (Appendix A). The difference between LBFGS and oLBFGS is the use of stochastic gradients in lieu of their deterministic counterparts.

Convergence properties of oLBFGS are then analyzed (Section 3). Under the assumption that the sample functions $f(\mathbf{w}, \boldsymbol{\theta})$ are strongly convex we show that the trace and determinant of the Hessian approximations computed by oLBFGS are upper and lower bounded, respectively (Lemma 3). These bounds are then used to limit the range of variation of the ratio between the Hessian approximations' largest and smallest eigenvalues (Lemma 4). In turn, this condition number limit is shown to be sufficient to prove convergence to the optimal argument $\mathbf{w}^*$ with probability 1 over realizations of the sample functions (Theorem 6). This is an important result because it ensures that oLBFGS doesn't suffer from the numerical problems that hinder oBFGS. We complement this almost sure convergence result with a characterization of the convergence rate which is shown to be at least $O(1/t)$ in expectation (Theorem 7). It is fair to emphasize that, different from the deterministic case, the convergence rate of oLBFGS is not better than the convergence rate of SGD. This is not a limitation of our analysis. The difference between stochastic and regular gradients introduces a noise term that dominates convergence once we are close to the optimum, which is where superlinear convergence rates manifest. In fact, the same convergence rate would be observed if exact Hessians were available. The best that can be proven of oLBFGS is that the convergence rate is not worse than that of SGD. Given that theoretical guarantees only state that the curvature correction does not exacerbate the problem's condition it is perhaps fairer to describe oLBFGS as an adaptive reconditioning strategy instead of a stochastic quasi-Newton method. The latter description refers to the genesis of the algorithm. The former is a more accurate description of its actual behavior.

To show the advantage of using oLBFGS as an adaptive reconditioning strategy we develop its application to SVM problems (Section 4) and perform a comparative numerical analysis with synthetic data. The conclusions of this numerical analysis are that oLBFGS performs as well as oBFGS and RES while outperforming SGD when convergence is measured with respect to the number of feature vectors processed. In terms of computation time, oLBFGS outperforms all three methods, SGD, oBFGS, and RES. The advantages of oLBFGS grow with the dimension of the feature vector and can be made arbitrarily large (Section 4.1). To further substantiate numerical claims we use oLBFGS to train a logistic regressor to predict the click through rate in a search engine advertising problem (Section 5). The logistic regression uses a heterogeneous feature vector with 174,026 binary entries that describe the user, the search, and the advertisement (Section 5.1). Being a large scale problem with heterogeneous data, the condition number of the logistic log likelihood objective is large and we expect to see significant advantages of oLBFGS relative to SGD. This expectation is fulfilled. The oLBFGS algorithm trains the regressor using less than 1% of the data required by SGD to obtain similar classification accuracy. (Section 5.3). We close the paper with concluding remarks (Section 6).

Notation Lowercase boldface $\mathbf{v}$ denotes a vector and uppercase boldface $\mathbf{A}$ a matrix. We use $\|\mathbf{v}\|$ to denote the Euclidean norm of vector $\mathbf{v}$ and $\|\mathbf{A}\|$ to denote the Euclidean norm of matrix $\mathbf{A}$. The trace of $\mathbf{A}$ is written as $\text{tr}(\mathbf{A})$ and the determinant as $\det(\mathbf{A})$. We use $\mathbf{I}$ for the identity matrix of appropriate dimension. The notation $\mathbf{A} \succeq \mathbf{B}$ implies that the matrix $\mathbf{A} - \mathbf{B}$ is positive semidefinite. The operator $\mathbb{E}_{\mathbf{x}}[\cdot]$ stands in for expectation over random variable $\mathbf{x}$ and $\mathbb{E}[\cdot]$ for expectation with respect to the distribution of a stochastic process.

2. Algorithm definition

Recall the definitions of the sample functions $f(\mathbf{w}, \boldsymbol{\theta})$ and the average function $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$. We assume the sample functions $f(\mathbf{w}, \boldsymbol{\theta})$ are strongly convex for all $\boldsymbol{\theta}$. This implies the objective

function $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$, being an average of the strongly convex sample functions, is also strongly convex. We define the gradient $\mathbf{s}(\mathbf{w}) := \nabla F(\mathbf{w})$ of the average function $F(\mathbf{w})$ and assume that it can be computed as

$$\mathbf{s}(\mathbf{w}) := \nabla F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\nabla f(\mathbf{w}, \boldsymbol{\theta})]. \tag{2}$$

Since the function $F(\mathbf{w})$ is strongly convex, gradients $\mathbf{s}(\mathbf{w})$ are descent directions that can be used to find the optimal argument $\mathbf{w}^*$ in (1). Introduce then a time index t, a step size ϵ_t, and a positive definite matrix $\mathbf{B}_t^{-1} \succ 0$ to define a generic descent algorithm through the iteration

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t - \epsilon_t \; \mathbf{B}_t^{-1} \mathbf{s}(\mathbf{w}_t) \;=\; \mathbf{w}_t - \epsilon_t \; \mathbf{d}_t. \tag{3}$$

where we have also defined the descent step $\mathbf{d}_t = \mathbf{B}_t^{-1}\mathbf{s}(\mathbf{w}_t)$. When $\mathbf{B}_t^{-1} = \mathbf{I}$ is the identity matrix, (3) reduces to gradient descent. When $\mathbf{B}_t = \mathbf{H}(\mathbf{w}_t) := \nabla^2 F(\mathbf{w}_t)$ is the Hessian of the objective function, (3) defines Newton's algorithm. In this paper we focus on quasi-Newton methods whereby we attempt to select matrices $\mathbf{B}_t$ close to the Hessian $\mathbf{H}(\mathbf{w}_t)$. Various methods are known to select matrices $\mathbf{B}_t$, including those by Broyden e.g., Broyden et al. (1973); Davidon, Fletcher, and Powell (DFP) e.g., Fletcher (2013); and Broyden, Fletcher, Goldfarb, and Shanno (BFGS) e.g., Byrd et al. (1987); Powell (1971). We work with the matrices $\mathbf{B}_t$ used in BFGS since they have been observed to work best in practice (see Byrd et al. (1987)).

In BFGS, the function's curvature $\mathbf{B}_t$ is approximated by a finite difference. Let $\mathbf{v}_t$ denote the variable variation at time t and $\mathbf{r}_t$ the gradient variation at time t which are respectively defined as

$$\mathbf{v}_t := \mathbf{w}_{t+1} - \mathbf{w}_t, \qquad \mathbf{r}_t := \mathbf{s}(\mathbf{w}_{t+1}) - \mathbf{s}(\mathbf{w}_t). \tag{4}$$

We select the matrix $\mathbf{B}_{t+1}$ to be used in the next time step so that it satisfies the secant condition $\mathbf{B}_{t+1}\mathbf{v}_t = \mathbf{r}_t$. The rationale for this selection is that the Hessian $\mathbf{H}(\mathbf{w}_t)$ satisfies this condition for $\mathbf{w}_{t+1}$ tending to $\mathbf{w}_t$. Notice however that the secant condition $\mathbf{B}_{t+1}\mathbf{v}_t = \mathbf{r}_t$ is not enough to completely specify $\mathbf{B}_{t+1}$. To resolve this indeterminacy, matrices $\mathbf{B}_{t+1}$ in BFGS are also required to be as close as possible to the previous Hessian approximation $\mathbf{B}_t$ in terms of differential entropy. These conditions can be resolved in closed form leading to the explicit expression – see, e.g., Nocedal and Wright (1999) –,

$$\mathbf{B}_{t+1} = \mathbf{B}_t + \frac{\mathbf{r}_t \mathbf{r}_t^T}{\mathbf{v}_t^T \mathbf{r}_t} - \frac{\mathbf{B}_t \mathbf{v}_t \mathbf{v}_t^T \mathbf{B}_t}{\mathbf{v}_t^T \mathbf{B}_t \mathbf{v}_t}. \tag{5}$$

While the expression in (5) permits updating the Hessian approximations $\mathbf{B}_{t+1}$, implementation of the descent step in (3) requires its inversion. This can be avoided by using the Sherman-Morrison formula in (5) to write

$$\mathbf{B}_{t+1}^{-1} \;=\; \mathbf{Z}_t^T \; \mathbf{B}_t^{-1} \; \mathbf{Z}_t \;+\; \rho_t \; \mathbf{v}_t \mathbf{v}_t^T, \tag{6}$$

where we defined the scalar ρ_t and the matrix $\mathbf{Z}_t$ as

$$\rho_t \;:=\; \frac{1}{\mathbf{v}_t^T \mathbf{r}_t}\;, \qquad \mathbf{Z}_t \;:=\; \mathbf{I} - \rho_t \mathbf{r}_t \mathbf{v}_t^T. \tag{7}$$

The updates in (5) and (6) require the inner product of the gradient and variable variations to be positive, i.e., $\mathbf{v}_t^T \mathbf{r}_t > 0$. This is always true if the objective $F(\mathbf{w})$ is strongly convex and further implies that $\mathbf{B}_{t+1}^{-1}$ stays positive definite if $\mathbf{B}_t^{-1} \succ \mathbf{0}$, Nocedal and Wright (1999).

Each BFGS iteration has a cost of $O(n^2)$ arithmetic operations. This is less than the $O(n^3)$ of each step in Newton's method but more than the $O(n)$ cost of each gradient descent iteration. In general, the relative convergence rates are such that the total computational cost of BFGS to achieve a target accuracy is smaller than the corresponding cost of gradient descent. Still, alternatives to reduce the computational cost of each iteration are of interest for large scale problems. Likewise, BFGS requires storage and propagation of the $O(n^2)$ elements of $\mathbf{B}_t^{-1}$, whereas gradient descent requires storage of $O(n)$ gradient elements only. This motivates alternatives that have smaller memory footprints. Both of these objectives are accomplished by the limited memory (L)BFGS algorithm that we describe in the following section.

2.1 LBFGS: Limited memory BFGS

As it follows from (6), the updated Hessian inverse approximation $\mathbf{B}_t^{-1}$ depends on $\mathbf{B}_{t-1}^{-1}$ and the curvature information pairs $\{\mathbf{v}_{t-1},\ \mathbf{r}_{t-1}\}$. In turn, to compute $\mathbf{B}_{t-1}^{-1}$, the estimate $\mathbf{B}_{t-2}^{-1}$ and the curvature pair $\{\mathbf{v}_{t-2}, \mathbf{r}_{t-2}\}$ are used. Proceeding recursively, it follows that $\mathbf{B}_t^{-1}$ is a function of the initial approximation $\mathbf{B}_0^{-1}$ and all previous t curvature information pairs $\{\mathbf{v}_u,\ \mathbf{r}_u\}_{u=0}^{t-1}$. The idea in LBFGS is to restrict the use of past curvature information to the last τ pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=t-\tau}^{t-1}$. Since earlier iterates $\{\mathbf{v}_u, \mathbf{r}_u\}$ with $u < t-\tau$ are likely to carry little information about the curvature at the current iterate $\mathbf{w}_t$, this restriction is expected to result in a minimal performance penalty.

For a precise definition, pick a positive definite matrix $\mathbf{B}_{t,0}^{-1}$ as the initial Hessian inverse approximation at step t. Proceed then to perform τ updates of the form in (6) using the last τ curvature information pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=t-\tau}^{t-1}$. Denoting as $\mathbf{B}_{t,u}^{-1}$ the curvature approximation after u updates are performed we have that the refined matrix approximation $\mathbf{B}_{t,u+1}^{-1}$ is given by [cf. (6)]

$$\mathbf{B}_{t,u+1}^{-1} \ = \mathbf{Z}_{t-\tau+u}^T\ \mathbf{B}_{t,u}^{-1}\ \mathbf{Z}_{t-\tau+u}\ +\ \rho_{t-\tau+u}\ \mathbf{v}_{t-\tau+u}\,\mathbf{v}_{t-\tau+u}^T, \tag{8}$$

where $u = 0, \ldots, \tau-1$ and the constants $\rho_{t-\tau+u}$ and rank-one plus identity matrices $\mathbf{Z}_{t-\tau+u}$ are as given in (7). The inverse Hessian approximation $\mathbf{B}_t^{-1}$ to be used in (3) is the one yielded after completing the τ updates in (8), i.e., $\mathbf{B}_t^{-1} = \mathbf{B}_{t,\tau}^{-1}$. Observe that when $t < \tau$ there are not enough pairs $\{\mathbf{v}_u, \mathbf{r}_u\}$ to perform τ updates. In such case we just redefine $\tau = t$ and proceed to use the $t = \tau$ available pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=0}^{t-1}$.

Implementation of the product $\mathbf{B}_t^{-1}\mathbf{s}(\mathbf{w}_t)$ in (3) for matrices $\mathbf{B}_t^{-1} = \mathbf{B}_{t,\tau}^{-1}$ obtained from the recursion in (8) does not need explicit computation of the matrix $\mathbf{B}_{t,\tau}^{-1}$. Although the details are not straightforward, observe that each iteration in (8) is similar to a rank-one update and that as such it is not unreasonable to expect that the product $\mathbf{B}_t^{-1}\mathbf{s}(\mathbf{w}_t) = \mathbf{B}_{t,\tau}^{-1}\mathbf{s}(\mathbf{w}_t)$ can be computed using τ recursive inner products. Assuming that this is possible, the implementation of the recursion in (8) doesn't need computation and storage of prior matrices $\mathbf{B}_{t-1}^{-1}$. Rather, it suffices to keep the τ most recent curvature information pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=t-\tau}^{t-1}$, thus reducing storage requirements from $O(n^2)$ to $O(\tau n)$. Furthermore, each of these inner products can be computed at a cost of n operations yielding a total computational cost of $O(\tau n)$ per LBFGS iteration. Hence, LBFGS decreases both the memory requirements and the computational cost of each iteration from the $O(n^2)$ required by regular BFGS to $O(\tau n)$. We present the details of this iteration in the context of the online (stochastic) LBFGS that we introduce in the following section.

2.2 Online (Stochastic) Limited memory BFGS

To implement (3) and (8) we need to compute gradients $\mathbf{s}(\mathbf{w}_t)$. This is impractical when the number of functions $f(\mathbf{w}, \boldsymbol{\theta})$ is large, as is the case in most stochastic problems of practical interest and motivates the use of stochastic gradients in lieu of actual gradients. Consider a given set of L realizations $\tilde{\boldsymbol{\theta}} = [\boldsymbol{\theta}_1; ...; \boldsymbol{\theta}_L]$ and define the stochastic gradient of $F(\mathbf{w})$ at $\mathbf{w}$ given samples $\tilde{\boldsymbol{\theta}}$ as

$$\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) := \frac{1}{L}\sum_{l=1}^{L} \nabla f(\mathbf{w}, \boldsymbol{\theta}_l). \tag{9}$$

In oLBFGS we use stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ for descent directions and curvature estimators. In particular, the descent iteration in (3) is replaced by the descent iteration

$$\mathbf{w}_{t+1} \ = \ \mathbf{w}_t - \epsilon_t\ \hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) \ = \ \mathbf{w}_t - \epsilon_t \hat{\mathbf{d}}_t, \tag{10}$$

where $\tilde{\boldsymbol{\theta}}_t = [\boldsymbol{\theta}_{t1}; ...; \boldsymbol{\theta}_{tL}]$ is the set of samples used at step t to compute the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ as per (9) and the matrix $\hat{\mathbf{B}}_t^{-1}$ is a function of past stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}_u, \tilde{\boldsymbol{\theta}}_u)$ with

$u \leq t$ instead of a function of past gradients $\mathbf{s}(\mathbf{w}_u)$ as in (3). As we also did in (3) we have defined the stochastic step $\hat{\mathbf{d}}_t := \hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ to simplify upcoming discussions.

To properly specify $\hat{\mathbf{B}}_t^{-1}$ we define the stochastic gradient variation $\hat{\mathbf{r}}_t$ at time t as the difference between the stochastic gradients $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t)$ and $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ associated with subsequent iterates $\mathbf{w}_{t+1}$ and $\mathbf{w}_t$ and the *common* set of samples $\tilde{\boldsymbol{\theta}}_t$ [cf. (4)],

$$\hat{\mathbf{r}}_t := \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t). \tag{11}$$

Observe that $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is the stochastic gradient used at time t in (10) but that $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t)$ is computed solely for the purpose of determining the stochastic gradient variation. The perhaps more natural definition $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_{t+1}) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ for the stochastic gradient variation, which relies on the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_{t+1})$ used at time $t+1$ in (10) is not sufficient to guarantee convergence; see e.g.,(Schraudolph et al. (2007); Mokhtari and Ribeiro (2014a)).

To define the oLBFGS algorithm we just need to provide stochastic versions of the definitions in (7) and (8). The scalar constants and identity plus rank-one matrices in (7) are redefined to the corresponding stochastic quantities

$$\hat{\rho}_{t-\tau+u} \;=\; \frac{1}{\mathbf{v}_{t-\tau+u}^T \hat{\mathbf{r}}_{t-\tau+u}} \quad \text{and} \quad \hat{\mathbf{Z}}_{t-\tau+u} \;=\; \mathbf{I} - \hat{\rho}_{t-\tau+u}\hat{\mathbf{r}}_{t-\tau+u}\mathbf{v}_{t-\tau+u}^T, \tag{12}$$

whereas the LBFGS matrix $\mathbf{B}_t^{-1} = \mathbf{B}_{t,\tau}^{-1}$ in (8) is replaced by the oLBFGS Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ which we define as the outcome of τ recursive applications of the update,

$$\hat{\mathbf{B}}_{t,u+1}^{-1} \;=\; \hat{\mathbf{Z}}_{t-\tau+u}^T\, \hat{\mathbf{B}}_{t,u}^{-1}\, \hat{\mathbf{Z}}_{t-\tau+u} + \hat{\rho}_{t-\tau+u}\, \mathbf{v}_{t-\tau+u}\, \mathbf{v}_{t-\tau+u}^T, \tag{13}$$

where the initial matrix $\hat{\mathbf{B}}_{t,0}^{-1}$ is given and the time index is $u = 0, \ldots, \tau-1$. The oLBFGS algorithm is defined by the stochastic descent iteration in (10) with matrices $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ computed by τ recursive applications of (13). Except for the fact that they use stochastic variables, (10) and (13) are identical to (3) and (8). Thus, as is the case in (3), the Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1}$ in (13) is a function of the initial Hessian inverse approximation $\mathbf{B}_{t,0}^{-1}$ and the τ most recent curvature information pairs $\{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=t-\tau}^{t-1}$. Likewise, when $t < \tau$ there are not enough pairs $\{\mathbf{v}_u, \hat{\mathbf{r}}_u\}$ to perform τ updates. In such case we just redefine $\tau = t$ and proceed to use the $t = \tau$ available pairs $\{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=0}^{t-1}$. We also point out that the update in (13) necessitates $\hat{\mathbf{r}}_u^T\mathbf{v}_u > 0$ for all time indexes u. This is true as long as the instantaneous functions $f(\mathbf{w}, \boldsymbol{\theta})$ are strongly convex with respect to $\mathbf{w}$ as we show in Lemma 2.

The equations in (10) and (13) are used conceptually but not in practical implementations. For the latter we exploit the structure of (13) to rearrange the terms in the computation of the product $\hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. To see how this is done consider the recursive update for the Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1}$ in (13) and make $u = \tau - 1$ to write

$$\hat{\mathbf{B}}_t^{-1} \;=\; \hat{\mathbf{B}}_{t,\tau}^{-1} \;=\; \left(\hat{\mathbf{Z}}_{t-1}^T\right) \hat{\mathbf{B}}_{t,\tau-1}^{-1} \left(\hat{\mathbf{Z}}_{t-1}\right) + \hat{\rho}_{t-1}\, \mathbf{v}_{t-1}\, \mathbf{v}_{t-1}^T. \tag{14}$$

Equation (14) shows the relation between the Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1}$ and the $(\tau-1)$st updated version of the initial Hessian inverse approximation $\hat{\mathbf{B}}_{t,\tau-1}^{-1}$ at step t. Set now $u = \tau - 2$ in (13) to express $\hat{\mathbf{B}}_{t,\tau-1}^{-1}$ in terms of $\hat{\mathbf{B}}_{t,\tau-2}^{-1}$ and substitute the result in (14) to rewrite $\hat{\mathbf{B}}_t^{-1}$ as

$$\hat{\mathbf{B}}_t^{-1} \;=\; \left(\hat{\mathbf{Z}}_{t-1}^T\hat{\mathbf{Z}}_{t-2}^T\right) \hat{\mathbf{B}}_{t,\tau-2}^{-1} \left(\hat{\mathbf{Z}}_{t-2}\hat{\mathbf{Z}}_{t-1}\right) + \hat{\rho}_{t-2} \left(\hat{\mathbf{Z}}_{t-1}^T\right) \mathbf{v}_{t-2}\, \mathbf{v}_{t-2}^T \left(\hat{\mathbf{Z}}_{t-1}\right) + \hat{\rho}_{t-1}\, \mathbf{v}_{t-1}\, \mathbf{v}_{t-1}^T. \tag{15}$$

We can proceed recursively by substituting $\hat{\mathbf{B}}_{t,\tau-2}^{-1}$ for its expression in terms of $\hat{\mathbf{B}}_{t,\tau-3}^{-1}$ and in the result substitute $\hat{\mathbf{B}}_{t,\tau-3}^{-1}$ for its expression in terms of $\hat{\mathbf{B}}_{t,\tau-3}^{-1}$ and so on. Observe that a new summand

is added in each of these substitutions from which it follows that repeating this process τ times yields

$$\hat{\mathbf{B}}_t^{-1} = \left(\hat{\mathbf{Z}}_{t-1}^T \dots \hat{\mathbf{Z}}_{t-\tau}^T\right) \hat{\mathbf{B}}_{t,0}^{-1} \left(\hat{\mathbf{Z}}_{t-\tau} \dots \hat{\mathbf{Z}}_{t-1}\right) + \hat{\rho}_{t-\tau} \left(\hat{\mathbf{Z}}_{t-1}^T \dots \hat{\mathbf{Z}}_{t-\tau+1}^T\right) \mathbf{v}_{t-\tau}\mathbf{v}_{t-\tau}^T \left(\hat{\mathbf{Z}}_{t-\tau+1} \dots \hat{\mathbf{Z}}_{t-1}\right) + \cdots + \hat{\rho}_{t-2} \left(\hat{\mathbf{Z}}_{t-1}^T\right) \mathbf{v}_{t-2}\mathbf{v}_{t-2}^T \left(\hat{\mathbf{Z}}_{t-1}\right) + \hat{\rho}_{t-1}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^T. \tag{16}$$

The important observation in (16) is that the matrix $\hat{\mathbf{Z}}_{t-1}$ and its transpose $\hat{\mathbf{Z}}_{t-1}^T$ are the first and last product terms of all summands except the last, that the matrices $\hat{\mathbf{Z}}_{t-2}$ and its transpose $\hat{\mathbf{Z}}_{t-2}^T$ are second and penultimate in all terms but the last two, and so on. Thus, when computing the oLBFGS step $\hat{\mathbf{d}}_t := \hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ the operations needed to compute the product with the next to last summand of (16) can be reused to compute the product with the second to last summand which in turn can be reused in determining the product with the third to last summand and so on. This observation compounded with the fact that multiplications with the identity plus rank one matrices $\hat{\mathbf{Z}}_{t-1}$ requires $O(n)$ operations yields an algorithm that can compute the oLBFGS step $\hat{\mathbf{d}}_t := \hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ in $O(\tau n)$ operations. We summarize the specifics of such computation in the following proposition where we consider the computation of the product $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ with a given arbitrary vector $\mathbf{p}$.

Proposition 1 *Consider the oLBFGS Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ obtained after τ recursive applications of the update in* (13) *with the scalar sequence $\hat{\rho}_{t-\tau+u}$ and identity plus rank-one matrix sequence $\hat{\mathbf{Z}}_{t-\tau+u}$ as defined in* (12) *for given variable and stochastic gradient variation pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=t-\tau}^{t-1}$. For a given vector $\mathbf{p} = \mathbf{p}_0$ define the sequence of vectors $\mathbf{p}_k$ through the recursion*

$$\mathbf{p}_{u+1} = \mathbf{p}_u - \alpha_u \hat{\mathbf{r}}_{t-u-1} \qquad \text{for } u = 0, \dots, \tau - 1, \tag{17}$$

where we also define the constants $\alpha_u := \hat{\rho}_{t-u-1}\mathbf{v}_{t-u-1}^T\mathbf{p}_u$. Further define the sequence of vectors $\mathbf{q}_k$ with initial value $\mathbf{q}_0 = \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$ and subsequent elements

$$\mathbf{q}_{u+1} = \mathbf{q}_u + (\alpha_{\tau-u-1} - \beta_u)\mathbf{v}_{t-\tau+u} \quad \text{for } u = 0, \dots, \tau - 1, \tag{18}$$

where we define constants $\beta_u := \hat{\rho}_{t-\tau+u}\hat{\mathbf{r}}_{t-\tau+u}^T\mathbf{q}_u$. The product $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ equals $\mathbf{q}_\tau$, i.e., $\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \mathbf{q}_\tau$.

Proof See Appendix A. ■

Proposition 1 asserts that it is possible to reduce the computation of the product $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ between the oLBFGS Hessian approximation matrix and arbitrary vector $\mathbf{p}$ to the computation of two vector sequences $\{\mathbf{p}_u\}_{u=0}^{\tau-1}$ and $\{\mathbf{q}_u\}_{u=0}^{\tau-1}$. The product $\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \mathbf{q}_\tau$ is given by the last element of the latter sequence. Since determination of each of the elements of each sequence requires $O(n)$ operations and the total number of elements in each sequence is τ the total operation cost to compute both sequences is of order $O(\tau n)$. In computing $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ we also need to add the cost of the product $\mathbf{q}_0 = \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$ that links both sequences. To maintain overall computation cost of order $O(\tau n)$ this matrix has to have a sparse or low rank structure. A common choice in LBFGS, that we adopt for oLBFGS, is to make $\hat{\mathbf{B}}_{t,0}^{-1} = \hat{\gamma}_t\mathbf{I}$. The scalar constant $\hat{\gamma}_t$ is a function of the variable and stochastic gradient variations $\mathbf{v}_{t-1}$ and $\hat{\mathbf{r}}_{t-1}$, explicitly given by

$$\hat{\gamma}_t = \frac{\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}}{\hat{\mathbf{r}}_{t-1}^T\hat{\mathbf{r}}_{t-1}} = \frac{\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}}{\|\hat{\mathbf{r}}_{t-1}\|^2}. \tag{19}$$

with the value at the first iteration being $\hat{\gamma}_0 = 1$. The scaling factor $\hat{\gamma}_t$ attempts to estimate one of the eigenvalues of the Hessian matrix at step t and has been observed to work well in practice; see e.g., Dong C. and Nocedal (1989); Nocedal and Wright (1999). Further observe that the cost of computing $\hat{\gamma}_t$ is of order $O(n)$ and that since $\hat{\mathbf{B}}_{t,0}^{-1}$ is diagonal cost of computing the product

Algorithm 1 Computation of oLBFGS step $\mathbf{q} = \hat{\mathbf{B}}_t^{-1}\mathbf{p}$ when called with $\mathbf{p} = \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$.

1: **function** $\mathbf{q} = \mathbf{q}_\tau$ = oLBFGS Step$\Big(\hat{\mathbf{B}}_{t,0}^{-1},\ \mathbf{p} = \mathbf{p}_0,\ \{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=t-\tau}^{t-1}\Big)$
2: **for** $u = 0, 1, \ldots, \tau - 1$ **do** {Loop to compute constants α_u and sequence $\mathbf{p}_u$}
3: Compute and store scalar $\alpha_u = \hat{\rho}_{t-u-1}\mathbf{v}_{t-u-1}^T\mathbf{p}_u$
4: Update sequence vector $\mathbf{p}_{u+1} = \mathbf{p}_u - \alpha_u\hat{\mathbf{r}}_{t-u-1}$. [cf. (17)]
5: **end for**
6: Multiply $\mathbf{p}_\tau$ by initial matrix: $\mathbf{q}_0 = \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$
7: **for** $u = 0, 1, \ldots, \tau - 1$ **do** {Loop to compute constants β_u and sequence $\mathbf{q}_u$}
8: Compute scalar $\beta_u = \hat{\rho}_{t-\tau+u}\hat{\mathbf{r}}_{t-\tau+u}^T\mathbf{q}_u$
9: Update sequence vector $\mathbf{q}_{u+1} = \mathbf{q}_u + (\alpha_{\tau-u-1} - \beta_u)\mathbf{v}_{t-\tau+u}$ [cf. (18)]
10: **end for** {return $\mathbf{q} = \mathbf{q}_\tau$}

$\mathbf{q}_0 = \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$ is also of order $O(n)$. We adopt the initialization in (19) in our subsequent analysis and numerical experiments.

The computation of the product $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ using the result in Proposition 1 is summarized in algorithmic form in the function in Algorithm 1. The function receives as arguments the initial matrix $\hat{\mathbf{B}}_{t,0}^{-1}$, the sequence of variable and stochastic gradient variations $\{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=t-\tau}^{t-1}$ and the vector $\mathbf{p}$ to produce the outcome $\mathbf{q} = \mathbf{q}_\tau = \hat{\mathbf{B}}_t^{-1}\mathbf{p}$. When called with the stochastic gradient $\mathbf{p} = \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$, the function outputs the oLBFGS step $\hat{\mathbf{d}}_t := \hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ needed to implement the oLBFGS descent step in (10). The core of Algorithm 1 is given by the loop in steps 2-5 that computes the constants α_u and sequence elements $\mathbf{p}_u$ as well as the loop in steps 7-10 that computes the constants β_u and sequence elements $\mathbf{q}_u$. The two loops are linked by the initialization of the second sequence with the outcome of the first which is performed in Step 6. To implement the first loop we require τ inner products in Step 4 and τ vector summations in Step 5 which yield a total of $2\tau n$ multiplications. Likewise, the second loop requires τ inner products and τ vector summations in steps 9 and 10, respectively, which yields a total cost of also $2\tau n$ multiplications. Since the initial Hessian inverse approximation matrix $\hat{\mathbf{B}}_{t,0}^{-1}$ is diagonal the cost of computation $\hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$ in Step 6 is n multiplications. Thus, Algorithm 1 requires a total of $(4\tau + 1)n$ multiplications which affirms the complexity cost of order $O(\tau n)$ for oLBFGS.

For reference, oLBFGS is also summarized in algorithmic form in Algorithm 2. As with any stochastic descent algorithm the descent iteration is implemented in three steps: the acquisition of L samples in Step 2, the computation of the stochastic gradient in Step 3, and the implementation of the descent update on the variable $\mathbf{w}_t$ in Step 6. Steps 4 and 5 are devoted to the computation of the oLBFGS descent direction $\hat{\mathbf{d}}_t$. In Step 4 we initialize the estimate $\hat{\mathbf{B}}_{t,0} = \hat{\gamma}_t\mathbf{I}$ as a scaled identity matrix using the expression for $\hat{\gamma}_t$ in (19) for $t > 0$. The value of $\gamma_t = \gamma_0$ for $t = 0$ is left as an input for the algorithm. We use $\hat{\gamma}_0 = 1$ in our numerical tests. In Step 5 we use Algorithm 1 for efficient computation of the descent direction $\hat{\mathbf{d}}_t = \hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. Step 7 determines the value of the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t)$ so that the variable variations $\mathbf{v}_t$ and stochastic gradient variations $\hat{\mathbf{r}}_t$ become available for the computation of the curvature approximation matrix $\hat{\mathbf{B}}_t^{-1}$. In Step 8 the variable variation $\mathbf{v}_t$ and stochastic gradient variation $\hat{\mathbf{r}}_t$ are computed to be used in the next iteration. We analyze convergence properties of this algorithm in Section 3, study its application to SVMs in Section 4, and develop an application to search engine advertisement in Section 5.

Algorithm 2 oLBFGS

Require: Initial value $\mathbf{w}_0$. Initial Hessian approximation parameter $\hat{\gamma}_0 = 1$.

1: **for** $t = 0, 1, 2, \ldots$ **do**

2: Acquire L independent samples $\tilde{\boldsymbol{\theta}}_t = [\boldsymbol{\theta}_{t1}, \ldots, \boldsymbol{\theta}_{tL}]$

3: Compute stochastic gradient: $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) = \dfrac{1}{L}\sum_{l=1}^{L} \nabla_{\mathbf{w}} f(\mathbf{w}_t, \boldsymbol{\theta}_{tl})$ [cf. (9)]

4: Initialize Hessian inverse estimate as $\hat{\mathbf{B}}_{t,0}^{-1} = \hat{\gamma}_t \mathbf{I}$ with $\hat{\gamma}_t = \dfrac{\mathbf{v}_{t-1}^T \hat{\mathbf{r}}_{t-1}}{\hat{\mathbf{r}}_{t-1}^T \hat{\mathbf{r}}_{t-1}}$ for $t > 0$ [cf (19)]

5: Compute descent direction with Algorithm 1: $\hat{\mathbf{d}}_t = \text{oLBFGS Step}\left(\hat{\mathbf{B}}_{t,0}^{-1},\ \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t),\ \{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=t-\tau}^{t-1}\right)$

6: Descend along direction $\hat{\mathbf{d}}_t$: $\mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon_t \hat{\mathbf{d}}_t$ [cf. (10)]

7: Compute stochastic gradient: $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) = \dfrac{1}{L}\sum_{l=1}^{L} \nabla_{\mathbf{w}} f(\mathbf{w}_{t+1}, \boldsymbol{\theta}_{tl})$ [cf. (9)]

8: Variations $\mathbf{v}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$ [variable, cf. (4)] $\hat{\mathbf{r}}_t = \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ [stoch. gradient, cf.(11)]

9: **end for**

3. Convergence analysis

For the subsequent analysis it is convenient to define the instantaneous objective function associated with samples $\tilde{\boldsymbol{\theta}} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L]$ as

$$\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) := \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{w}, \boldsymbol{\theta}_l). \tag{20}$$

The definition of the instantaneous objective function $\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ in association with the fact that $F(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$ implies that

$$F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})]. \tag{21}$$

Our goal here is to show that as time progresses the sequence of variable iterates $\mathbf{w}_t$ approaches the optimal argument $\mathbf{w}^*$. In proving this result we make the following assumptions.

Assumption 1 The instantaneous functions $\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ are twice differentiable and the eigenvalues of the instantaneous Hessian $\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) = \nabla_{\mathbf{w}}^2 \hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ are bounded between constants $0 < \tilde{m}$ and $\tilde{M} < \infty$ for all random variables $\tilde{\boldsymbol{\theta}}$,

$$\tilde{m}\mathbf{I} \ \preceq\ \hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) \ \preceq\ \tilde{M}\mathbf{I}. \tag{22}$$

Assumption 2 The second moment of the norm of the stochastic gradient is bounded for all $\mathbf{w}$. i.e., there exists a constant S^2 such that for all variables $\mathbf{w}$ it holds

$$\mathbb{E}_{\boldsymbol{\theta}}\left[\|\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\|^2 \,\middle|\, \mathbf{w}_t\right] \leq S^2. \tag{23}$$

Assumption 3 The step size sequence is selected as nonsummable but square summable, i.e.,

$$\sum_{t=0}^{\infty} \epsilon_t = \infty, \quad \text{and} \quad \sum_{t=0}^{\infty} \epsilon_t^2 < \infty. \tag{24}$$

Assumptions 2 and 3 are customary in stochastic optimization. The restriction imposed by Assumption 2 is intended to limit the random variation of stochastic gradients. If the variance of their norm is unbounded it is possible to have rare events that derail progress towards convergence. The condition in Assumption 3 balances descent towards optimal arguments – which requires a slowly decreasing stepsize – with the eventual elimination of random variations – which requires

rapidly decreasing stepsizes. An effective step size choice for which Assumption 3 holds is to make $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$, for given parameters ϵ_0 and T_0 that control the initial step size and its speed of decrease, respectively. Assumption 1 is stronger than usual and specific to oLBFGS. Observe that considering the linearity of the expectation operator and the expression in (21) it follows that the Hessian of the average function can be written as $\nabla^2_{\mathbf{w}} F(\mathbf{w}) = \mathbf{H}(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})]$. Combining this observation with the bounds in (22) we conclude that there are constants $m \geq \tilde{m}$ and $M \leq \tilde{M}$ such that

$$\tilde{m}\mathbf{I} \preceq m\mathbf{I} \preceq \mathbf{H}(\mathbf{w}) \preceq M\mathbf{I} \preceq \tilde{M}\mathbf{I}. \tag{25}$$

The bounds in (25) are customary in convergence proofs of descent methods. For the results here the stronger condition spelled in Assumption 1 is needed. This assumption in necessary to guarantee that the inner product $\hat{\mathbf{r}}_t^T \mathbf{v}_t > 0$ is positive as we show in the following lemma.

Lemma 2 *Consider the stochastic gradient variation $\hat{\mathbf{r}}_t$ defined in* (11) *and the variable variation $\mathbf{v}_t$ defined in* (4)*. Let Assumption 1 hold so that we have lower and upper bounds $\tilde{m}$ and $\tilde{M}$ on the eigenvalues of the instantaneous Hessians. Then, for all steps t the inner product of variable and stochastic gradient variations $\hat{\mathbf{r}}_t^T \mathbf{v}_t$ is bounded below as*

$$\tilde{m}\|\mathbf{v}_t\|^2 \leq \hat{\mathbf{r}}_t^T \mathbf{v}_t \ . \tag{26}$$

Furthermore, the ratio of stochastic gradient variation squared norm $\|\hat{\mathbf{r}}_t\|^2 = \hat{\mathbf{r}}_t^T \hat{\mathbf{r}}_t$ to inner product of variable and stochastic gradient variations is bounded as

$$\tilde{m} \ \leq \ \frac{\hat{\mathbf{r}}_t^T \hat{\mathbf{r}}_t}{\hat{\mathbf{r}}_t^T \mathbf{v}_t} \ = \ \frac{\|\hat{\mathbf{r}}_t\|^2}{\hat{\mathbf{r}}_t^T \mathbf{v}_t} \ \leq \ \tilde{M}. \tag{27}$$

Proof See Appendix B. ■

According to Lemma 2, strong convexity of instantaneous functions $\hat{f}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ guaranties positiveness of the inner product $\mathbf{v}_t^T \hat{\mathbf{r}}_t$ as long as the variable variation is not identically null. In turn, this implies that the constant $\hat{\gamma}_t$ in (19) is nonnegative and that, as a consequence, the initial Hessian inverse approximation $\hat{\mathbf{B}}_{t,0}^{-1}$ is positive definite for all steps t. The positive definiteness of $\hat{\mathbf{B}}_{t,0}^{-1}$ in association with the positiveness of the inner product of variable and stochastic gradient variations $\mathbf{v}_t^T \hat{\mathbf{r}}_t > 0$ further guarantees that all the matrices $\hat{\mathbf{B}}_{t,u+1}^{-1}$, including the matrix $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ in particular, that follow the update rule in (13) stay positive definite – see Mokhtari and Ribeiro (2014a) for details. This proves that (10) is a proper stochastic descent iteration because the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is moderated by a positive definite matrix. However, this fact alone is not enough to guarantee convergence because the minimum and maximum eigenvalues of $\hat{\mathbf{B}}_t^{-1}$ could become arbitrarily small and arbitrarily large, respectively. To prove convergence we show this is not possible by deriving explicit lower and upper bounds on these eigenvalues.

The analysis is easier if we consider the matrix $\hat{\mathbf{B}}_t$ – as opposed to $\hat{\mathbf{B}}_t^{-1}$. Consider then the update in (13), and use the Sherman-Morrison formula to rewrite as an update that relates $\hat{\mathbf{B}}_{t,u+1}$ to $\hat{\mathbf{B}}_{t,u}$,

$$\hat{\mathbf{B}}_{t,u+1} \ = \ \hat{\mathbf{B}}_{t,u} - \frac{\hat{\mathbf{B}}_{t,u}\mathbf{v}_{t-\tau+u}\mathbf{v}_{t-\tau+u}^T\hat{\mathbf{B}}_{t,u}}{\mathbf{v}_{t-\tau+u}^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_{t-\tau+u}} + \frac{\hat{\mathbf{r}}_{t-\tau+u}\hat{\mathbf{r}}_{t-\tau+u}^T}{\mathbf{v}_{t-\tau+u}^T\hat{\mathbf{r}}_{t-\tau+u}}, \tag{28}$$

for $u = 0, \ldots, \tau - 1$ and $\hat{\mathbf{B}}_{t,0} = 1/\hat{\gamma}_t \mathbf{I}$ as per (19). As in (13), the Hessian approximation at step t is $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$. In the following lemma we use the update formula in (28) to find bounds on the trace and determinant of the Hessian approximation $\hat{\mathbf{B}}_t$.

Lemma 3 *Consider the Hessian approximation* $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$ *defined by the recursion in* (28) *with* $\hat{\mathbf{B}}_{t,0} = \hat{\gamma}_t^{-1}\mathbf{I}$ *and* $\hat{\gamma}_t$ *as given by* (19)*. If Assumption 1 holds true, the trace* $\mathrm{tr}(\hat{\mathbf{B}}_t)$ *of the Hessian approximation* $\hat{\mathbf{B}}_t$ *is uniformly upper bounded for all times* $t \geq 1$*,*

$$\mathrm{tr}\left(\hat{\mathbf{B}}_t\right) \;\leq\; (n+\tau)\tilde{M}. \tag{29}$$

Likewise, if Assumption 1 holds true, the determinant $\det(\hat{\mathbf{B}}_t)$ *of the Hessian approximation* $\hat{\mathbf{B}}_t$ *is uniformly lower bounded for all times* t

$$\det\left(\hat{\mathbf{B}}_t\right) \;\geq\; \frac{\tilde{m}^{n+\tau}}{[(n+\tau)\tilde{M}]^{\tau}}\;. \tag{30}$$

Proof See Appendix C. ■

Lemma 3 states that the trace and determinants of the Hessian approximation matrix $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$ are bounded for all times $t \geq 1$. For time $t = 0$ we can write a similar bound that takes into account the fact that the constant γ_t that initializes the recursion in (28) is $\gamma_0 = 1$. Given that we are interested in an asymptotic convergence analysis, this bound in inconsequential. The bounds on the trace and determinant of $\hat{\mathbf{B}}_t$ are respectivey equivalent to bounds in the sum and product of its eigenvalues. Further considering that the matrix $\hat{\mathbf{B}}_t$ is positive definite, as it follows from Lemma 2, these bounds can be further transformed into bounds on the smalls and largest eigenvalue of $\hat{\mathbf{B}}_t$. The resulting bounds are formally stated in the following lemma.

Lemma 4 *Consider the Hessian approximation* $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$ *defined by the recursion in* (28) *with* $\hat{\mathbf{B}}_{t,0} = \hat{\gamma}_t^{-1}\mathbf{I}$ *and* $\hat{\gamma}_t$ *as given by* (19)*. Define the strictly positive constant* $0 < c := \tilde{m}^{n+\tau}/[(n+\tau)\tilde{M}]^{n+\tau-1}$ *and the finite constant* $C := (n+\tau)\tilde{M} < \infty$*. If Assumption 1 holds true, the range of eigenvalues of* $\hat{\mathbf{B}}_t$ *is bounded by* c *and* C *for all time steps* $t \geq 1$*, i.e.,*

$$\frac{\tilde{m}^{n+\tau}}{\left[(n+\tau)\tilde{M}\right]^{n+\tau-1}}\mathbf{I} \;=:\; c\mathbf{I} \;\preceq\; \hat{\mathbf{B}}_t \;\preceq\; C\mathbf{I} \;:=\; (n+\tau)\tilde{M}\,\mathbf{I}. \tag{31}$$

Proof See Appendix D. ■

The bounds in Lemma 4 imply that their respective inverses are bounds on the range of the eigenvalues of the Hessian inverse approximation matrix $\hat{\mathbf{B}}_t^{-1}$. Specifically, the minimum eigenvalue of the Hessian inverse approximation $\hat{\mathbf{B}}_t^{-1}$ is larger than $1/C$ and the maximum eigenvalue of $\hat{\mathbf{B}}_t^{-1}$ does not exceed $1/c$, or, equivalently,

$$\frac{1}{C}\mathbf{I} \;\preceq\; \hat{\mathbf{B}}_t^{-1} \;\preceq\; \frac{1}{c}\mathbf{I}. \tag{32}$$

We further emphasize that the bounds in (32), or (31) for that matter, limit the conditioning of $\hat{\mathbf{B}}_t^{-1}$ for all realizations of the random samples $\{\tilde{\boldsymbol{\theta}}_t\}_{t=0}^{\infty}$, irrespective of the particular random draw. Having matrices $\hat{\mathbf{B}}_t^{-1}$ that are strictly positive definite with eigenvalues uniformly upper bounded by $1/c$ leads to the conclusion that if $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is a descent direction, the same holds true of $\hat{\mathbf{B}}_t^{-1}\,\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. The stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is not a descent direction in general, but we know that this is true for its conditional expectation $\mathbb{E}[\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\,|\,\mathbf{w}_t] = \nabla F(\mathbf{w}_t)$. Hence, we conclude that $\hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$ is an average descent direction since $\mathbb{E}[\hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)\big|\mathbf{w}_t] = \hat{\mathbf{B}}_t^{-1}\nabla F(\mathbf{w}_t)$. Stochastic optimization methods whose displacements $\mathbf{w}_{t+1} - \mathbf{w}_t$ are descent directions on average are expected to approach optimal arguments. We show that this is true of oLBFGS in the following lemma.

Lemma 5 *Consider the online Limited memory BFGS algorithm as defined by the descent iteration in* (10) *with matrices* $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ *obtained after* τ *recursive applications of the update in* (13) *initialized with* $\hat{\mathbf{B}}_{t,0}^{-1} = \hat{\gamma}_t \mathbf{I}$ *and* $\hat{\gamma}_t$ *as given by* (19). *If Assumptions 1 and 2 hold true, the sequence of average function values* $F(\mathbf{w}_t)$ *satisfies*

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] \leq F(\mathbf{w}_t) - \frac{\epsilon_t}{C}\|\nabla F(\mathbf{w}_t)\|^2 + \frac{MS^2\epsilon_t^2}{2c^2}. \tag{33}$$

Proof See Appendix E. ■

Setting aside the term $MS^2\epsilon_t^2/2c^2$ for the sake of argument, (33) defines a supermartingale relationship for the sequence of average functions $F(\mathbf{w}_t)$. This implies that the sequence $\epsilon_t\|\nabla F(\mathbf{w}_t)\|^2/C$ is almost surely summable which, given that the step sizes ϵ_t are nonsummable as per (24), further implies that the limit infimum $\liminf_{t\to\infty}\|\nabla F(\mathbf{w}_t)\|$ of the gradient norm $\|\nabla F(\mathbf{w}_t)\|$ is almost surely null. This latter observation is equivalent to having $\liminf_{t\to\infty}\|\mathbf{w}_t - \mathbf{w}^*\|^2 = 0$ with probability 1 over realizations of the random samples $\{\tilde{\boldsymbol{\theta}}_t\}_{t=0}^{\infty}$. The term $MS^2\epsilon_t^2/2c^2$ is a relatively minor nuisance that can be taken care of with a technical argument that we present in the proof of the following theorem.

Theorem 6 *Consider the online Limited memory BFGS algorithm as defined by the descent iteration in* (10) *with matrices* $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ *obtained after* τ *recursive applications of the update in* (13) *initialized with* $\hat{\mathbf{B}}_{t,0}^{-1} = \hat{\gamma}_t \mathbf{I}$ *and* $\hat{\gamma}_t$ *as given by* (19). *If Assumptions 1-3 hold true the limit infimum of the squared Euclidean distance to optimality* $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ *converges to zero almost surely, i.e.,*

$$\Pr\left[\liminf_{t\to\infty}\|\mathbf{w}_t - \mathbf{w}^*\|^2 = 0\right] = 1, \tag{34}$$

where the probability is over realizations of the random samples $\{\tilde{\boldsymbol{\theta}}_t\}_{t=0}^{\infty}$.

Proof See Appendix F. ■

Theorem 6 establishes convergence of a subsequence of the oLBFGS algorithm summarized in Algorithm 2. The lower and upper bounds on the eigenvalues of $\hat{\mathbf{B}}_t$ derived in Lemma 4 play a fundamental role in the proofs of the prerequisite Lemma 5 and Theorem 6 proper. Roughly speaking, the lower bound on the eigenvalues of $\hat{\mathbf{B}}_t$ results in an upper bound on the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$ which limits the effect of random variations on the stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$. If this bound does not exist – as is the case, e.g., of regular stochastic BFGS – we may observe catastrophic amplification of random variations of the stochastic gradient. The upper bound on the eigenvalues of $\hat{\mathbf{B}}_t$, which results in a lower bound on the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$, guarantees that the random variations in the curvature estimate $\hat{\mathbf{B}}_t$ do not yield matrices with arbitrarily small norm. If this bound does not hold, it is possible to end up halting progress before convergence as the stochastic gradient is nullified by multiplication with an arbitrarily small eigenvalue.

The result in Theorem 6 is strong because it holds almost surely over realizations of the random samples $\{\tilde{\boldsymbol{\theta}}_t\}_{t=0}^{\infty}$ but not stronger than the same convergence guarantees that hold for SGD. We complement the convergence result in Theorem 6 with a characterization of the expected convergence rate that we introduce in the following theorem.

Theorem 7 *Consider the online Limited memory BFGS algorithm as defined by the descent iteration in* (10) *with matrices* $\hat{\mathbf{B}}_t^{-1} = \hat{\mathbf{B}}_{t,\tau}^{-1}$ *obtained after* τ *recursive applications of the update in* (13) *initialized with* $\hat{\mathbf{B}}_{t,0}^{-1} = \hat{\gamma}_t \mathbf{I}$ *and* $\hat{\gamma}_t$ *as given by* (19). *Let Assumptions 1 and 2 hold, and further assume that the stepsize sequence is of the form* $\epsilon_t = \epsilon_0/(t + T_0)$ *with the parameters* ϵ_0 *and* T_0

satisfying the inequality $2m\epsilon_0 T_0/C > 1$. *Then, the difference between the expected optimal objective* $\mathbb{E}\left[F(\mathbf{w}_t)\right]$ *and the optimal objective* $F(\mathbf{w}^*)$ *is bounded as*

$$\mathbb{E}\left[F(\mathbf{w}_t)\right] - F(\mathbf{w}^*) \;\leq\; \frac{C_0}{T_0 + t}\;, \tag{35}$$

where the constant C_0 *is defined as*

$$C_0 := \max\left\{\frac{\epsilon_0^2\; T_0^2 CMS^2}{2c^2(2m\epsilon_0 T_0 - C)}\;,\; T_0\left(F(\mathbf{w}_0) - \;F(\mathbf{w}^*)\right)\right\}. \tag{36}$$

Proof See Appendix G. ■

Theorem 7 shows that under specified assumptions the expected error in terms of the objective value after t oLBFGS iterations is of order $O(1/t)$. As is the case of Theorem 6, this result is not better than the convergence rate of conventional SGD. As can be seen in the proof of Theorem 7, the convergence rate is dominated by the noise term introduced by the difference between stochastic and regular gradients. This noise term would be present even if exact Hessians were available and in that sense the best that can be proven of oLBFGS is that the convergence rate is not worse than that of SGD. Given that theorems 6 and 7 parallel the theoretical guarantees of SGD it is perhaps fairer to describe oLBFGS as an adaptive reconditioning strategy instead of a stochastic quasi-Newton method. The latter description refers to the genesis of the algorithm, but the former is more accurate description of its behavior. Do notice that while the convergence rate doesn't change, improvements in convergence time are significant as we illustrate with the numerical experiments that we present in the next two sections.

4. Support vector machines

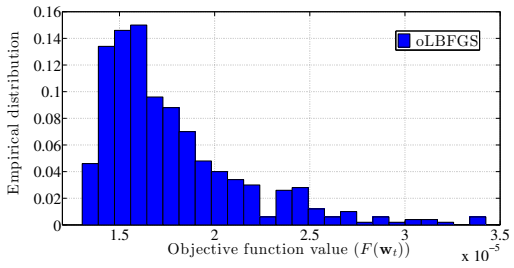
Given a training set with points whose classes are known the goal of an SVM is to find a hyperplane that best separates the training set. Let $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be a training set containing N pairs of the form $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbb{R}^n$ is a feature vector and $y_i \in \{-1, 1\}$ is the corresponding class. The goal is to find a hyperplane supported by a vector $\mathbf{w} \in \mathbb{R}^n$ which separates the training set so that $\mathbf{w}^T\mathbf{x}_i > 0$ for all points with $y_i = 1$ and $\mathbf{w}^T\mathbf{x}_i < 0$ for all points with $y_i = -1$. A loss function $l((\mathbf{x}, y); \mathbf{w})$ defines a measure of distance between the point $\mathbf{x}_i$ and the hyperplane supported by $\mathbf{w}$. We then select the hyperplane supporting vector as

$$\mathbf{w}^* := \underset{\mathbf{w}}{\operatorname{argmin}}\;\; \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} l((\mathbf{x}_i, y_i); \mathbf{w}), \tag{37}$$

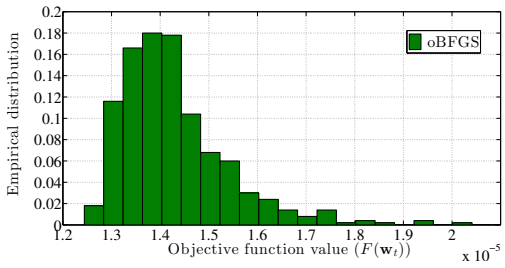
where we have also added the regularization term $\lambda\|\mathbf{w}\|^2/2$ for some constant $\lambda > 0$. Common selections for the loss function are the hinge loss $l((\mathbf{x}, y); \mathbf{w}) = \max(0, 1 - y(\mathbf{w}^T\mathbf{x}))$ and the squared hinge loss $l((\mathbf{x}, y); \mathbf{w}) = \max(0, 1 - y(\mathbf{w}^T\mathbf{x}))^2$. See, e.g., Bottou (2010). To model (37) as a problem in the form of (1), define $\boldsymbol{\theta}_i = (\mathbf{x}_i, y_i)$ as a given training point and the probability distribution of θ as uniform on the training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N = \{\boldsymbol{\theta}_i\}_{i=1}^N$. It then suffices to define

$$f(\mathbf{w}, \boldsymbol{\theta}) = \;f(\mathbf{w}, (\mathbf{x}, y)) \;:=\; \frac{\lambda}{2}\|\mathbf{w}\|^2 + l((\mathbf{x}, y); \mathbf{w}), \tag{38}$$

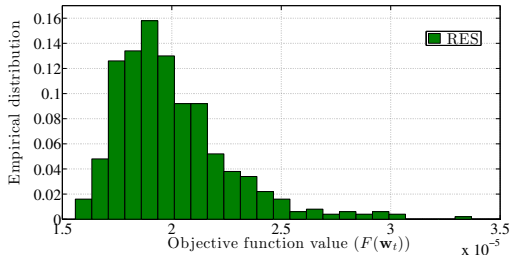
as sample functions to see that the objective in (37) can be written as the average $F(\mathbf{w}) = \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w}, \boldsymbol{\theta})]$ as in (1). We can then use SGD, oBFGS, RES, and oLBFGS to find the optimal classifier $\mathbf{w}^*$. There are also several algorithms that accelerate SGD through the use of memory. These algorithms reduce execution times because they reduce randomness, not because they improve curvature, but



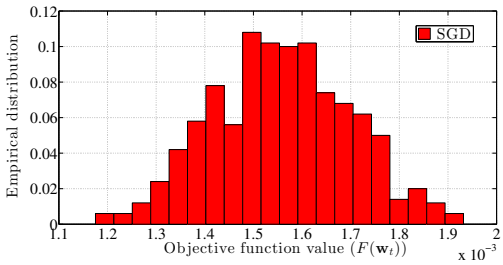
(a) oLBFGS



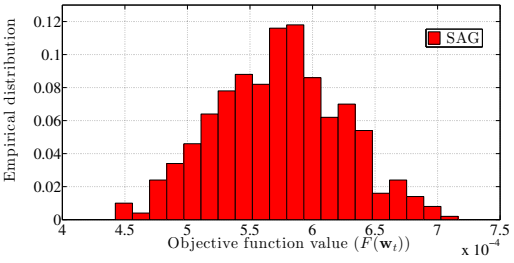
(b) oBFGS



(c) RES



(d) SGD



(e) SAG

	Objective function value		
Algorithm	Minimum	Average	Maximum
oLBFGS	1.3×10^{-5}	1.7×10^{-5}	3.4×10^{-5}
oBFGS	1.2×10^{-5}	1.4×10^{-5}	2.0×10^{-5}
RES	1.5×10^{-5}	1.9×10^{-5}	3.3×10^{-5}
SGD	1.2×10^{-3}	1.6×10^{-3}	1.9×10^{-3}
SAG	4.4×10^{-4}	5.7×10^{-4}	7.1×10^{-4}

(f) Statistics

Figure 1: Histograms of objective function value $F(\mathbf{w}_t)$ after processing $Lt = 4 \times 10^4$ feature vectors for $n = 10^2$. The values of objective function for oLBFGS, oBFGS and RES are close to each other and smaller than the objective function values for SAG and SGD.

are nonetheless alternatives to oLBFGS. We further add Stochastic Average Gradient (SAG) to the comparison set. SAG is a variant of SGD that uses an average of stochastic gradients as a descent direction (Schmidt et al. (2013)). The performances of other SGD algorithms with memory are similar to SAG. For these five algorithms we want to compare achieved objective values with respect to the number of feature vectors processed (Section 4.1) as well as with respect to processing times (Section 4.2).

4.1 Convergence versus number of feature vectors processed

For numerical tests we use the squared hinge loss $l((\mathbf{x}, y); \mathbf{w}) = \max(0, 1 - y(\mathbf{x}^T\mathbf{w}))^2$ in (37). The training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ contains $N = 10^4$ feature vectors, half of which belong to the class $y_i = -1$ with the other half belonging to the class $y_i = 1$. For the class $y_i = -1$ each of the n components of each of the feature vectors $\mathbf{x}_i \in \mathbb{R}^n$ is chosen uniformly at random from the

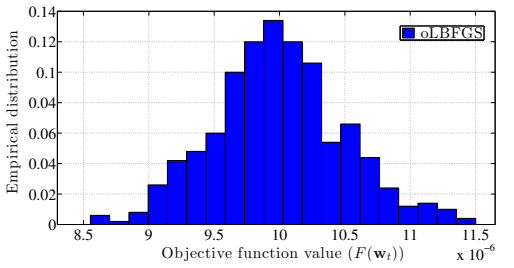
interval $[-0.8, 0.2]$. Likewise, each of the n components of each of the feature vectors $\mathbf{x}_i \in \mathbb{R}^n$ is chosen uniformly at random from the interval $[-0.2, 0.8]$ for the class $y_i = 1$. In all of our numerical experiments the parameter λ in (37) is set to $\lambda = 10^{-4}$. In order to study the advantages of oLBFGS we consider two different cases where the dimensions of the feature vectors are $n = 10^2$ and $n = 10^3$. The size of memory for oLBFGS is set to $\tau = 10$ in both cases. For SGD and SAG the sample size in (9) is $L = 1$ and for RES, oBFGS and oLBFGS is $L = 5$. In all tests, the number of feature vectors processed is represented by the product Lt between the iteration index and the sample size used to compute stochastic gradients. This is done because the sample sizes are different. For all five algorithms we use a decreasing stepsize sequence of the form $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$. We report results for $\epsilon_0 = 2 \times 10^{-2}$ and $T_0 = 10^2$ for RES, oLBFGS and oBFGS, which are the values that yield best average performance after processing 4×10^4 feature vectors. Further improvements can be obtained by tuning stepsize parameters individually for each individual algorithm and feature dimension n. Since these improvements are minor we report common parameters for easier reproducibility. For SGD and SAG, whose performance is more variable, we tune the various parameters individually for each dimension n and report results for the combination that yields best average performance after processing 4×10^4 feature vectors.

Figures 1 and 2 show the empirical distributions of the objective function value $F(\mathbf{w}_t)$ attained after processing $Lt = 4 \times 10^4$ feature vectors using $J = 10^3$ realizations for the cases that $n = 10^2$ and $n = 10^3$, respectively. According to Figure 1 the averages of objective value function for oLBFGS, oBFGS and RES are 1.7×10^{-5}, 1.4×10^{-5} and 1.9×10^{-5}, respectively. These numbers show that the performance of oLBFGS is very close to the performances of oBFGS and RES. This similarity holds despite the fact that oLBFGS uses only the last $\tau = 10$ stochastic gradients to estimate curvature whereas oBFGS and RES utilize all past stochastic gradients to do so. The advantage of oLBFGS is in the smaller computational cost of processing feature vectors as we discuss in Section 4.2. The corresponding average objective values achieved by SGD and SAG after processing $Lt = 4 \times 10^4$ feature vectors are 1.6×10^{-3} and 5.7×10^{-4}, respectively. Both of these are at least an order of magnitude larger than the average objective value achieved by oLBFGS – or RES and oBFGS for that matter.

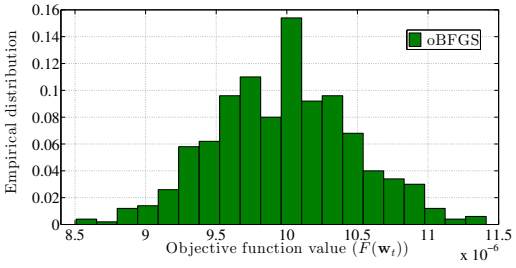
Figure 2 repeats the study in Figure 1 for the case in which the feature vector dimension is increased to $n = 10^3$. The performance of oLBGS is still about the same as the performances of oBFGS and RES. The average objective function values achieved after processing $Lt = 4 \times 10^4$ feature vectors are 9.9×10^{-6}, 9.8×10^{-6} and 9.5×10^{-6} for oLBFGS, oBFGS and RES, respectively. The relative performance with respect to SGD and SAG, however, is now larger. The averages of objective function values for SAG and SGD in this case are 2.1×10^{-2} and 4.5×10^{-2}, respectively. These values are more than 3 orders of magnitude larger than the corresponding values achieved by oLBFGS. This relative improvement can be further increased if we consider problems of even larger dimension. Further observe that oBFGS and RES start to become impractical if we further increase the feature vector dimension since the respective iterations have computational costs of order $O(n^2)$ and $O(n^3)$. We analyze this in detail in the following section.

4.2 Convergence versus processing time

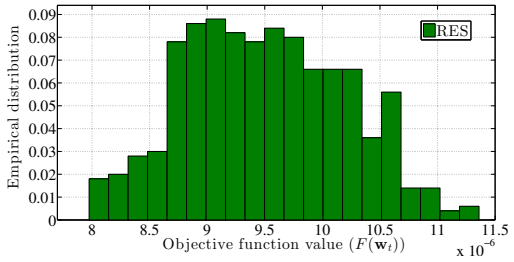
The analysis in Section 4.1 is relevant for online implementations in which the goal is to make the best possible use of the information provided by each new acquired feature vector. In implementations where computational cost is of dominant interest we have to account for the fact that the respective iteration costs are of order $O(n)$ for SGD and SAG, of order $O(\tau n)$ for oLBFGS, and of orders $O(n^2)$ and $O(n^3)$ for oBFGS and RES. As we increase the problem dimension we expect the convergence time advantages of oBFGS and RES in terms of number of feature vectors processed to be overwhelmed by the increased computational cost of each iteration. For oLBFGS, on the contrary, we expect the convergence time advantages in terms of number of feature vectors processed to persist in terms of processing time. To demonstrate that this is the case we repeat the experiments



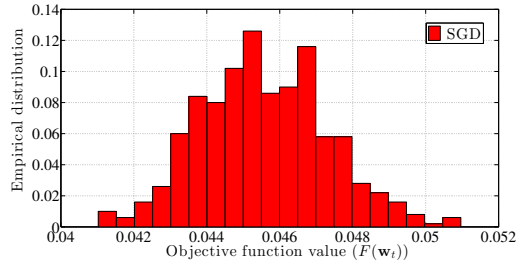
(a) oLBFGS



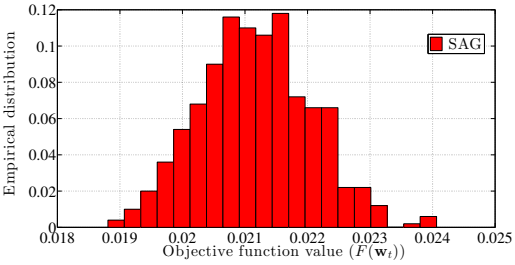
(b) oBFGS



(c) RES



(d) SGD



(e) SAG

	Objective function value		
Algorithm	Minimum	Average	Maximum
oLBFGS	8.6×10^{-6}	9.9×10^{-6}	11.5×10^{-6}
oBFGS	8.5×10^{-6}	9.8×10^{-6}	11.4×10^{-6}
RES	7.9×10^{-6}	9.5×10^{-6}	11.3×10^{-6}
SGD	4.1×10^{-2}	4.5×10^{-2}	5.1×10^{-2}
SAG	1.9×10^{-2}	2.1×10^{-2}	2.4×10^{-2}

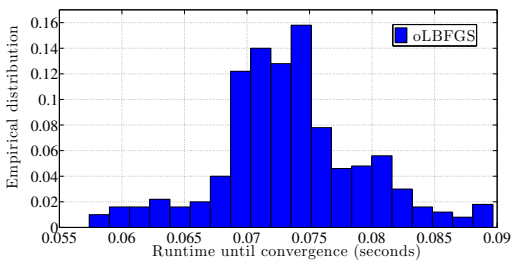
(f) Statistics

Figure 2: Histograms of objective function value $F(\mathbf{w}_t)$ after processing $Lt = 4\times10^4$ feature vectors for $n = 10^3$. The values of objective function for oBFGS, oLBFGS and RES are close to each other and smaller than the objective function values for SAG and SGD.

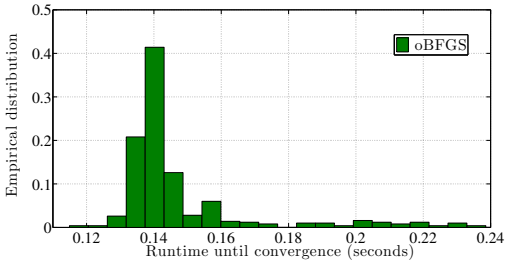
in Section 4.1 but record the processing time required to achieve a target objective value. The parameters used here are the same parameters of Section 4.1.

In Figure 3 we consider $n = 10^2$ and record the processing time required to achieve the objective function value $F(\mathbf{w}_t) = 10^{-4}$. Histograms representing empirical distributions of execution times measured in seconds (s) are shown for oLBFGS, oBFGS, RES, SGD, and SAG. We also summarize the average minimum and maximum times observed for each algorithm. The average run times for oBFGS and RES are 0.14 s and 0.26 s which are better than the average run times of SGD and SAG that stand at 0.63 s and 0.50 s. The advantage, however, is less marked than when measured with respect to the number of feature vector processed. For oLBGS the advantage with respect to SGD and SAG is still close to one order of magnitude since the average convergence time stands at 0.073 s. When measured in computation time oLBGS is also better than RES and oBFGS, as expected.

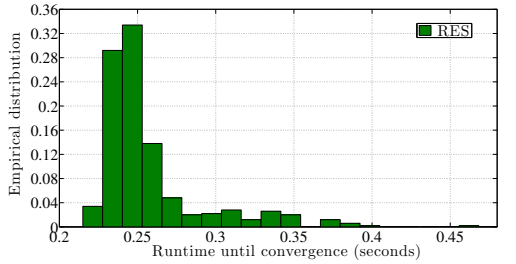
Figure 4 presents the analogous histograms and summary statistics when the feature vector dimension is $n = 10^3$ and the algorithm is run until achieving the objective value $F(\mathbf{w}_t) = 10^{-5}$.



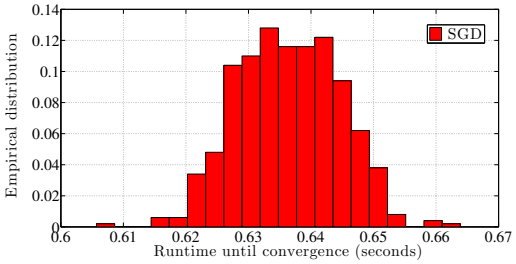
(a) oLBFGS



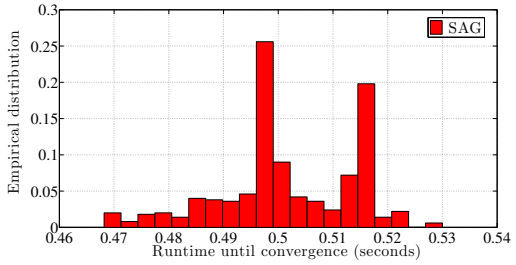
(b) oBFGS



(c) RES



(d) SGD



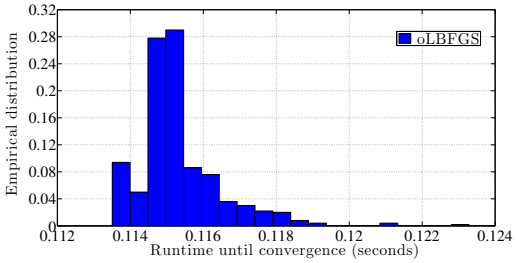
(e) SAG

	CPU runtime (seconds)		
Algorithm	Minimum	Average	Maximum
oLBFGS	0.0574	0.0734	0.0897
oBFGS	0.1149	0.1485	0.2375
RES	0.2150	0.2609	0.4623
SGD	0.6057	0.6364	0.6639
SAG	0.4682	0.5017	0.5300

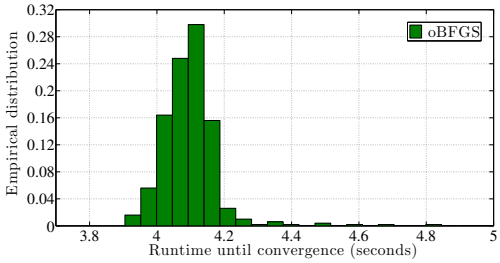
(f) Statistics

Figure 3: Histograms of required CPU runtime for achieving objective function value $F(\mathbf{w}_t) = 10^{-4}$ when $n = 10^2$. The convergence time of oLBFGS is smaller than the required runtimes of oBFGS and RES, while SAG and SGD are slower than all the three quasi-Newton methods.

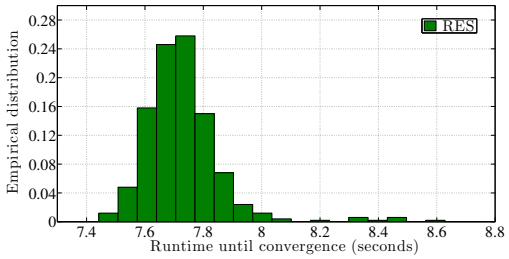
For this problem and metric the performances of RES and oBFGS are worse than the corresponding performances of SGD and SAG. The respective average convergence times are 7.7 s and 4.1 s for RES and oBFGS and 1.4 s and 2.0 s for SAG and SGD. The oLBFGS algorithm, however, has an average convergence time of 0.11 s. This is still an order of magnitude faster than the first order methods SAG and SGD – and has an even larger advantage with respect to oBFGS and RES, by extension. The relative reduction of execution times of oLBGS relative to all other 4 methods becomes more marked for problems of larger dimension. We investigate these advantages on the search engine advertising problem that we introduce in the following section.



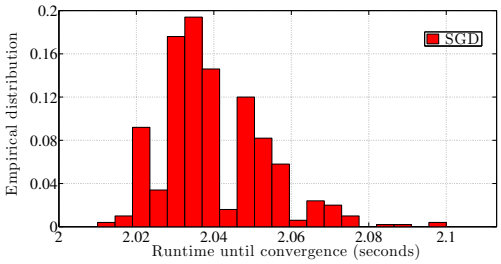
(a) oLBFGS



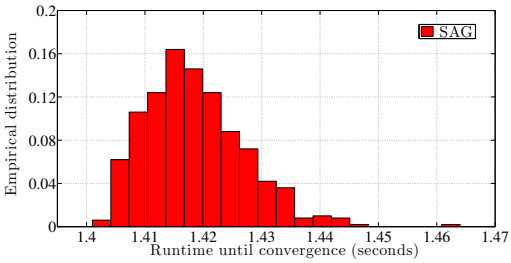
(b) oBFGS



(c) RES



(d) SGD



(e) SAG

	CPU runtime (seconds)		
Algorithm	Minimum	Average	Maximum
oLBFGS	0.1139	0.1153	0.1230
oBFGS	3.90	4.11	4.83
RES	7.44	7.73	8.61
SGD	2.01	2.03	2.10
SAG	1.40	1.42	1.46

(f) Statistics

Figure 4: Histograms of required CPU runtime for achieving objective function value $F(\mathbf{w}_t) = 10^{-5}$ when $n = 10^3$. SAG and SGD have a faster convergence time in comparison to oBFGS and RES, while oLBFGS is the fastest algorithm among all.

5. Search engine advertising

We apply oLBFGS to the problem of predicting the click-through rate (CTR) of an advertisement displayed in response to a specific search engine query by a specific visitor. In these problems we are given meta information about an advertisement, the words that appear in the query, as well as some information about the visitor and are asked to predict the likelihood that this particular ad is clicked by this particular user when performing this particular query. The information specific to the ad includes descriptors of different characteristics such as the words that appear in the title, the name of the advertiser, keywords that identify the product, and the position on the page where the ad is to be displayed. The information specific to the user is also heterogeneous and includes gender, age, and propensity to click on ads. To train a classifier we are given information about past queries along with the corresponding click success of the ads displayed in response to the query. The ad metadata along with user data and search words define a feature vector that we use to train a

Table 1: Components of the feature vector for prediction of advertisements click-through rates. For each feature class we report the total number of components in the feature vector as well as the maximum and average number of nonzero components.

		Nonzero components	
Feature type	**Total components**	**Maximum (observed/structure)**	**Mean (observed)**
Age	6	1 (structure)	1.0
Gender	3	1 (structure)	1.0
Impression	3	1 (structure)	1.0
Depth	3	1 (structure)	1.0
Position	3	1 (structure)	1.0
Query	20,000	125 (observed)	3.0
Title	20,000	29 (observed)	8.8
Keyword	20,000	16 (observed)	2.1
Advertiser ID	5,184	1 (structure)	1.0
Advertisement ID	108,824	1 (structure)	1.0
Total	**174,026**	**148 (observed)**	**20.9**

logistic regressor that predicts the CTR of future ads. Given the heterogeneity of the components of the feature vector we expect a logistic cost function with skewed level sets and consequent large benefits from the use of oLBFGS.

5.1 Feature vectors

For the CTR problem considered here we use the Tencent search engine data set Sun (2012). This data set contains the outcomes of 236 million (236×10^6) searches along with information about the ad, the query, and the user. The information contained in each sample point is the following:

- User profile: If known, age and gender of visitor performing query.
- Depth: Total number of advertisements displayed in the search results page.
- Position: Position of the advertisement in the search page.
- Impression: Number of times the ad was displayed to the user who issued the query.
- Query: The words that appear in the user's query.
- Title: The words that appear in the title of ad.
- Keywords: Selected keywords that specify the type of product.
- Ad ID: Unique identifier assigned to each specific advertisement.
- Advertiser ID: Unique identifier assigned to each specific advertiser.
- Clicks: Number of times the user clicked on the ad.

From this information we create a set of feature vectors $\{\mathbf{x}_i\}_{i=1}^N$, with corresponding labels $y_i \in \{-1, 1\}$. The label associated with feature vector $\mathbf{x}_i$ is $y_i = 1$ if the number of clicks in the ad is more than 0. Otherwise the label is $y_i = -1$. We use a binary encoding for all the features in

the vector $\mathbf{x}_i$. For the age of the user we use the six age intervals $(0, 12]$, $(12, 18]$, $(18, 24]$, $(24, 30]$, $(30, 40]$, and $(40, \infty)$ to construct six indicator entries in $\mathbf{x}_i$ that take the value 1 if the age of the user is known to be in the corresponding interval. E.g., a 21 year old user has an age that falls in the third interval which implies that we make $[\mathbf{x}_i]_3 = 1$ and $[\mathbf{x}_i]_k = 0$ for all other k between 1 and 6. If the age of the user is unknown we make $[\mathbf{x}_i]_k = 0$ for all k between 1 and 6. For the gender of the visitors we use the next three components of $\mathbf{x}_i$ to indicate male, female, or unknown gender. For a male user we make $[\mathbf{x}_i]_7 = 1$, for a female user $[\mathbf{x}_i]_8 = 1$, and for visitors of unknown gender we make $[\mathbf{x}_i]_9 = 1$. The next three components of $\mathbf{x}_i$ are used for the depth feature. If the the number of advertisements displayed in the search page is 1 we make $[\mathbf{x}_i]_{10} = 1$, if 2 different ads are shown we make $[\mathbf{x}_i]_{11} = 1$, and for depths of 3 or more we make $[\mathbf{x}_i]_{12} = 1$. To indicate the position of the ad in the search page we also use three components of $\mathbf{x}_i$. We use $[\mathbf{x}_i]_{13} = 1$, $[\mathbf{x}_i]_{14} = 1$, and $[\mathbf{x}_i]_{15} = 1$ to indicate that the ad is displayed in the first, second, and third position, respectively. Likewise we use $[\mathbf{x}_i]_{16}$, $[\mathbf{x}_i]_{17}$ and $[\mathbf{x}_i]_{18}$ to indicate that the impression of the ad is 1, 2 or more than 3.

For the words that appear in the query we have in the order of 10^5 distinct words. To reduce the number of elements necessary for this encoding we create 20,000 bags of words through random hashing with each bag containing 5 or 6 distinct words. Each of these bags is assigned an index k. For each of the words in the query we find the bag in which this word appears. If the word appears in the kth bag we indicate this occurrence by setting the $k + 18$th component of the feature vector to $[\mathbf{x}_i]_{k+18} = 1$. Observe that since we use 20,000 bags, components 19 through 20,018 of $\mathbf{x}_i$ indicate the presence of specific words in the query. Further note that we may have more than one $\mathbf{x}_i$ component different from zero because there may be many words in the query, but that the total number of nonzero elements is much smaller than 20,000. On average, 3.0 of these elements of the feature vector are nonzero. The same bags of words are used to encode the words that appear in the title of the ad and the product keywords. We encode the words that appear in the title of the ad by using the next $20,000$ components of vector $\mathbf{x}_i$, i.e. components $20,019$ through $40,018$. Components $40,019$ through $60,018$ are used to encode product keywords. As in the case of the words in the search just a few of these components are nonzero. On average, the number of nonzero components of feature vectors that describe the title features is 8.8. For product keywords the average is 2.1. Since the number of distinct advertisers in the training set is $5,184$ we use feature components $60,019$ through 65202 to encode this information. For the kth advertiser ID we set the $k + 60,018th$ component of the feature vector to $[\mathbf{x}_i]_{k+60,018} = 1$. Since the number of distinct advertisements is $108,824$ we allocate the last $108,824$ components of the feature vector to encode the ad ID. Observe that only one out of $5,184$ advertiser ID components and one of the $108,824$ advertisement ID components are nonzero.

In total, the length of the feature vector is 174,026 where each of the components are either 0 or 1. The vector is very sparse. We observe a maximum of 148 nonzero elements and an average of 20.9 nonzero elements in the training set – see Table 5. This is important because the cost of implementing inner products in the oLBFGS training of the logistic regressor that we introduce in the following section is proportional to the number of nonzero elements in $\mathbf{x}_i$.

5.2 Logistic regression of click-through rate

We use the training set to estimate the CTR with a logistic regression as in, e.g., Zhang et al. (2013b). For that purpose let $\mathbf{x} \in \mathbb{R}^n$ be a vector containing the features described in Section 5.1, $\mathbf{w} \in \mathbb{R}^n$ a classifier that we want to train, and $y \in -1, 1$ an indicator variable that takes the value $y = 1$ when the ad presented to the user is clicked and $y = -1$ when the ad is not clicked by the user. We hypothesize that the CTR, defined as the probability of observing $y = 1$, can be written as the logistic function

$$\text{CTR}(\mathbf{x}; \mathbf{w}) \; := \text{P}\left[y = 1 \,\middle|\, \mathbf{x}; \mathbf{w}\right] \; = \; \frac{1}{1 + \exp\left(-\mathbf{x}^T \mathbf{w}\right)} \; . \tag{39}$$

We read (39) as stating that for a feature vector $\mathbf{x}$ the CTR is determined by the inner product $\mathbf{x}^T\mathbf{w}$ through the given logistic transformation.

Consider now the training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ which contains N realizations of features $\mathbf{x}_i$ and respective click outcomes y_i and further define the sets $\mathcal{S}_1 := \{(\mathbf{x}_i, y_i) \in \mathcal{S} : y_i = 1\}$ and $\mathcal{S}_{-1} := \{(\mathbf{x}_i, y_i) \in \mathcal{S} : y_i = -1\}$ containing clicked and unclicked advertisements, respectively. With the data given in $\mathcal{S}$ we define the optimal classifier $\mathbf{w}^*$ as a maximum likelihood estimate (MLE) of $\mathbf{w}$ given the model in (39) and the training set $\mathcal{S}$. This MLE can be found as the minimizer of the log-likelihood loss

$$\begin{aligned}\mathbf{w}^* &:= \operatorname{argmin} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} \log\Big(1 + \exp\big(-y_i\mathbf{x}_i^T\mathbf{w}\big)\Big) \\ &= \operatorname{argmin} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\Bigg[\sum_{\mathbf{x}_i\in\mathcal{S}_1} \log\Big(1 + \exp(-\mathbf{x}_i^T\mathbf{w})\Big) + \sum_{\mathbf{x}_i\in\mathcal{S}_{-1}} \log\Big(1 + \exp(\mathbf{x}_i^T\mathbf{w})\Big)\Bigg], \quad (40)\end{aligned}$$

where we have added the regularization term $\lambda\|\mathbf{w}\|^2/2$ to disincentivize large values in the weight vector $\mathbf{w}^*$; see e.g., Ng (2004).

The practical use of (39) and (40) is as follows. We use the data collected in the training set $\mathcal{S}$ to determine the vector $\mathbf{w}^*$ in (40). When a user issues a query we concatenate the user and query specific elements of the feature vector with the ad specific elements of several candidate ads. We then proceed to display the advertisement with, say, the largest CTR. We can interpret the set $\mathcal{S}$ as having been acquired offline or online. In the former case we want to use a stochastic optimization algorithm because computing gradients is infeasible – recall that we are considering training samples with a number of elements N in the order of 10^6. The performance metric of interest in this case is the logistic cost as a function of computational time. If elements of $\mathcal{S}$ are acquired online we update $\mathbf{w}$ whenever a new vector becomes available so as to adapt to changes in preferences. In this case we want to exploit the information in new samples as much as possible. The correct metric in this case is the logistic cost as a function of the number of feature vectors processed. We use the latter metric for the numerical experiments in the following section.

5.3 Numerical Results

Out of the 236×10^6 in the Tencent dataset we select 10^6 sample points to use as the training set $\mathcal{S}$ and 10^5 sample points to use as a test set $\mathcal{T}$. To select elements of the training and test set we divide the first 1.1×10^6 sample points of the complete dataset in 10^5 consecutive blocks with 11 elements. The first 10 elements of the block are assigned to the training set and the 11th element to the test set. To solve for the optimal classifier we implement SGD and oLBFGS by selecting feature vectors $\mathbf{x}_i$ at random from the training set $\mathcal{S}$. In all of our numerical experiments the regularization parameter in (40) is $\lambda = 10^{-6}$. The stepsizes for both algorithms are of the form $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$. We set $\epsilon_0 = 10^{-2}$ and $T_0 = 10^4$ for oLBFGS and $\epsilon_0 = 10^{-1}$ and $T_0 = 10^6$ for SGD. For SGD the sample size in (9) is set to $L = 20$ whereas for oLBFGS it is set to $L = 100$. The values of parameters ϵ_0, T_0, and L are chosen to yield best convergence times in a rough parameter optimization search. Observe the relatively large values of L that are used to compute stochastic gradients. This is necessary due to the extreme sparsity of the feature vectors $\mathbf{x}_i$ that contain an average of only 20.9 nonzero out 174,026 elements. Even when considering $L = 100$ vectors they are close to orthogonal. The size of memory for oLBFGS is set to $\tau = 10$. With $L = 100$ features with an average sparsity of 20.9 nonzero elements and memory $\tau = 10$ the cost of each LBGS iteration is in the order of 2.1×10^4 operations.

Figure 5 illustrates the convergence path of SGD and oLBFGS on the advertising training set. We depict the value of the log likelihood objective in (40) evaluated at $\mathbf{w} = \mathbf{w}_t$ where $\mathbf{w}_t$ is the classifier iterate determined by SGD or oLBFGS. The horizontal axis is scaled by the number of feature vectors L that are used in the evaluation of stochastic gradients. This results in a plot of log

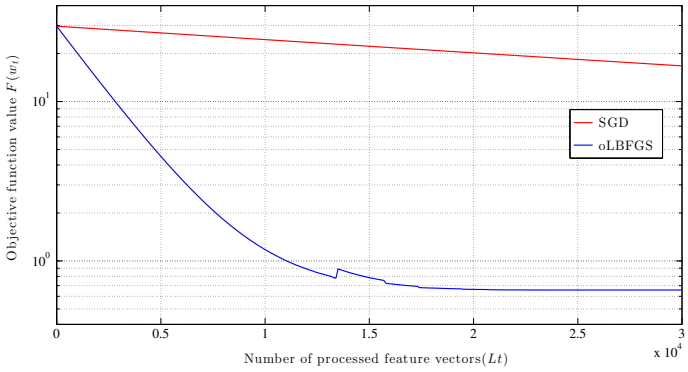


Figure 5: Illustration of Negative log-likelihood value for oLBFGS and SGD after processing certain amount of feature vectors. The accuracy of oLBFGS is better than SGD after processing a specific number of feature vectors.

likelihood cost versus the number Lt of feature vectors processed. To read iteration indexes from Figure 5 divide the horizontal axis values by $L = 100$ for oLBGS and $L = 20$ for SGD. Consistent with the synthetic data results in Section 4, the curvature correction of oLBFGS results in significant reductions in convergence time. For way of illustration observe that after processing $Lt = 3 \times 10^4$ feature vectors the objective value achieved by oLBFGS is $F(\mathbf{w}_t) = 0.65$, while for SGD it still stands at $F(\mathbf{w}_t) = 16$ which is a meager reduction from the random initialization point at which $F(\mathbf{w}_0) = 30$. In fact, oLBFGS converges to the minimum possible log likelihood cost $F(\mathbf{w}_t) = 0.65$ after processing 1.7×10^4 feature vectors. This illustration hints that oLBGS makes better use of the information available in feature vectors.

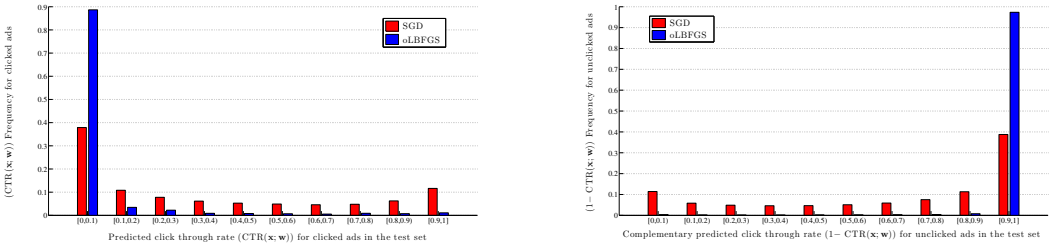
To corroborate that the advantage of oLBGS is not just an artifact of the structure of the log likelihood cost in (40) we process 2×10^4 feature vectors with SGD and oLBFGS and evaluate the predictive accuracy of the respective classifiers on the test set. As measures of predictive accuracy we adopt the frequency histogram of the predicted click through rate $\text{CTR}(\mathbf{x}; \mathbf{w})$ for all clicked ads and the frequency histogram of the complementary predicted click through rate $1 - \text{CTR}(\mathbf{x}; \mathbf{w})$ for all the ads that were *not* clicked. To do so we separate the test set by defining the set $\mathcal{T}_1 := \{(\mathbf{x}_i, y_i) \in \mathcal{T} : y_i = 1\}$ of clicked ads and the set $\mathcal{T}_{-1} := \{(\mathbf{x}_i, y_i) \in \mathcal{T} : y_i = -1\}$ of ads in the test set that were not clicked. For a given classifier $\mathbf{w}$ we compute the predicted probability $\text{CTR}(\mathbf{x}_i; \mathbf{w})$ for each of the ads in the clicked set $\mathcal{T}_1$. We then consider a given interval $[a, b]$ and define the frequency histogram of the predicted click through rate as the fraction of clicked ads for which the prediction $\text{CTR}(\mathbf{x}_i; \mathbf{w})$ falls in $[a, b]$,

$$\mathcal{H}_1(\mathbf{w}; a, b) := \frac{1}{\#(\mathcal{T}_1)} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}_1} \mathbb{I}\Big\{\text{CTR}(\mathbf{x}_i; \mathbf{w}) \in [a, b]\Big\}, \tag{41}$$

where $\#(\mathcal{T}_1)$ denotes the cardinality of the set $\mathcal{T}_1$. Likewise, we consider the ads in the set $\mathcal{T}_{-1}$ that were not clicked and compute the prediction $1 - \text{CTR}(\mathbf{x}_i; \mathbf{w})$ on the probability of the ad not being clicked. We then consider a given interval $[a, b]$ and define the frequency histogram $\mathcal{H}_{-1}(\mathbf{w}; a, b)$ as the fraction of unclicked ads for which the prediction $1 - \text{CTR}(\mathbf{x}_i; \mathbf{w})$ falls in $[a, b]$,

$$\mathcal{H}_{-1}(\mathbf{w}; a, b) := \frac{1}{\#(\mathcal{T}_{-1})} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}_{-1}} \mathbb{I}\Big\{1 - \text{CTR}(\mathbf{x}_i; \mathbf{w}) \in [a, b]\Big\}. \tag{42}$$

The histogram $\mathcal{H}_1(\mathbf{w}; a, b)$ in (41) allows us to study how large the predicted probability $\text{CTR}(\mathbf{x}_i; \mathbf{w})$ is for the clicked ads. Conversely, the histogram $\mathcal{H}_{-1}(\mathbf{w}; a, b)$ in (42) gives an indication of how large



(a) Histogram $\mathcal{H}_1(\mathbf{w}; a, b)$, [cf. (41)].

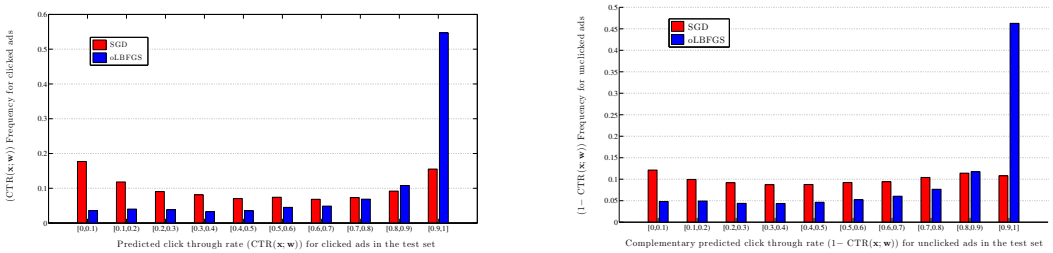
(b) Histogram $\mathcal{H}_{-1}(\mathbf{w}; a, b)$, [cf. (42)].

Figure 6: Performance of classifier after processing 2×10^4 feature vectors with SGD and oLBFGS for the cost in (40). Histograms for: (a) predicted click through rate $\text{CTR}(\mathbf{x}; \mathbf{w})$ for all clicked ads; and (b) complementary predicted click through rate $1 - \text{CTR}(\mathbf{x}; \mathbf{w})$ for all unclicked ads. For an ideal classifier that predicts a click probability $\text{CTR}(\mathbf{x}; \mathbf{w}) = 1$ for all clicked ads and a click probability $\text{CTR}(\mathbf{x}; \mathbf{w}) = 0$ for all unclicked ads the frequency counts in $\mathcal{H}_1(\mathbf{w}; a, b)$ and $\mathcal{H}_{-1}(\mathbf{w}; a, b)$ would accumulate in the $[0.9, 1]$ bin. Neither SGD nor oLBFGS compute acceptable classifiers because the number of clicked ads in the test set is very small and predicting $\text{CTR}(\mathbf{x}; \mathbf{w}) = 0$ for all ads is close to the minimum of (40).

the predicted probability $1 - \text{CTR}(\mathbf{x}_i; \mathbf{w})$ is for the unclicked ads. An ideal classifier is one for which the frequency counts in $\mathcal{H}_1(\mathbf{w}; a, b)$ accumulate at $\text{CTR}(\mathbf{x}_i; \mathbf{w}) = 1$ and for which $\mathcal{H}_{-1}(\mathbf{w}; a, b)$ accumulates observations at $1 - \text{CTR}(\mathbf{x}_i; \mathbf{w}) = 1$. This corresponds to a classifier that predicts a click probability of 1 for all ads that were clicked and a click probability of 0 for all ads that were not clicked.

Fig. 6(a) shows the histograms of predicted click through rate $\text{CTR}(\mathbf{x}; \mathbf{w})$ for all clicked ads by oLBFGS and SGD classifiers after processing 2×10^4 training sample points. oLBFGS classifier for 88% of test points in $\mathcal{T}_1$ predicts $\text{CTR}(\mathbf{x}; \mathbf{w})$ in the interval $[0, 0.1]$ and the classifier computed by SGD estimates the click through rate $\text{CTR}(\mathbf{x}; \mathbf{w})$ in the same interval for 37% of clicked ads in the test set. These numbers shows the inaccurate click through rate predictions of both classifiers for the test points with label $y = 1$. Although, SGD and oLBFGS classifiers have catastrophic performances in predicting click through rate $\text{CTR}(\mathbf{x}; \mathbf{w})$ for the clicked ads in the test set, they perform well in estimating complementary predicted click through rate $1 - \text{CTR}(\mathbf{x}; \mathbf{w})$ for the test points with label $y = -1$. This observation implied by Fig. 6(b) which shows the histograms of complementary predicted click through rate $1 - \text{CTR}(\mathbf{x}; \mathbf{w})$ for all *not* clicked ads by oLBFGS and SGD classifiers after processing 2×10^4 training sample points. As it shows after processing 2×10^4 sample points of the training set the predicted probability $1 - \text{CTR}(\mathbf{x}; \mathbf{w})$ by the SGD classifier for 38.8% of the test points are in the interval $[0.9, 1]$, while for the classifier computed by oLBFGS 97.3% of predicted probability $1 - \text{CTR}(\mathbf{x}; \mathbf{w})$ are in the interval $[0.9, 1]$ which is a significant performance.

The reason for the inaccurate predictions of both classifiers is that most elements in the training set $\mathcal{S}$ are unclicked ads. Thus, the minimizer $\mathbf{w}^*$ of the log likelihood cost in (40) is close to a classifier that predicts $\text{CTR}(\mathbf{x}; \mathbf{w}^*) \approx 0$ for most ads. Indeed, out of the 10^6 elements in the training set, 94.8% of them have labels $y_i = -1$ and only the remaining 5.2×10^4 feature vectors correspond to clicked ads. To overcome this problem we replicate observations with labels $y_i = 1$ to balance the representation of both labels in the training set. Equivalently, we introduce a constant γ and redefine the log likelihood objective in (40) to give a larger weight to feature vectors that correspond



(a) Histogram $\mathcal{H}_1(\mathbf{w}; a, b)$, [cf. (41)]. (b) Histogram $\mathcal{H}_{-1}(\mathbf{w}; a, b)$, [cf. (42)].

Figure 7: Performance of classifier after processing 2×10^4 feature vectors with SGD and oLBFGS for the cost in (43). Histograms for: (a) predicted click through rate $\text{CTR}(\mathbf{x}; \mathbf{w})$ for all clicked ads; and (b) complementary predicted click through rate $1 - \text{CTR}(\mathbf{x}; \mathbf{w})$ for all unclicked ads. For an ideal classifier that predicts a click probability $\text{CTR}(\mathbf{x}; \mathbf{w}) = 1$ for all clicked ads and a click probability $\text{CTR}(\mathbf{x}; \mathbf{w}) = 0$ for all unclicked ads the frequency counts in $\mathcal{H}_1(\mathbf{w}; a, b)$ and $\mathcal{H}_{-1}(\mathbf{w}; a, b)$ would accumulate in the $[0.9, 1]$ bin. The classifier computed by oLBFGS is much more accurate than the one computed by SGD.

to clicked ads,

$$\mathbf{w}^* \;=\; \text{argmin}\; \frac{\lambda}{2}\|\mathbf{w}\|^2 \;+\; \frac{1}{M}\Bigg[\; \gamma \sum_{\mathbf{x}_i \in \mathcal{S}_1} \log\Big(1 + \exp(-\mathbf{x}_i^T \mathbf{w})\Big) + \sum_{\mathbf{x}_i \in \mathcal{S}_{-1}} \log\Big(1 + \exp(\mathbf{x}_i^T \mathbf{w})\Big)\Bigg], \quad (43)$$

where we defined $M := \gamma \#(\mathcal{S}_1) + \#(\mathcal{S}_{-1})$ to account for the replication of clicked featured vectors that is implicit in (43). To implement SGD and oLBFGS in the weighted log function in (43) we need to bias the random choice of feature vector so that vectors in $\mathcal{S}_1$ are γ times more likely to be selected than vectors in $\mathcal{S}_2$. Although our justification to introduce γ is to balance the types of feature vectors, γ is just a tradeoff constant to increase the percentage of correct predictions for clicked ads – which is close to zero in Figure 6 – at the cost of reducing the accuracy of correct predictions of unclicked ads – which is close to one in Figure 6.

We repeat the experiment of processing 2×10^4 feature vectors that we sumamrized in Figure 6 but now we use the objective cost in (43) instead of the cost in (40). We set $\gamma = 18.2$ which makes replicated clicked ads as numerous as unclicked ads. The resulting SGD and oLBFGS histograms of the predicted click through rates for all clicked ads and complementary predicted click through rates for all unclicked ads are shown in Figure 7. In particular, Figure 7(a) shows the histograms of predicted click through rate $\text{CTR}(\mathbf{x}; \mathbf{w})$ for all clicked ads after processing 2×10^4 training sample points. The modification of the log likelihood cost increases the accuracy of the oLBFGS classifier which is now predicting a click probability $\text{CTR}(\mathbf{x}; \mathbf{w}) \in [0.9, 1]$ for 54.7% of the ads that were indeed clicked. There is also improvement for the SGD classifier but the prediction is much less impressive. Only 15.5% of the clicked ads are associated with a click probability prediction in the interval $[0.9, 1]$. This improvement is at the cost of reducing the complementary predicted click through rate $1 - \text{CTR}(\mathbf{x}; \mathbf{w})$ for the ads that were indeed not clicked. However, the classifier computed by oLBFGS after processing 2×10^4 feature vectors still predicts a probability $1 - \text{CTR}(\mathbf{x}; \mathbf{w}) \in [0.9, 1]$ for 46.3% of the unclicked ads. The corresponding frequency for the SGD classifier is 10.8%.

Do note that the relatively high prediction accuracies in Figure 7 are a reflection of sample bias to some extent. Since ads were chosen for display because they were deemed likely to be clicked they are not a completely random test set. Still, the point to be made here is that oLBFGS succeeds in finding an optimal classifier when SGD fails. It would take the processing of about 10^6 feature vectors for SGD to achieve the same accuracy of oLBFGs.

6. Conclusions

An online limited memory version of the (oL)BFGS algorithm was studied for solving strongly convex optimization problems with stochastic objectives. Almost sure convergence was established by bounding the traces and determinants of curvature estimation matrices under the assumption that sample functions have well behaved Hessians. The convergence rate of oLBFGS was further determined to be at least of order $O(1/t)$ in expectation. This rate is customary of stochastic optimization algorithms which are limited by their ability to smooth out the noise in stochastic gradient estimates. The application of oLBFGS to support vector machines was also developed and numerical tests on synthetic data were provided. The numerical results show that oLBFGS affords important reductions with respect to stochastic gradient descent (SGD) in terms of the number of feature vectors that need to be processed to achieve a target accuracy as well as in the associated execution time. Moreover, oLBFGS also exhibits a significant execution time reduction when compared to other stochastic quasi-Newton methods. These reductions increase with the problem dimension and can become arbitrarily large. A detailed comparison between oLBFGS and SGD for training a logistic regressor in a large scale search engine advertising problem was also presented. The numerical tests show that oLBFGS trains the regressor using less than 1% of the data required by SGD to obtain similar classification accuracy.

Acknowledgments

We acknowledge the support of the National Science Foundation (NSF CAREER CCF-0952867) and the Office of Naval Research (ONR N00014-12-1-0997).

Appendix A. Proof of Proposition 1

We begin by observing that the $\mathbf{p}_u$ sequence in (17) is defined so that we can write $\mathbf{p}_{u+1} = \hat{\mathbf{Z}}_{t-u-1}\mathbf{p}_u$ with $\mathbf{p}_0 = \mathbf{p}$. Indeed, use the explicit expression for $\hat{\mathbf{Z}}_{t-u-1}$ in (12) to write the product $\hat{\mathbf{Z}}_{t-u-1}\mathbf{p}_u$ as

$$\hat{\mathbf{Z}}_{t-u-1}\mathbf{p}_u \;=\; \Big(\mathbf{I} - \hat{\rho}_{t-u-1}\hat{\mathbf{r}}_{t-u-1}\mathbf{v}_{t-u-1}^T\Big)\mathbf{p}_u \;=\; \mathbf{p}_u - \alpha_u\hat{\mathbf{r}}_{t-u-1} \;=\; \mathbf{p}_{u+1}, \tag{44}$$

where the second equality follows from the definition $\alpha_u := \hat{\rho}_{t-u-1}\mathbf{v}_{t-u-1}^T\mathbf{p}_u$ and the third equality from the definition of the $\mathbf{p}_u$ sequence in (17).

Recall now the oLBFGS Hessian inverse approximation expression in (16). It follows that for computing the product $\hat{\mathbf{B}}_t^{-1}\mathbf{p}$ we can multiply each of the $\tau+1$ summands in the right hand side of (16) by $\mathbf{p} = \mathbf{p}_0$. Implementing this procedure yields

$$\begin{aligned}\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \Big(\hat{\mathbf{Z}}_{t-1}^T\ldots\hat{\mathbf{Z}}_{t-\tau}^T\Big)\hat{\mathbf{B}}_{t,0}^{-1}\Big(\hat{\mathbf{Z}}_{t-\tau}\ldots\hat{\mathbf{Z}}_{t-1}\Big)\,\mathbf{p}_0 + \hat{\rho}_{t-\tau}\Big(\hat{\mathbf{Z}}_{t-1}^T\ldots\hat{\mathbf{Z}}_{t-\tau+1}^T\Big)\,\mathbf{v}_{t-\tau}\mathbf{v}_{t-\tau}^T\Big(\hat{\mathbf{Z}}_{t-\tau+1}\ldots\hat{\mathbf{Z}}_{t-1}\Big)\,\mathbf{p}_0\\ + \cdots + \hat{\rho}_{t-2}\Big(\hat{\mathbf{Z}}_{t-1}^T\Big)\,\mathbf{v}_{t-2}\mathbf{v}_{t-2}^T\Big(\hat{\mathbf{Z}}_{t-1}\Big)\,\mathbf{p}_0 + \hat{\rho}_{t-1}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^T\mathbf{p}_0.\end{aligned} \tag{45}$$

The fundamental observation in (45) is that all summands except the last contain the product $\hat{\mathbf{Z}}_{t-1}\mathbf{p}_0$. This product cannot only be computed efficiently but, as shown in (44), is given by $\mathbf{p}_1 = \hat{\mathbf{Z}}_{t-1}\mathbf{p}_0$. A not so fundamental, yet still important observation, is that the last term can be simplified to $\hat{\rho}_{t-1}\mathbf{v}_{t-1}\mathbf{v}_{t-1}^T\mathbf{p}_0 = \alpha_0\mathbf{v}_{t-1}$ given the definition of $\alpha_0 := \hat{\rho}_{t-1}\mathbf{v}_{t-1}^T\mathbf{p}_0$. Implementing both of these substitutions in (45) yields

$$\begin{aligned}\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \Big(\hat{\mathbf{Z}}_{t-1}^T\ldots\hat{\mathbf{Z}}_{t-\tau}^T\Big)\hat{\mathbf{B}}_{t,0}^{-1}\Big(\hat{\mathbf{Z}}_{t-\tau}\ldots\hat{\mathbf{Z}}_{t-2}\Big)\,\mathbf{p}_1 + \hat{\rho}_{t-\tau}\Big(\hat{\mathbf{Z}}_{t-1}^T\ldots\hat{\mathbf{Z}}_{t-\tau+1}^T\Big)\,\mathbf{v}_{t-\tau}\mathbf{v}_{t-\tau}^T\Big(\hat{\mathbf{Z}}_{t-\tau+1}\ldots\hat{\mathbf{Z}}_{t-2}\Big)\,\mathbf{p}_1\\ + \cdots + \hat{\rho}_{t-2}\Big(\hat{\mathbf{Z}}_{t-1}^T\Big)\,\mathbf{v}_{t-2}\mathbf{v}_{t-2}^T\mathbf{p}_1 + \alpha_0\mathbf{v}_{t-1}.\end{aligned} \tag{46}$$

The structure of (46) is analogous to the structure of (45). In all terms except the last two we require determination of the product $\hat{\mathbf{Z}}_{t-2}\mathbf{p}_1$, which, as per (44) can be computed with $2n$ multiplications and is given by $\mathbf{p}_2 = \hat{\mathbf{Z}}_{t-2}\mathbf{p}_1$. Likewise, in the second to last term we can simplify the product $\hat{\rho}_{t-2}\mathbf{v}_{t-2}\mathbf{v}_{t-2}^T\mathbf{p}_1 = \alpha_1\mathbf{v}_{t-2}$ using the definition $\alpha_1 = \hat{\rho}_{t-2}\mathbf{v}_{t-2}^T\mathbf{p}_1$. Implementing these substitutions in (46) yields an expression that is, again, analogous. In all of the resulting summands except the last three we need to compute the product $\hat{\mathbf{Z}}_{t-3}\mathbf{p}_2$, which is given by $\mathbf{p}_3 = \hat{\mathbf{Z}}_{t-3}\mathbf{p}_2$ and in the third to last term we can simplify the product $\hat{\rho}_{t-3}\mathbf{v}_{t-3}\mathbf{v}_{t-3}^T\mathbf{p}_2 = \alpha_2\mathbf{v}_{t-3}$. Repeating this process keeps yielding terms with analogous structure and, after $\tau - 1$ repetitions we simplify (46) to

$$\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \left(\hat{\mathbf{Z}}_{t-1}^T \ldots \hat{\mathbf{Z}}_{t-\tau+1}^T \hat{\mathbf{Z}}_{t-\tau}^T\right) \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau + \left(\hat{\mathbf{Z}}_{t-1}^T \ldots \hat{\mathbf{Z}}_{t-\tau+1}^T\right) \alpha_{\tau-1}\mathbf{v}_{t-\tau} + \ldots + \hat{\mathbf{Z}}_{t-1}^T \alpha_1 \mathbf{v}_{t-2} + \alpha_0 \mathbf{v}_{t-1}. \tag{47}$$

In the first summand in (47) we can substitute the definition of the first element of the $\mathbf{q}_u$ sequence $\mathbf{q}_0 := \hat{\mathbf{B}}_{t,0}^{-1}\mathbf{p}_\tau$. More important, observe that the matrix $\hat{\mathbf{Z}}_{t-1}^T$ is the first factor in all but the last summand. Likewise, the matrix $\hat{\mathbf{Z}}_{t-2}^T$ is the second factor in all but the last two summands and, in general, the matrix $\hat{\mathbf{Z}}_{t-u}^T$ is the uth factor in all but the last u summands. Pulling these common factors recursively through (47) it follows that $\hat{\mathbf{B}}_t^{-1}\mathbf{p}_t$ can be equivalently written as

$$\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \alpha_0\mathbf{v}_{t-1} + \hat{\mathbf{Z}}_{t-1}^T \left[\alpha_1 \mathbf{v}_{t-2} + \hat{\mathbf{Z}}_{t-2}^T \left[\ldots \left[\alpha_{\tau-2}\mathbf{v}_{t-\tau+1} + \hat{\mathbf{Z}}_{t-\tau+1}^T \left[\alpha_{\tau-1}\mathbf{v}_{t-\tau} + \hat{\mathbf{Z}}_{t-\tau}^T \mathbf{q}_0\right]\right] \ldots \right]\right]. \tag{48}$$

To conclude the proof we just need to note that the recursive definition of $\mathbf{q}_u$ in (18) is a computation of the nested elements of (48). To see this consider the innermost element of (48) and use the definition of $\beta_0 := \hat{\rho}_{t-\tau}\hat{\mathbf{r}}_{t-\tau}^T\mathbf{q}_0$ to conclude that $\alpha_{\tau-1}\mathbf{v}_{t-\tau} + \hat{\mathbf{Z}}_{t-\tau}^T\mathbf{q}_0$ is given by

$$\alpha_{\tau-1}\mathbf{v}_{t-\tau} + \hat{\mathbf{Z}}_{t-\tau}^T\mathbf{q}_0 \;=\; \alpha_{\tau-1}\mathbf{v}_{t-\tau} + \mathbf{q}_0 - \hat{\rho}_{t-\tau}\mathbf{v}_{t-\tau}\hat{\mathbf{r}}_{t-\tau}^T\mathbf{q}_0 \;=\; \mathbf{q}_0 + (\alpha_{\tau-1} - \beta_0)\mathbf{v}_{t-\tau} \;=\; \mathbf{q}_1 \tag{49}$$

where in the last equality we use the definition of $\mathbf{q}_1$ [cf. (18). Substituting this simplification into (48) eliminates the innermost nested term and leads to

$$\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \alpha_0\mathbf{v}_{t-1} + \hat{\mathbf{Z}}_{t-1}^T \left[\alpha_1 \mathbf{v}_{t-2} + \hat{\mathbf{Z}}_{t-2}^T \left[\ldots \left[\alpha_{\tau-2}\mathbf{v}_{t-\tau+1} + \hat{\mathbf{Z}}_{t-\tau+1}^T \mathbf{q}_1\right] \ldots \right]\right]. \tag{50}$$

Mimicking the computations in (49) we can see that the innermost term in (50) is $\alpha_{\tau-2}\mathbf{v}_{t-\tau+1} + \hat{\mathbf{Z}}_{t-\tau+1}^T\mathbf{q}_1 = \mathbf{q}_2$ and obtain an analogous expression that we can substitute for $\mathbf{q}_3$ and so on. Repeating this process $\tau - 2$ times leads to the last term being $\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \alpha_0\mathbf{v}_{t-1} + \hat{\mathbf{Z}}_{t-1}^T\mathbf{q}_{\tau-1}$ which we can write as $\alpha_0\mathbf{v}_{t-1} + \hat{\mathbf{Z}}_{t-1}^T\mathbf{q}_{\tau-1} = \mathbf{q}_\tau$ by repeating the operations in (49). This final observation yields $\hat{\mathbf{B}}_t^{-1}\mathbf{p} = \mathbf{q}_\tau$.

Appendix B. Proof of Lemma 2

As per (22) in Assumption 1 the eigenvalues of the instantaneous Hessian $\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})$ are bounded by $\tilde{m}$ and $\tilde{M}$. Thus, for any given vector $\mathbf{z}$ it holds

$$\tilde{m}\|\mathbf{z}\|^2 \leq \mathbf{z}^T\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}})\mathbf{z} \leq \tilde{M}\|\mathbf{z}\|^2. \tag{51}$$

For given $\mathbf{w}_t$ and $\mathbf{w}_{t+1}$ define the mean instantaneous Hessian $\hat{\mathbf{G}}_t$ as the average Hessian value along the segment $[\mathbf{w}_t, \mathbf{w}_{t+1}]$

$$\hat{\mathbf{G}}_t = \int_0^1 \hat{\mathbf{H}}\left(\mathbf{w}_t + \tau(\mathbf{w}_{t+1} - \mathbf{w}_t), \tilde{\boldsymbol{\theta}}_t\right) d\tau. \tag{52}$$

Consider now the instantaneous gradient $\hat{\mathbf{s}}(\mathbf{w}_t + \tau(\mathbf{w}_{t+1} - \mathbf{w}_t), \tilde{\boldsymbol{\theta}}_t)$ evaluated at $\mathbf{w}_t + \tau(\mathbf{w}_{t+1} - \mathbf{w}_t)$ and observe that its derivative with respect to τ is $\partial\hat{\mathbf{s}}\big(\mathbf{w}_t + \tau(\mathbf{w}_{t+1} - \mathbf{w}_t), \tilde{\boldsymbol{\theta}}_t\big)/\partial\tau = \hat{\mathbf{H}}(\mathbf{w}_t + \tau(\mathbf{w}_{t+1} - \mathbf{w}_t), \tilde{\boldsymbol{\theta}}_t)(\mathbf{w}_{t+1} - \mathbf{w}_t)$. Then according to the fundamental theorem of calculus

$$\int_0^1 \hat{\mathbf{H}}\left(\mathbf{w}_t + \tau(\mathbf{w}_{t+1} - \mathbf{w}_t)\,,\, \tilde{\boldsymbol{\theta}}_t\right)(\mathbf{w}_{t+1} - \mathbf{w}_t)\; d\tau \;=\; \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t). \tag{53}$$

Using the definitions of the mean instantaneous Hessian $\hat{\mathbf{G}}_t$ in (52) as well as the definitions of the stochastic gradient variations $\hat{\mathbf{r}}_t$ and variable variations $\mathbf{v}_t$ in (11) and (4) we can rewrite (53) as

$$\hat{\mathbf{G}}_t \mathbf{v}_t = \hat{\mathbf{r}}_t. \tag{54}$$

Invoking (51) for the integrand in (52), i.e., for $\hat{\mathbf{H}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) = \hat{\mathbf{H}}\big(\mathbf{w}_t + \tau(\mathbf{w}_{t+1} - \mathbf{w}_t), \tilde{\boldsymbol{\theta}}\big)$, it follows that for all vectors $\mathbf{z}$ the mean instantaneous Hessian $\hat{\mathbf{G}}_t$ satisfies

$$\tilde{m}\|\mathbf{z}\|^2 \leq \mathbf{z}^T \hat{\mathbf{G}}_t \mathbf{z} \leq \tilde{M}\|\mathbf{z}\|^2. \tag{55}$$

The claim in (26) follows from (54) and (55). Indeed, consider the ratio of inner products $\hat{\mathbf{r}}_t^T \mathbf{v}_t / \mathbf{v}_t^T \mathbf{v}_t$ and use (54) and the first inequality in (55) to write

$$\frac{\hat{\mathbf{r}}_t^T \mathbf{v}_t}{\mathbf{v}_t^T \mathbf{v}_t} = \frac{\mathbf{v}_t^T \hat{\mathbf{G}}_t \mathbf{v}_t}{\mathbf{v}_t^T \mathbf{v}_t} \geq \tilde{m}. \tag{56}$$

It follows that (26) is true for all times t.

To prove (27) we operate (54) and (55). Considering the ratio of inner products $\hat{\mathbf{r}}_t^T \hat{\mathbf{r}}_t / \hat{\mathbf{r}}_t^T \mathbf{v}_t$ and observing that (54) states $\hat{\mathbf{G}}_t \mathbf{v}_t = \hat{\mathbf{r}}_t$, we can write

$$\frac{\hat{\mathbf{r}}_t^T \hat{\mathbf{r}}_t}{\hat{\mathbf{r}}_t^T \mathbf{v}_t} = \frac{\mathbf{v}_t^T \hat{\mathbf{G}}_t^2 \mathbf{v}_t}{\mathbf{v}_t^T \hat{\mathbf{G}}_t \mathbf{v}_t} \tag{57}$$

Since the mean instantaneous Hessian $\hat{\mathbf{G}}_t$ is positive definite according to (55), we can define $\mathbf{z}_t = \hat{\mathbf{G}}_t^{1/2} \mathbf{v}_t$. Substituting this observation into (57) we can conclude

$$\frac{\hat{\mathbf{r}}_t^T \hat{\mathbf{r}}_t}{\hat{\mathbf{r}}_t^T \mathbf{v}_t} = \frac{\mathbf{z}_t^T \hat{\mathbf{G}}_t \mathbf{z}_t}{\mathbf{z}_t^T \mathbf{z}_t}. \tag{58}$$

Observing (58) and the inequalities in (55), it follows that (27) is true.

Appendix C. Proof of Lemma 3

We begin with the trace upper bound in (29). Consider the recursive update formula for the Hessian approximation $\hat{\mathbf{B}}_t$ as defined in (28). To simplify notation we define s as a new index such that $s = t - \tau + u$. Introduce this simplified notation in (28) and compute the trace of both sides. Since traces are linear function of their arguments we obtain

$$\operatorname{tr}\left(\hat{\mathbf{B}}_{t,u+1}\right) = \operatorname{tr}\left(\hat{\mathbf{B}}_{t,u}\right) - \operatorname{tr}\left(\frac{\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}\right) + \operatorname{tr}\left(\frac{\hat{\mathbf{r}}_s\hat{\mathbf{r}}_s^T}{\mathbf{v}_s^T\hat{\mathbf{r}}_s}\right). \tag{59}$$

Recall that the trace of a matrix product is independent of the order of the factors to conclude that the second summand of (59) can be simplified to

$$\operatorname{tr}\left(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\right) \;=\; \operatorname{tr}\left(\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\right) \;=\; \mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\hat{\mathbf{B}}_{t,u}\mathbf{v}_s \;=\; \left\|\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\right\|^2, \tag{60}$$

where the second equality follows because $\mathbf{v}_s^T \hat{\mathbf{B}}_{t,u} \hat{\mathbf{B}}_{t,u} \mathbf{v}_s$ is a scalar and the second equality by observing that the term $\mathbf{v}_s^T \hat{\mathbf{B}}_{t,u} \hat{\mathbf{B}}_{t,u} \mathbf{v}_s$ is the inner product of the vector $\hat{\mathbf{B}}_{t,u} \mathbf{v}_s$ with itself. Use the same procedure for the last summand of (59) so as to write $\text{tr}(\hat{\mathbf{r}}_s \hat{\mathbf{r}}_s^T) = \hat{\mathbf{r}}_s^T \hat{\mathbf{r}}_s = \|\hat{\mathbf{r}}_s\|^2$. Substituting this latter observation as well as (60) into (59) we can simplify the trace of $\hat{\mathbf{B}}_{t,u+1}$ to

$$\text{tr}\left(\hat{\mathbf{B}}_{t,u+1}\right) = \text{tr}\left(\hat{\mathbf{B}}_{t,u}\right) - \frac{\|\hat{\mathbf{B}}_{t,u} \mathbf{v}_s\|^2}{\mathbf{v}_s^T \hat{\mathbf{B}}_{t,u} \mathbf{v}_s} + \frac{\|\hat{\mathbf{r}}_s\|^2}{\hat{\mathbf{r}}_s^T \mathbf{v}_s}. \tag{61}$$

The second term in the right hand side of (61) is negative because, as we have already shown, the matrix $\hat{\mathbf{B}}_{t,u}$ is positive definite. The third term is the one for which we have derived the bound that appears in (27) of Lemma 2. Using this two observations we can conclude that the trace of $\hat{\mathbf{B}}_{t,u+1}$ can be bounded as

$$\text{tr}\left(\hat{\mathbf{B}}_{t,u+1}\right) \leq \text{tr}\left(\hat{\mathbf{B}}_{t,u}\right) + \tilde{M}. \tag{62}$$

By considering (62) as a recursive expression for $u = 0, \dots \tau - 1$, we can conclude that

$$\text{tr}\left(\hat{\mathbf{B}}_{t,u}\right) \leq \text{tr}\left(\hat{\mathbf{B}}_{t,0}\right) + u\tilde{M}. \tag{63}$$

To finalize the proof of (29) we need to find a bound for the initial trace $\text{tr}(\hat{\mathbf{B}}_{t,0})$. To do so we consider the definition $\hat{\mathbf{B}}_{t,0} = \mathbf{I}/\hat{\gamma}_t$ with $\hat{\gamma}_t$ as given by (19). Using this definition of $\hat{\mathbf{B}}_{t,0}$ as a scaled identity it follows that we can write the trace of $\hat{\mathbf{B}}_{t,0}$ as

$$\text{tr}\left(\hat{\mathbf{B}}_{t,0}\right) \;=\; \text{tr}\left(\frac{\mathbf{I}}{\hat{\gamma}_t}\right) \;=\; \frac{n}{\hat{\gamma}_t}. \tag{64}$$

Substituting the definition of $\hat{\gamma}_t$ into the rightmost side of (19) it follows that for all times $t \geq 1$,

$$\text{tr}\left(\hat{\mathbf{B}}_{t,0}\right) \;=\; n \frac{\hat{\mathbf{r}}_{t-1}^T \hat{\mathbf{r}}_{t-1}}{\mathbf{v}_{t-1}^T \hat{\mathbf{r}}_{t-1}} \;=\; n \frac{\|\hat{\mathbf{r}}_{t-1}\|^2}{\mathbf{v}_{t-1}^T \hat{\mathbf{r}}_{t-1}}\,. \tag{65}$$

The term $\|\hat{\mathbf{r}}_{t-1}\|^2 / \mathbf{v}_{t-1}^T \hat{\mathbf{r}}_{t-1}$ in (77) is of the same form of the rightmost term in (61). We can then, as we did in going from (61) to (62) apply the bound that we provide in (27) of Lemma 2 to conclude that for all times $t \geq 1$

$$\text{tr}\left(\hat{\mathbf{B}}_{t,0}\right) \;\leq\; n\tilde{M}. \tag{66}$$

Substituting (66) into (63) and pulling common factors leads to the conclusion that for all times $t \geq 1$ and indices $0 \leq u \leq \tau$ it holds

$$\text{tr}\left(\hat{\mathbf{B}}_{t,u}\right) \;\leq\; (n+u)\tilde{M}. \tag{67}$$

The bound in (29) follows by making $u = \tau$ in (67) and recalling that, by definition, $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$. For time $t = 0$ we have $\hat{\gamma}_t = \hat{\gamma}_0 = 1$ and (77) reduces to $\text{tr}(\hat{\mathbf{B}}_{t,0}) = n$ while (67) reduces to $\text{tr}(\hat{\mathbf{B}}_{t,\tau}) \leq (1+\tau)\tilde{M}$. Furthermore, for $t < \tau$ we make $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,t}$ instead of $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$. In this case the bound in (67) can be tightened to $\text{tr}(\hat{\mathbf{B}}_{t,\tau}) \leq (n+t)\tilde{M}$. Given that we are interested in an asymptotic convergence analysis, these bounds are inconsequential.

We consider now the determinant lower bound in (30). As we did in (59) begin by considering the recursive update in (28) and define s as a new index such that $s = t - \tau + u$ to simplify notation. Compute the determinant of both sides of (28), factorize $\hat{\mathbf{B}}_{t,u}$ on the right hand side, and use the fact that the determinant of a product is the product of the determinants to conclude that

$$\det\left(\hat{\mathbf{B}}_{t,u+1}\right) = \det\left(\hat{\mathbf{B}}_{t,u}\right) \det\left(\mathbf{I} - \frac{\mathbf{v}_s (\hat{\mathbf{B}}_{t,u} \mathbf{v}_s)^T}{\mathbf{v}_s^T \hat{\mathbf{B}}_{t,u} \mathbf{v}_s} + \frac{\hat{\mathbf{B}}_{t,u}^{-1} \hat{\mathbf{r}}_s \hat{\mathbf{r}}_s^T}{\hat{\mathbf{r}}_s^T \mathbf{v}_s}\right). \tag{68}$$

To simplify the right hand side of (68) we should first know that for any vectors $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$ and $\mathbf{u}_4$, we can write $\det(\mathbf{I} + \mathbf{u}_1\mathbf{u}_2^T + \mathbf{u}_3\mathbf{u}_4^T) = (1 + \mathbf{u}_1^T\mathbf{u}_2)(1 + \mathbf{u}_3^T\mathbf{u}_4) - (\mathbf{u}_1^T\mathbf{u}_4)(\mathbf{u}_2^T\mathbf{u}_3)$ – see, e.g., Li and Fukushima (2001), Lemma 3.3). Setting $\mathbf{u}_1 = \mathbf{v}_s$, $\mathbf{u}_2 = \hat{\mathbf{B}}_{t,u}\mathbf{v}_s/\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s$, $\mathbf{u}_3 = \hat{\mathbf{B}}_{t,u}^{-1}\hat{\mathbf{r}}_s$ and $\mathbf{u}_4 = \hat{\mathbf{r}}_s/\hat{\mathbf{r}}_s^T\mathbf{v}_s$, implies that $\det(\mathbf{I} + \mathbf{u}_1\mathbf{u}_2^T + \mathbf{u}_3\mathbf{u}_4^T)$ is equivalent to the last term in the right hand side of (68). Applying these substitutions implies that $(1 + \mathbf{u}_1^T\mathbf{u}_2) = 1 - \mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s/\mathbf{v}_s\hat{\mathbf{B}}_{t,u}\mathbf{v}_s = 0$ and $\mathbf{u}_1^T\mathbf{u}_4 = -\mathbf{v}_s^T\hat{\mathbf{r}}_s/\hat{\mathbf{r}}_s^T\mathbf{v}_s = -1$. Hence, the term $\det(\mathbf{I} + \mathbf{u}_1\mathbf{u}_2^T + \mathbf{u}_3\mathbf{u}_4^T)$ can be simplified as $\mathbf{u}_2^T\mathbf{u}_3$. By this simplification we can write the right hand side of (68) as

$$\det\left[\mathbf{I} - \frac{\mathbf{v}_s(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s)^T}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s} + \frac{\hat{\mathbf{B}}_{t,u}^{-1}\hat{\mathbf{r}}_s\hat{\mathbf{r}}_s^T}{\hat{\mathbf{r}}_s^T\mathbf{v}_s}\right] = \frac{\left(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s\right)^T}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}\hat{\mathbf{B}}_{t,u}^{-1}\hat{\mathbf{r}}_s. \tag{69}$$

To further simplify (69) write $(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s)^T = \mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}^T$ and observer that since $\hat{\mathbf{B}}_{t,u}$ is symmetric we have $\hat{\mathbf{B}}_{t,u}^T\hat{\mathbf{B}}_{t,u}^{-1} = \hat{\mathbf{B}}_{t,u}\hat{\mathbf{B}}_{t,u}^{-1} = \mathbf{I}$. Therefore,

$$\det\left[\mathbf{I} - \frac{\mathbf{v}_s(\hat{\mathbf{B}}_{t,u}\mathbf{v}_s)^T}{\mathbf{v}_i^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s} + \frac{\hat{\mathbf{B}}_{t,u}^{-1}\hat{\mathbf{r}}_s\hat{\mathbf{r}}_s^T}{\hat{\mathbf{r}}_s^T\mathbf{v}_s}\right] = \frac{\hat{\mathbf{r}}_s^T\mathbf{v}_s}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}. \tag{70}$$

Substitute the simplification in (70) for the corresponding factor in (68). Further multiply and divide the right hand side by the nonzero norm $\|\mathbf{v}_s\|$ and regroup terms to obtain

$$\det\left(\hat{\mathbf{B}}_{t,u+1}\right) = \det\left(\hat{\mathbf{B}}_{t,u}\right)\frac{\hat{\mathbf{r}}_s^T\mathbf{v}_s}{\|\mathbf{v}_s\|}\frac{\|\mathbf{v}_s\|}{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}. \tag{71}$$

To bound the third factor in (71) observe that the largest possible value for the normalized quadratic form $\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s/\|\mathbf{v}_s\|^2$ occurs when $\mathbf{v}_s$ is an eigenvector of $\hat{\mathbf{B}}_{t,u}$ associated with its largest eigenvalue. In such case the value attained is precisely the largest eigenvalue of $\hat{\mathbf{B}}_{t,u}$ implying that we can write

$$\frac{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}{\|\mathbf{v}_s\|} \leq \lambda_{\max}\left(\hat{\mathbf{B}}_{t,u}\right). \tag{72}$$

But to bound the largest eigenvalue $\lambda_{\max}(\hat{\mathbf{B}}_{t,u})$ we can just use the fact that the trace of a matrix coincides with the sum of its eigenvalues. In particular, it must be that $\lambda_{\max}(\hat{\mathbf{B}}_{t,u}) \leq \operatorname{tr}(\hat{\mathbf{B}}_{t,u})$ because all the eigenvalues of the positive definite matrix $\hat{\mathbf{B}}_{t,u}$ are positive. Combining this observation with the trace bound in (67) leads to

$$\frac{\mathbf{v}_s^T\hat{\mathbf{B}}_{t,u}\mathbf{v}_s}{\|\mathbf{v}_s\|} \;\leq\; \operatorname{tr}\left(\hat{\mathbf{B}}_{t,u}\right) \;\leq\; (n+u)\tilde{M}. \tag{73}$$

We can also bound the second factor in the right hand side of (71) if we reorder the inequality in (26) of Lemma 2 to conclude that $\hat{\mathbf{r}}_s^T\mathbf{v}_s/\|\mathbf{v}_s\| \leq \tilde{m}$. This bound, along with the inverse of the inequality in (73) substituted in (71) leads to

$$\det\left(\hat{\mathbf{B}}_{t,u+1}\right) \geq \frac{\tilde{m}}{n\tilde{M} + u\tilde{M}}\det\left(\hat{\mathbf{B}}_{t,u}\right). \tag{74}$$

Apply (74) recursively between indexes $u = 0$ and $u = \tau - 1$ and further observing that $u \leq \tau$ in all of the resulting factors it follows that

$$\det\left(\hat{\mathbf{B}}_{t,\tau}\right) \;\geq\; \left[\frac{\tilde{m}}{(n+\tau)\tilde{M}}\right]^{\tau}\det\left(\hat{\mathbf{B}}_{t,0}\right). \tag{75}$$

To finalize the derivation of (30) we just need to bound the determinant of the initial curvature approximation matrix $\hat{\mathbf{B}}_{t,0}$. To do so we consider, again, the definition $\hat{\mathbf{B}}_{t,0} = \mathbf{I}/\hat{\gamma}_t$ with $\hat{\gamma}_t$ as given by (19). Using this definition of $\hat{\mathbf{B}}_{t,0}$ as a scaled identity it follows that we can write the determinant of $\hat{\mathbf{B}}_{t,0}$ as

$$\det\left(\hat{\mathbf{B}}_{t,0}\right) \;=\; \det\left(\frac{\mathbf{I}}{\hat{\gamma}_t}\right) \;=\; \frac{1}{\hat{\gamma}_t^n}. \tag{76}$$

Substituting the definition of $\hat{\gamma}_t$ into the rightmost side of (76) it follows that for all times $t \geq 1$,

$$\det\left(\hat{\mathbf{B}}_{t,0}\right) \;=\; \left(\frac{\hat{\mathbf{r}}_{t-1}^T\hat{\mathbf{r}}_{t-1}}{\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}}\right)^n \;=\; \left(\frac{\|\hat{\mathbf{r}}_{t-1}\|^2}{\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}}\right)^n. \tag{77}$$

The term $\|\hat{\mathbf{r}}_{t-1}\|^2/\mathbf{v}_{t-1}^T\hat{\mathbf{r}}_{t-1}$ has lower and upper bounds that we provide in (27) of Lemma 2. Using the lower bound in (27) it follows that the initial determinant must be such that

$$\det\left(\hat{\mathbf{B}}_{t,0}\right) \geq \tilde{m}^n. \tag{78}$$

Substituting the upper bound in (78) for the determinant of the initial curvature approximation matrix in (75) allows us to conclude that for all times $t \geq 1$

$$\det\left(\hat{\mathbf{B}}_{t,\tau}\right) \;\geq\; \tilde{m}^n\left[\frac{\tilde{m}}{(n+\tau)\tilde{M}}\right]^\tau. \tag{79}$$

The bound in (30) follows by making $u = \tau$ in (79) and recalling that, by definition, $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$. At time $t = 0$ the initialization constant is set to $\hat{\gamma}_t = \hat{\gamma}_0 = 1$ and (78) reduces to $\det(\hat{\mathbf{B}}_{t,0}) = 1$ while (79) reduces to $\det(\hat{\mathbf{B}}_{t,\tau}) \leq [\tilde{m}/(1+\tau)\tilde{M}]^\tau$. For $t < \tau$ we make $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,t}$ instead of $\hat{\mathbf{B}}_t = \hat{\mathbf{B}}_{t,\tau}$. In this case the bound in (67) can be tightened to $\det(\hat{\mathbf{B}}_{t,\tau}) \leq \tilde{m}[\tilde{m}^n/(1+\tau)\tilde{M}]^\tau$. As in the case of the trace, given that we are interested in an asymptotic convergence analysis, these bounds are inconsequential.

Appendix D. Proof of Lemma 4

We first prove the upper bound inequality in (31). Let us define λ_i as the ith largest eigenvalue of matrix $\hat{\mathbf{B}}_t$. Considering the result in Lemma 3 that $\text{tr}(\hat{\mathbf{B}}_t) \leq (n+\tau)\tilde{M}$ for all steps $t \geq 1$, we obtain that the sum of eigenvalues of the Hessian approximation $\hat{\mathbf{B}}_t$ satisfy

$$\sum_{i=1}^n \lambda_i \;=\; \text{tr}\left(\hat{\mathbf{B}}_t\right) \;\leq\; (n+\tau)\tilde{M}. \tag{80}$$

Considering the upper bound for the sum of eigenvalues in (80) and recalling that all the eigenvalues of the matrix $\hat{\mathbf{B}}_t$ are positive because $\hat{\mathbf{B}}_t$ is positive definite, we can conclude that each of the eigenvalues of $\hat{\mathbf{B}}_t$ is less than the upper bound for their sum in (80). We then have $\lambda_i \leq (n+\tau)\tilde{M}$ for all i from where the right inequality in (31) follows.

To prove the lower bound inequality in (31) consider the second result of Lemma 3 which provides a lower bound for the determinant of the Hessian approximation matrix $\hat{\mathbf{B}}_t$. According to the fact that determinant of a matrix is the product of its eigenvalues, it follows that the product of the eigenvalues of $\hat{\mathbf{B}}_t$ is bounded below by the lower bound in (30), or, equivalently, $\prod_{i=1}^n \lambda_i \geq \tilde{m}^{n+\tau}/[(n+\tau)\tilde{M}]^\tau$. Hence, for any given eigenvalue of $\hat{\mathbf{B}}_t$, say λ_j, we have

$$\lambda_j \;\geq\; \frac{1}{\prod_{k=1,k\neq j}^n \lambda_k} \times \frac{\tilde{m}^{n+\tau}}{\left[(n+\tau)\tilde{M}\right]^\tau}. \tag{81}$$

But in the first part of this proof we have already showed that $(n+\tau)\tilde{M}$ is a lower bound for the eigenvalues of $\hat{\mathbf{B}}_t$. We can then conclude that the product of the $n-1$ eigenvalues $\prod_{k=1,k\neq j}^{n}\lambda_k$ is bounded above by $[(n+\tau)\tilde{M}]^{n-1}$, i.e.,

$$\prod_{k=1,k\neq j}^{n}\lambda_k \leq \left[(n+\tau)\tilde{M}\right]^{n-1}. \tag{82}$$

Combining the inequalities in (81) and (82) we conclude that for any specific eigenvalue of $\hat{\mathbf{B}}_t$ can be lower bounded as

$$\lambda_j \;\geq\; \frac{1}{\left[(n+\tau)\tilde{M}\right]^{n-1}} \times \frac{\tilde{m}^{n+\tau}}{\left[(n+\tau)\tilde{M}\right]^{\tau}}. \tag{83}$$

Since inequality (83) is true for all the eigenvalues of $\hat{\mathbf{B}}_t$, the left inequality (31) holds true.

Appendix E. Proof of Lemma 5

The proof is standard in stochastic optimization and provided here for reference. As it follows from Assumption 1 the eigenvalues of the Hessian $\mathbf{H}(\mathbf{w}_t) = \mathbb{E}_{\tilde{\boldsymbol{\theta}}}[\hat{\mathbf{H}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)] = \nabla^2_{\mathbf{w}}F(\mathbf{w}_t)$ are bounded between $0<m$ and $M<\infty$ as stated in (25). Taking a Taylor's expansion of the function $F(\mathbf{w})$ around $\mathbf{w}=\mathbf{w}_t$ and using the upper bound in the Hessian eigenvalues we can write

$$F(\mathbf{w}_{t+1}) \;\leq F(\mathbf{w}_t) + \nabla F(\mathbf{w}_t)^T(\mathbf{w}_{t+1}-\mathbf{w}_t) + \frac{M}{2}\|\mathbf{w}_{t+1}-\mathbf{w}_t\|^2. \tag{84}$$

From the definition of the oLBFGS update in (3) we can write the difference of two consecutive variables $\mathbf{w}_{t+1}-\mathbf{w}_t$ as $-\epsilon_t\hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)$. Making this substitution in (84), taking expectation with $\mathbf{w}_t$ given in both sides of the resulting inequality, and observing the fact that when $\mathbf{w}_t$ is given the Hessian approximation $\hat{\mathbf{B}}_t^{-1}$ is deterministic we can write

$$\mathbb{E}\left[F(\mathbf{w}_{t+1})\,\middle|\,\mathbf{w}_t\right] \leq\; F(\mathbf{w}_t) - \epsilon_t\nabla F(\mathbf{w}_t)^T\hat{\mathbf{B}}_t^{-1}\mathbb{E}\left[\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)\,\middle|\,\mathbf{w}_t\right] + \frac{\epsilon^2 M}{2}\,\mathbb{E}\left[\left\|\hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)\right\|^2\,\middle|\,\mathbf{w}_t\right]. \tag{85}$$

We proceed to bound the third term in the right hand side of (85). Start by observing that the 2-norm of a product is not larger than the product of the 2-norms and that, as noted above, with $\mathbf{w}_t$ given the matrix $\hat{\mathbf{B}}_t^{-1}$ is also given to write

$$\mathbb{E}\left[\left\|\hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)\right\|^2\,\middle|\,\mathbf{w}_t\right] \;\leq\; \left\|\hat{\mathbf{B}}_t^{-1}\right\|^2\;\mathbb{E}\left[\left\|\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)\right\|^2\middle|\mathbf{w}_t\right] \tag{86}$$

Notice that, as stated in (32), $1/c$ is an upper bound for the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$. Further observe that the second moment of the norm of the stochastic gradient is bounded by $\mathbb{E}\left[\|\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)\|^2\,\middle|\,\mathbf{w}_t\right]\leq S^2$, as stated in Assumption 2. These two upper bounds substituted in (86) yield

$$\mathbb{E}\left[\left\|\hat{\mathbf{B}}_t^{-1}\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)\right\|^2\,\middle|\,\mathbf{w}_t\right] \leq \frac{S^2}{c^2}. \tag{87}$$

Substituting the upper bound in (87) for the third term of (85) and further using the fact that $\mathbb{E}\left[\hat{\mathbf{s}}(\mathbf{w}_t,\tilde{\boldsymbol{\theta}}_t)\,\middle|\,\mathbf{w}_t\right] = \nabla F(\mathbf{w}_t)$ in the second term leads to

$$\mathbb{E}\left[F(\mathbf{w}_{t+1})\,\middle|\,\mathbf{w}_t\right] \leq F(\mathbf{w}_t) - \epsilon_t\nabla F(\mathbf{w}_t)^T\hat{\mathbf{B}}_t^{-1}\nabla F(\mathbf{w}_t) + \frac{\epsilon_t^2 M S^2}{2c^2}. \tag{88}$$

We now find a lower bound for the second term in the right hand side of (88). As stated in (32), $1/C$ is a lower bound for the eigenvalues of $\hat{\mathbf{B}}_t^{-1}$. This lower bound implies that

$$\nabla F(\mathbf{w}_t)^T \hat{\mathbf{B}}_t^{-1} \nabla F(\mathbf{w}_t) \geq \frac{1}{C} \|\nabla F(\mathbf{w}_t)\|^2 \tag{89}$$

By substituting the lower bound in (89) for the corresponding summand in (88) the result in (33) follows.

Appendix F. Proof of Theorem 6

The proof uses the relationship in the statement (33) of Lemma 5 to build a supermartingale sequence. This is also a standard technique in stochastic optimization and provided here for reference. To construct the supermartingale sequence define the stochastic process α_t with values

$$\alpha_t := F(\mathbf{w}_t) + \frac{MS^2}{2c^2} \sum_{u=t}^{\infty} \epsilon_u^2. \tag{90}$$

Observe that α_t is well defined because the $\sum_{u=t}^{\infty} \epsilon_u^2 < \sum_{u=0}^{\infty} \epsilon_u^2 < \infty$ is summable. Further define the sequence β_t with values

$$\beta_t := \frac{\epsilon_t}{C} \|\nabla F(\mathbf{w}_t)\|^2. \tag{91}$$

Let now $\mathcal{F}_t$ be a sigma-algebra measuring α_t, β_t, and $\mathbf{w}_t$. The conditional expectation of α_{t+1} given $\mathcal{F}_t$ can be written as

$$\mathbb{E}\left[\alpha_{t+1} \,\middle|\, \mathcal{F}_t\right] = \mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathcal{F}_t\right] + \frac{MS^2}{2c^2} \sum_{u=t+1}^{\infty} \epsilon_u^2, \tag{92}$$

because the term $(MS^2/2c^2) \sum_{u=t+1}^{\infty} \epsilon_u^2$ is just a deterministic constant. Substituting (33) of Lemma 5 into (92) and using the definitions of α_t in (90) and β_t in (91) yields

$$\mathbb{E}\left[\alpha_{t+1} \,\middle|\, \alpha_t\right] \leq \alpha_t - \beta_t \tag{93}$$

Since the sequences α_t and β_t are nonnegative it follows from (93) that they satisfy the conditions of the supermartingale convergence theorem – see e.g. (Theorem E7.4 in Solo and Kong (1995)) . Therefore, we conclude that: (i) The sequence α_t converges almost surely. (ii) The sum $\sum_{t=0}^{\infty} \beta_t < \infty$ is almost surely finite. Using the explicit form of β_t in (91) we have that $\sum_{t=0}^{\infty} \beta_t < \infty$ is equivalent to

$$\sum_{t=0}^{\infty} \frac{\epsilon_t}{C} \|\nabla F(\mathbf{w}_t)\|^2 < \infty, \qquad \text{a.s.} \tag{94}$$

Since the sequence of stepsizes is nonsummable, for (94) to be true we need to have a vanishing subsequence embedded in $\|\nabla F(\mathbf{w}_t)\|^2$. By definition, this implies that the limit infimum of the sequence $\|\nabla F(\mathbf{w}_t)\|^2$ is null almost surely,

$$\liminf_{t \to \infty} \|\nabla F(\mathbf{w}_t)\|^2 = 0, \qquad \text{a.s.} \tag{95}$$

To transform the gradient bound in (95) into a bound pertaining to the squared distance to optimality $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ simply observe that the lower bound m on the eigenvalues of $\mathbf{H}(\mathbf{w}_t)$ applied to a Taylor's expansion around the optimal argument $\mathbf{w}^*$ implies that

$$F(\mathbf{w}^*) \geq F(\mathbf{w}_t) + \nabla F(\mathbf{w}_t)^T (\mathbf{w}^* - \mathbf{w}_t) + \frac{m}{2} \|\mathbf{w}^* - \mathbf{w}_t\|^2. \tag{96}$$

Observe now that since $\mathbf{w}^*$ is the minimizing argument of $F(\mathbf{w})$ we must have $F(\mathbf{w}^*) - F(\mathbf{w}_t) \leq 0$ for all $\mathbf{w}$. Using this fact and reordering terms we simplify (96) to

$$\frac{m}{2} \|\mathbf{w}^* - \mathbf{w}_t\|^2 \leq \nabla F(\mathbf{w}_t)^T (\mathbf{w}_t - \mathbf{w}^*). \tag{97}$$

Further observe that the Cauchy-Schwarz inequality implies that $\nabla F(\mathbf{w}_t)^T(\mathbf{w}_t - \mathbf{w}^*) \leq \|\nabla F(\mathbf{w}_t)\| \|\mathbf{w}_t - \mathbf{w}^*\|$. Substitution of this bound in (97) and simplification of a $\|\mathbf{w}^* - \mathbf{w}_t\|$ factor yields

$$\frac{m}{2} \|\mathbf{w}_t - \mathbf{w}^*\| \leq \|\nabla F(\mathbf{w}_t)\|. \tag{98}$$

Since the limit infimum of $\|\nabla F(\mathbf{w}_t)\|$ is null as stated in (95) the result in (34) follows from considering the bound in (98) in the limit as the iteration index $t \to \infty$.

Appendix G. Proof of Theorem 7

The proof follows along the lines of (Mokhtari and Ribeiro (2014a)) and is presented here for completeness. Theorem 7 claims that the sequence of expected objective values $\mathbb{E}[F(\mathbf{w}_t)]$ approaches the optimal objective $F(\mathbf{w}^*)$ at a linear rate $O(1/t)$. Before proceeding to the proof of Theorem 7 we repeat a technical lemma of (Mokhtari and Ribeiro (2014a)) that provides a sufficient condition for a sequence u_t to exhibit a linear convergence rate.

Lemma 8 (Mokhtari and Ribeiro (2014a)) *Let $a > 1$, $b > 0$ and $t_0 > 0$ be given constants and $u_t \geq 0$ be a nonnegative sequence that satisfies the inequality*

$$u_{t+1} \leq \left(1 - \frac{a}{t + t_0}\right) u_t + \frac{b}{(t + t_0)^2}, \tag{99}$$

for all times $t \geq 0$. The sequence u_t is then bounded as

$$u_t \leq \frac{Q}{t + t_0}, \tag{100}$$

for all times $t \geq 0$, where the constant Q is defined as

$$Q := \max\left[\frac{b}{a-1}, t_0 u_0\right]. \tag{101}$$

Proof We prove (100) using induction. To prove the claim for $t = 0$ simply observe that the definition of Q in (101) implies that

$$Q := \max\left[\frac{b}{a-1}, t_0 u_0\right] \geq t_0 u_0, \tag{102}$$

because the maximum of two numbers is at least equal to both of them. By rearranging the terms in (102) we can conclude that

$$u_0 \leq \frac{Q}{t_0}. \tag{103}$$

Comparing (103) and (100) it follows that the latter inequality is true for $t = 0$.

Introduce now the induction hypothesis that (100) is true for $t = s$. To show that this implies that (100) is also true for $t = s + 1$ substitute the induction hypothesis $u_s \leq Q/(s + t_0)$ into the recursive relationship in (99). This substitution shows that u_{s+1} is bounded as

$$u_{s+1} \leq \left(1 - \frac{a}{s + t_0}\right) \frac{Q}{s + t_0} + \frac{b}{(s + t_0)^2}. \tag{104}$$

Observe now that according to the definition of Q in (101), we know that $b/(a-1) \leq Q$ because Q is the maximum of $b/(a-1)$ and $t_0 u_0$. Reorder this bound to show that $b \leq Q(a-1)$ and substitute into (104) to write

$$u_{s+1} \leq \left(1 - \frac{a}{s+t_0}\right) \frac{Q}{s+t_0} + \frac{(a-1)Q}{(s+t_0)^2} \ . \tag{105}$$

Pulling out $Q/(s+t_0)^2$ as a common factor and simplifying and reordering terms it follows that (105) is equivalent to

$$u_{s+1} \ \leq \ \frac{Q\big[s+t_0-a+(a-1)\big]}{(s+t_0)^2} \ = \ \frac{s+t_0-1}{(s+t_0)^2}\, Q. \tag{106}$$

To complete the induction step use the difference of squares formula for $(s+t_0)^2 - 1$ to conclude that

$$\big[(s+t_0)-1\big]\big[(s+t_0)+1\big] \ = \ (s+t_0)^2 - 1 \ \leq \ (s+t_0)^2. \tag{107}$$

Reordering terms in (107) it follows that $\big[(s+t_0)-1\big]/(s+t_0)^2 \leq 1/\big[(s+t_0)+1\big]$, which upon substitution into (106) leads to the conclusion that

$$u_{s+1} \leq \frac{Q}{s+t_0+1}. \tag{108}$$

Eq. (108) implies that the assumed validity of (100) for $t = s$ implies the validity of (100) for $t = s+1$. Combined with the validity of (100) for $t = 0$, which was already proved, it follows that (100) is true for all times $t \geq 0$. ■

Lemma 8 shows that satisfying (99) is sufficient for a sequence to have the linear rate of convergence specified in (100). In the following proof of Theorem 7 we show that if the stepsize sequence parameters ϵ_0 and T_0 satisfy $2\epsilon_0 T_0/C > 1$ the sequence $\mathbb{E}\left[F(\mathbf{w}_t)\right] - F(\mathbf{w}^*)$ of expected optimality gaps satisfies (99) with $a = 2\epsilon_0 T_0/C$, $b = \epsilon_0^2 T_0^2 M S^2/2c^2$ and $t_0 = T_0$. The result in (35) then follows as a direct consequence of Lemma 8.

Proof of Theorem 7: Consider the result in (33) of Lemma 5 and subtract the average function optimal value $F(\mathbf{w}^*)$ from both sides of the inequality to conclude that the sequence of optimality gaps in the RES algorithm satisfies

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] - \ F(\mathbf{w}^*) \ \leq \ F(\mathbf{w}_t) - \ F(\mathbf{w}^*) - \frac{\epsilon_t}{C}\|\nabla F(\mathbf{w}_t)\|^2 + \frac{\epsilon_t^2 M S^2}{2c^2}. \tag{109}$$

We proceed to find a lower bound for the gradient norm $\|\nabla F(\mathbf{w}_t)\|$ in terms of the error of the objective value $F(\mathbf{w}_t) - \ F(\mathbf{w}^*)$ – this is a standard derivation which we include for completeness, see, e.g., Boyd and Vandenberghe (2004). As it follows from Assumption 1 the eigenvalues of the Hessian $\mathbf{H}(\mathbf{w}_t)$ are bounded between $0 < m$ and $M < \infty$ as stated in (25). Taking a Taylor's expansion of the objective function $F(\mathbf{y})$ around $\mathbf{w}$ and using the lower bound in the Hessian eigenvalues we can write

$$F(\mathbf{y}) \ \geq \ F(\mathbf{w}) + \nabla F(\mathbf{w})^T(\mathbf{y}-\mathbf{w}) + \frac{m}{2}\|\mathbf{y}-\mathbf{w}\|^2. \tag{110}$$

For fixed $\mathbf{w}$, the right hand side of (110) is a quadratic function of $\mathbf{y}$ whose minimum argument we can find by setting its gradient to zero. Doing this yields the minimizing argument $\hat{\mathbf{y}} = \mathbf{w} - (1/m)\nabla F(\mathbf{w})$ implying that for all $\mathbf{y}$ we must have

$$\begin{aligned} F(\mathbf{y}) \ &\geq \ F(\mathbf{w}) + \nabla F(\mathbf{w})^T(\hat{\mathbf{y}}-\mathbf{w}) + \frac{m}{2}\|\hat{\mathbf{y}}-\mathbf{w}\|^2 \\ &= \ F(\mathbf{w}) - \frac{1}{2m}\|\nabla F(\mathbf{w})\|^2. \end{aligned} \tag{111}$$

The bound in (111) is true for all $\mathbf{w}$ and $\mathbf{y}$. In particular, for $\mathbf{y} = \mathbf{w}^*$ and $\mathbf{w} = \mathbf{w}_t$ (111) yields

$$F(\mathbf{w}^*) \;\geq\; F(\mathbf{w}_t) - \frac{1}{2m}\|\nabla F(\mathbf{w}_t)\|^2. \tag{112}$$

Rearrange terms in (112) to obtain a bound on the gradient norm squared $\|\nabla F(\mathbf{w}_t)\|^2$. Further substitute the result in (109) and regroup terms to obtain the bound

$$\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right] - \; F(\mathbf{w}^*) \;\leq\; \left(1 - \frac{2m\epsilon_t}{C}\right)\left(F(\mathbf{w}_t) - \; F(\mathbf{w}^*)\right) + \frac{\epsilon_t^2 M S^2}{2c^2}. \tag{113}$$

Take now expected values on both sides of (113). The resulting double expectation in the left hand side simplifies to $\mathbb{E}\left[\mathbb{E}\left[F(\mathbf{w}_{t+1}) \,\middle|\, \mathbf{w}_t\right]\right] = \mathbb{E}\left[F(\mathbf{w}_{t+1})\right]$, which allow us to conclude that (113) implies that

$$\mathbb{E}\left[F(\mathbf{w}_{t+1})\right] - \; F(\mathbf{w}^*) \;\leq\; \left(1 - \frac{2m\epsilon_t}{C}\right)\left(\mathbb{E}\left[F(\mathbf{w}_t)\right] - \; F(\mathbf{w}^*)\right) + \frac{\epsilon_t^2 M S^2}{2c^2}. \tag{114}$$

Furhter substituting $\epsilon_t = \epsilon_0 T_0/(T_0 + t)$, which is the assumed form of the step size sequence by hypothesis, we can rewrite (114) as

$$\mathbb{E}\left[F(\mathbf{w}_{t+1})\right] - \; F(\mathbf{w}^*) \;\leq\; \left(1 - \frac{2m\epsilon_0 T_0}{(T_0 + t)C}\right)\left(\mathbb{E}\left[F(\mathbf{w}_t)\right] - F(\mathbf{w}^*)\right) + \left(\frac{\epsilon_0 T_0}{T_0 + t}\right)^2 \frac{M S^2}{2c^2}. \tag{115}$$

Given that the product $2m\epsilon_0 T_0/C > 1$ as per the hypothesis, the sequence $\mathbb{E}\left[F(\mathbf{w}_{t+1})\right] - \; F(\mathbf{w}^*)$ satisfies the hypotheses of Lemma 8 with $a = 2m\epsilon_0 T_0/C$, $b = \epsilon_0^2 T_0^2 M S^2/2c^2$. It then follows from (100) and (101) that (35) is true for the C_0 constant defined in (36) upon identifying u_t with $\mathbb{E}\left[F(\mathbf{x}_{t+1})\right] - \; F(\mathbf{x}^*)$, C_0 with Q, and substituting $c = 2m\epsilon_0 T_0/C$, $b = \epsilon_0^2 T_0^2 M S^2/2c^2$ and $t_0 = T_0$ for their explicit values. ■

References

J. R. Birge, X. Chen, L. Qi, and Z. Wei. A stochastic newton method for stochastic quadratic programs with resource. *Technical report*, University of Michigan, Ann Arbor, MI 1995.

A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research*, 10:1737–1754, 2009.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, ACM, 1992.

L. Bottou. Large-scale machine learning with stochastic gradient descent. *In Proceedings of COMPSTAT'2010*, pages 177–186, Physica-Verlag HD, 2010.

L. Bottou and Y. L. Cun. On-line learning for very large datasets. In *Applied Stochastic Models in Business and Industry*, volume 21. pp. 137-151, 2005.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, Cambridge, U.K, 1 edition, 2004.

C. G. Broyden, J. E. Dennis Jr., Wang, and J. J. More. On the local and superlinear convergence of quasi-newton methods. *IMA J. Appl. Math*, 12(3):223–245, June 1973.

R. H. Byrd, J. Nocedal, and Y. Yuan. Global convergence of a class of quasi-newton methods on convex problems. *SIAM J. Numer. Anal.*, 24(5):1171–1190, October 1987.

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

R. Fletcher. Practical methods of optimizations. John Wiley and Sons 2013.

Jr. J. E. Dennis and J. J. More. A characterization of super linear convergence and its application to quasi-newton methods. *Mathematics of computation*, 28(126):549–560, 1974.

J. Konecny and P. Richtarik. Semi-stochstic gradient descent methods. *arXiv preprint arXiv*, 1312.1666, 2013.

N. LeRoux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets. *arXiv preprint arXiv*, 1202.6258, 2012.

D. H. Li and M. Fukushima. A modified bfgs method and its global convergence in nonconvex minimization. *Journal of Computational and Applied Mathematics*, 129(1):15–35, 2001.

A. Mokhtari and A. Ribeiro. Res: Regularized stochastic bfgs algorithm. *arXiv preprint arXiv*, 1401.7625, 2014a.

A. Mokhtari and A. Ribeiro. A quasi-newton method for large scale support vector machines. In *Proc. Int. Conf. Acoustics Speech Signal Process.*, volume (to appear). Florence Italy, May 4-9 2014b.

A. Y. Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. *Proceedings of the twenty-first international conference on Machine learning*, page 78, 2004.

J. Nocedal and S. J. Wright. *Numerical optimization*. Springer-Verlag, New York, NY, 2 edition, 1999.

M. J. D. Powell. *Some global convergence properties of a variable metric algorithm for minimization without exact line search*. Academic Press, London, UK, 2 edition, 1971.

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

N. N. Schraudolph, J. Yu, and S. Gnter. A stochastic quasi-newton method for online convex optimization. *In Proc. 11th Intl. Conf. on Artificial Intelligence and Statistics (AIstats)*, page 433 440, Soc. for Artificial Intelligence and Statistics, 2007.

S. Shalev-Shwartz and N. Srebro. Svm optimization: inverse dependence on training set size. In *In Proceedings of the 25th international conference on Machine learning*. pp. 928-935, ACM 2008.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. *In Proceedings of the 24th international conference on Machine learning*, pages 807–814, ACM, 2007.

V. Solo and X. Kong. *Adaptive Signal Processing Algorithms: Stability and Performance*. NJ: Prentice-Hall, Englewood Cliffs, 1995.

G. Sun. Kdd cup track 2 soso.com ads prediction challenge, 2012. Accessed August 1 2012.

A. Ruszczynskiand W. Syski. Stochastic approximation method with gradient averaging for unconstrained problems. *IEEE Transactions on Automatic Control*, 28(12):1097–1105, 1983.

V. Vapnik. *The nature of statistical learning theory*. springer, 2 edition, 1999.

M. Zargham, A. Ribeiro, and A. Jadbabaie. Accelerated backpressure algorithm. *arXiv preprint arXiv*, 1302.1475, 2013.

L. Zhang, M. Mahdavi, and R. Jin. Linear convergence with condition number independent access of full gradients. *In Advances in Neural Information Processing Systems*, pages 980–988, 2013a.

T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. *In Proceedings of the twenty-first international conference on Machine learning*, page 919926, ACM, 2004.

Y. Zhang, J. C. Duchi, and M. J. Wainwright. Communication-efficient algorithms for statistical optimization. *The Journal of Machine Learning Research 14.1*, pages 3321–3363, 2013b.