Semistochastic quadratic bound methods

Aleksandr Aravkin
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
saravkin@us.ibm.com

Anna Choromanska
Columbia University
NY, USA
aec2163@columbia.edu

Tony Jebara
Columbia University
NY, USA
jebara@cs.columbia.edu

Dimitri Kanevsky
Google INC
76 Ninth Ave, NY 10011
dkanevsky@google.com

Abstract

Partition functions arise in a variety of settings, including conditional random fields, logistic regression, and latent gaussian models. In this paper, we consider semistochastic quadratic bound (SQB) methods for maximum likelihood inference based on partition function optimization. Batch methods based on the quadratic bound were recently proposed for this class of problems, and performed favorably in comparison to state-of-the-art techniques. Semistochastic methods fall in between batch algorithms, which use all the data, and stochastic gradient type methods, which use small random selections at each iteration. We build semistochastic quadratic bound-based methods, and prove both global convergence (to a stationary point) under very weak assumptions, and linear convergence rate under stronger assumptions on the objective. To make the proposed methods faster and more stable, we consider inexact subproblem minimization and batch-size selection schemes. The efficacy of SQB methods is demonstrated via comparison with several state-of-the-art techniques on commonly used datasets.

1 Introduction

The problem of optimizing a cost function expressed as the sum of a loss term over each sample in an input dataset is pervasive in machine learning. Standard learning systems based on batch methods such as Broyden-Fletcher-Goldfarb- Shanno(BFGS), quasi-Newton [1], full-rank and memory-limited (LBFGS), steepest descent (see e.g. [2]) or conjugate gradient [3] need to make a full pass through an entire dataset before updating the parameter vector. Even though these methods can converge quickly (sometimes in several passes through the dataset), as datasets grow in size, this learning strategy becomes increasingly inefficient. To faciliate learning on massive datasets, the community increasingly turns to stochastic methods.

Stochastic optimization methods interleave the update of parameters after only processing a small mini-batch of examples (potentially as small as a single data-point). They typically have low computational complexity per iteration ([4, 5, 6]). Furthermore, they often converge in significantly less iterations than their batch counterparts ([7, 8]) which results in addititional significant computational savings. Due to its simplicity and low computational cost, the most popular contemporary stochastic learning technique is stochastic gradient descent (SGD) [9, 4, 10]. SGD updates the parameter vector using the gradient of the objective function as evaluated on a single example (or, alternatively, a small mini-batch of examples). This algorithm admits multiple extensions, including (i) stochastic average gradient method (SAG) that averages the most recently computed gradients for each training example [11], (ii) methods that compute the (weighted) average of all previous gradients [12, 13], (iii) averaged stochastic gradient descent method (ASGD) that computes a running average of parameters obtained by SGD [14], (iv) stochastic dual coordinate ascent, that optimizes the dual objective with respect to a single dual vector or a mini-batch of dual vectors chosen uniformly at random [15, 16], (v) variance reduction techniques [17, 18, 11, 19] (some do not require storage of gradients, c.f. [17]), (vi) majorization-minimization techniques that minimize a majoring surrogate of an objective function [20, 21] and (vii) gain adaptation techniques [22, 23].

Semistochastic methods can be viewed as an interpolation between the expensive reliable updates used by full batch methods, and inexpensive noisy updates used by stochastic methods. They inherit the best of both worlds by approaching the solution more quickly when close to the optimum (like a full batch method) while simultaneously reducing the computational complexity per iteration (though less aggressively than stochastic methods). Several semistochastic extensions have been explored in previous works [24, 25, 26]. Recently, convergence theory and sampling strategies for these methods have been explored in [27, 28] and linked to results in finite sampling theory in [29].

Additionally, incorporating second-order information (i.e. Hessian) into the optimization problem ([22, 7, 30, 31, 32, 33]) was shown to often improve the performance of traditional SGD methods which typically provide fast improvement initially, but are slow to converge near the optimum (see e.g. [27]), require step-size tuning and are difficult to parallelize [34]. This paper focuses on semistochastic extension of a recently developed quadratic bound majorization technique [35], and we call the new algorithm *semistochastic quadratic bound* (SQB) method. The bound computes the update on the parameter vector using the product of the gradient of the objective function and an inverse of a second-order term that is a descriptor of the curvature of the objective function (different than the Hessian). We discuss implementation details, in particular curvature approximation, inexact solvers, and batch-size selection strategies, which make the running time of our algorithm comparable to the gradient methods and also make the method easily parallelizable. We show global convergence of the method to a stationary point under very weak assumptions (in particular convexity is not required) and a linear convergence rate when the size of the mini-batch grows sufficiently fast, following the techniques of [27]. This rate of convergence matches state-of-the-art incremental techniques [17, 15, 11, 20] (furthermore it is better than in case of standard stochastic gradient methods [9, 10] which typically have sublinear convergence rate [11, 36]). Compared to other existing majorization-minimization incremental techniques [20], our approach uses much tighter bounds which, as shown in [35], can lead to faster convergence.

The paper is organized as follows: Section 2 reviews quadratic bound majorization technique. Section 3 discusses stochastic and semistochastic extensions of the bound, and presents convergence theory for the proposed methods. In particular, we discuss very general stationary convergence theory under very weak assumptions, and also present a much stronger theory, including convergence rate analysis, for logistic regression. Section 4 discusses implementation details, and Section 5 shows numerical experiments illustrating the use of the proposed methods for l_2-regularized logistic regression problems. Conclusions end the paper.

2 Quadratic bound methods

Let Ω be a discrete probability space, and take any log-linear density model

$$p(y|x_j, \boldsymbol{\theta}) = \frac{1}{Z_{x_j}(\boldsymbol{\theta})} h_{x_j}(y) \exp\left(\boldsymbol{\theta}^\top \mathbf{f}_{x_j}(y)\right) \tag{1}$$

parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^d$, where $\{(x_1, y_1), \ldots, (x_T, y_T)\}$ are iid input-output pairs, $\mathbf{f}_{x_j} : \Omega \mapsto \mathbb{R}^d$ is a continuous vector-valued function mapping and $h_{x_j} : \Omega \mapsto \mathbb{R}^+$ is a fixed non-negative measure. The *partition function* $Z_{x_j}(\boldsymbol{\theta})$ is a scalar that ensures that $p(y|x_j, \boldsymbol{\theta})$ is a true density, so in particular (1) integrates to 1:

$$Z_{x_j}(\boldsymbol{\theta}) = \sum_y h_{x_j}(y) \exp(\boldsymbol{\theta}^\top \mathbf{f}_{x_j}(y)) \; . \tag{2}$$

[35] propose a fast method to find a tight quadratic bound for $Z_{x_j}(\boldsymbol{\theta})$, shown in the subroutine Bound Computation in Algorithm 1, which finds $z, \mathbf{r}, \mathbf{S}$ so that

$$Z_{x_j}(\boldsymbol{\theta}) \leq z \exp(\tfrac{1}{2}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^\top \mathbf{S}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}) + (\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^\top \mathbf{r}) \tag{3}$$

for any $\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}, \mathbf{f}_{x_j}(y) \in \mathbb{R}^d$ and $h_{x_j}(y) \in \mathbb{R}^+$ for all $y \in \Omega$.

The (regularized) maximum likelihood inference problem is equivalent to

$$\min_{\boldsymbol{\theta}}\left\{\mathcal{L}_\eta(\boldsymbol{\theta}) := -\frac{1}{T}\sum_{j=1}^{T}\log(p(y_j|x_j,\boldsymbol{\theta})) + \frac{\eta}{2}\|\boldsymbol{\theta}\|^2 \approx \frac{1}{T}\sum_{j=1}^{T}\left(\log(Z_{x_j}(\boldsymbol{\theta})) - \boldsymbol{\theta}^\top \mathbf{f}_{x_j}(y_j)\right) + \frac{\eta}{2}\|\boldsymbol{\theta}\|^2\right\}, \tag{4}$$

where $\approx$ means equal up to an additive constant. The bound (3) suggests the iterative minimization scheme

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha_k(\boldsymbol{\Sigma}^k + \eta \boldsymbol{I})^{-1}(\boldsymbol{\mu}^k + \eta\boldsymbol{\theta}^k). \tag{5}$$

where $\boldsymbol{\Sigma}^k$ and $\boldsymbol{\mu}^k$ are computed using Algorithm 1, η is the regularization term and α_k is the step size at iteration k.

In this paper, we consider applying the bound to randomly selected batches of data; any such selection we denote $\mathcal{T} \subset [1,\dots,T]$ or $\mathcal{S} \subset [1,\dots,T]$.

Algorithm 1 Semistochastic Quadratic Bound (SQB)

```
Input Parameters $\tilde{\boldsymbol{\theta}}, \mathbf{f}_{x_j}(y) \in \mathbb{R}^d$ and $h_{x_j}(y) \in \mathbb{R}^+$ for $y \in \Omega$, $j \in \mathcal{T}$
Initialize $\boldsymbol{\mu}_\mathcal{T} = \mathbf{0}, \boldsymbol{\Sigma}_\mathcal{T} = \mathbf{0}(d,d)$
For each $j \in \mathcal{T}$

    Subroutine **Bound Computation:**
    $z \to 0^+, \mathbf{r} = \mathbf{0}, \mathbf{S} = z\mathbf{I}$
    For each $y \in \Omega$
        $\alpha = h_{x_j}(y)\exp(\tilde{\boldsymbol{\theta}}^\top \mathbf{f}_{x_j}(y))$
        $\mathbf{S} += \frac{\tanh(\frac{1}{2}\log(\alpha/z))}{2\log(\alpha/z)}(\mathbf{f}_{x_j}(y) - \mathbf{r})(\mathbf{f}_{x_j}(y) - \mathbf{r})^\top$
        $\mathbf{r} = \frac{z}{z+\alpha}\mathbf{r} + \frac{\alpha}{z+\alpha}\mathbf{f}_{x_j}(y)$
        $z += \alpha$
    Subroutine output $z, \mathbf{r}, \mathbf{S}$

    $\boldsymbol{\mu}_\mathcal{T} += \mathbf{r} - \mathbf{f}_{x_j}(y)$
    $\boldsymbol{\Sigma}_\mathcal{T} += \mathbf{S}$
$\boldsymbol{\mu}_\mathcal{T} /= |\mathcal{T}|$
$\boldsymbol{\Sigma}_\mathcal{T} /= |\mathcal{T}|$
Output $\boldsymbol{\mu}_\mathcal{T}, \boldsymbol{\Sigma}_\mathcal{T}$
```

3 Stochastic and semistochastic extensions

The bounding method proposed in [35] is summarized in Algorithm 1 with $\mathcal{T} = [1,\dots,T]$ at every iteration. When T is large, this strategy can be expensive. In fact, computing the bound has complexity $O(Tnd^2)$, since Tn outer products must be summed to obtain $\boldsymbol{\Sigma}$, and each other product has complexity $O(d^2)$. When the dimension d is large, considerable speedups can be gained by obtaining a factored form of $\boldsymbol{\Sigma}$, as described in Section 4.1. Nonetheless, in either strategy, the size of T is a serious issue.

A natural approach is to subsample a smaller selection $\mathcal{T}$ from dataset Ω, so that at each iteration, we run Algorithm 1 over $\mathcal{T}$ rather than over the full data to get $\boldsymbol{\mu}_\mathcal{T}, \boldsymbol{\Sigma}_\mathcal{T}$. When $|\mathcal{T}|$ is fixed (and smaller than T), we refer to the resulting method as a *stochastic* extension. If instead $|\mathcal{T}|$ is allowed to grow as iterations proceed, we call this method *semistochastic*; these methods are analyzed in [27]. All of the numerical experiments we present focus on semistochastic methods. One can also decouple the computation of gradient and curvature approximations, using different data selections (which we call $\mathcal{T}$ and $\mathcal{S}$). We show that this development is theoretically justifiable and practically very useful.

For the stochastic and semistochastic methods discussed here, the quadratic bound property (3) does not hold for $Z(\boldsymbol{\theta})$, so the convergence analysis of [35] does not immediately apply. Nonetheless, it is possible to analyze the algorithm in terms of sampling strategies for $\mathcal{T}$.

The appeal of the stochastic modification is that when $|\mathcal{T}| << T$, the complexity $O(|\mathcal{T}|nd)$ of Algorithm 1 to compute $\boldsymbol{\mu}_\mathcal{T}, \boldsymbol{\Sigma}_\mathcal{S}$ is much lower; and then we can still implement a (modified) iteration (5). Intuitively, one expects that even small samples from the data can give good updates for the overall problem. This intuition is supported by the experimental results, which show that in terms of effective passes through the data, SQB is competitive with state of the art methods.

We now present the theoretical analysis of Algorithm 1. We first prove that under very weak assumption, in particular using only the Lipschitz property, but not requiring convexity of the problem, the proposed algorithm converges to a stationary point. The proof technique easily carries over to other objectives, such as the ones used in maximum latent conditional likelihood problems (for details see [35]), since it relies mainly only on the sampling method used to obtain $\mathcal{T}$. Then we focus on problem (4), which is convex, and strictly convex under appropriate assumptions on the data. We use the structure of (4) to prove much stronger results, and in particular analyze the rate of convergence of Algorithm 1.

3.1 General Convergence Theory

We present a general global convergence theory, that relies on the Lipschitz property of the objective and on the sampling strategy in the context of Algorithm 1. The end result we show here is that any limit point of the iterates is *stationary*. We begin with two simple preliminary results.

Lemma 1 *If every $i \in [1, \ldots, T]$ is equally likely to appear in $\mathcal{T}$, then $E[\boldsymbol{\mu}_{\mathcal{T}}] = \boldsymbol{\mu}$.*

Proof 1 *Algorithm 1 returns $\boldsymbol{\mu}_{\mathcal{T}} = \frac{1}{|\mathcal{T}|}\sum_{j\in\mathcal{T}} \psi_j(\boldsymbol{\theta})$, where $\psi_j(\boldsymbol{\theta}) = -\nabla_{\boldsymbol{\theta}} \log(p(y_j|x_j, \boldsymbol{\theta}))$. If each j has an equal chance to appear in $\mathcal{T}$, then*

$$E\left[\frac{1}{|\mathcal{T}|}\sum_{j\in\mathcal{T}} \psi_j(\boldsymbol{\theta})\right] = \frac{1}{|\mathcal{T}|}\sum_{j\in\mathcal{T}} E[\psi_j(\boldsymbol{\theta})] = \frac{1}{|\mathcal{T}|}\sum_{j\in\mathcal{T}} \boldsymbol{\mu} = \boldsymbol{\mu} \;.$$

Note that the hypothesis here is very weak: there is no stipulation that the batch size be of a certain size, grow with iterations, etc. This lemma therefore applies to a wide class of randomized bound methods.

Lemma 2 *Denote by $\lambda_{\min}$ the infimum over all possible eigenvalues of $\boldsymbol{\Sigma}_{\mathcal{S}}$ over all choices of batches ($\lambda_{\min}$ may be 0). Then $E[(\boldsymbol{\Sigma}_{\mathcal{S}} + \eta\boldsymbol{I})^{-1}]$ satisfies*

$$\frac{1}{\eta + \lambda_{\max}}\boldsymbol{I} \leq E[(\boldsymbol{\Sigma}_{\mathcal{S}} + \eta\boldsymbol{I})^{-1}] \leq \frac{1}{\eta + \lambda_{\min}}\boldsymbol{I} \;.$$

Proof 2 *For any vector $\boldsymbol{x}$ and any realization of $\boldsymbol{\Sigma}_{\mathcal{S}}$, we have*

$$\frac{1}{\eta + \lambda_{\max}}\|\boldsymbol{x}\|^2 \leq \boldsymbol{x}^T(\boldsymbol{\Sigma}_{\mathcal{S}} + \eta\boldsymbol{I})^{-1}\boldsymbol{x} \leq \frac{1}{\eta + \lambda_{\min}}\|\boldsymbol{x}\|^2 \;,$$

where $\lambda_{\max}$ depends on the data. Taking the expectation over $\mathcal{T}$ of be above inequality gives the result.

Theorem 1 *For any problem of form* (4)*, apply iteration* (5)*, where at each iteration $\boldsymbol{\mu}_{\mathcal{T}}, \boldsymbol{\Sigma}_{\mathcal{S}}$ are obtained by Algorithm 1 for two independently drawn batches subsets $\mathcal{T}, \mathcal{S} \subset [1, \ldots, T]$ selected to satisfy the assumptions of Lemma 1. Finally, suppose also that the step sizes α_k are square summable but not summable. Then $\mathcal{L}_\eta(\boldsymbol{\theta}^k)$ converges to a finite value, and $\nabla\mathcal{L}_\eta(\boldsymbol{\theta}^k) \to 0$. Furthermore, every limit point of $\boldsymbol{\theta}^k$ is a stationary point of $\mathcal{L}_\eta$.*

Theorem 1 states the conclusions of [37, Proposition 3], and so to prove it we need only check that the hypotheses of this proposition are satisfied.

Proof 3 *[37] consider algorithms of the form*

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha_k(\boldsymbol{s}^k + \boldsymbol{w}^k) \;.$$

In the context of iteration (5)*, at each iteration we have*

$$\boldsymbol{s}^k + \boldsymbol{w}^k = (\boldsymbol{\Sigma}_{\mathcal{S}}^k + \lambda\boldsymbol{I})^{-1}\boldsymbol{g}_{\mathcal{T}}^k,$$

where $\boldsymbol{g}_{\mathcal{T}}^k = \boldsymbol{\mu}_{\mathcal{T}}^k + \eta\boldsymbol{\theta}^k$, and $\boldsymbol{g}^k$ is the full gradient of the regularized problem (4)*. We choose*

$$\boldsymbol{s}^k \;\; = E[(\boldsymbol{\Sigma}_{\mathcal{S}}^k + \eta\boldsymbol{I})^{-1}]\boldsymbol{g}^k, \quad \boldsymbol{w}^k = (\boldsymbol{\Sigma}_{\mathcal{S}}^k + \eta\boldsymbol{I})^{-1}\boldsymbol{g}_{\mathcal{T}}^k - \boldsymbol{s}^k.$$

We now have the following results:

1. *Unbiased error:*
$$E[\boldsymbol{w}^k] = E[(\boldsymbol{\Sigma}_{\mathcal{S}}^k + \eta \boldsymbol{I})^{-1}\boldsymbol{g}_{\mathcal{T}}^k - \boldsymbol{s}^k] = E[(\boldsymbol{\Sigma}_{\mathcal{S}}^k + \eta \boldsymbol{I})^{-1}]E[\boldsymbol{g}_{\mathcal{T}}^k] - \boldsymbol{s}^k = 0\ , \tag{6}$$
where the second equality is obtained by independence of the batches $\mathcal{T}$ and $\mathcal{S}$, and the last equality uses Lemma 1.

2. *Gradient related condition:*
$$(\boldsymbol{g}^k)^T\boldsymbol{s}^k = (g^k)^T E[(\boldsymbol{\Sigma}_{\mathcal{S}}^k + \eta \boldsymbol{I})^{-1}]\boldsymbol{g}^k \quad \geq \frac{\|\boldsymbol{g}^k\|^2}{\eta + \lambda_{\max}}. \tag{7}$$

3. *Bounded direction:*
$$\|\boldsymbol{s}^k\| \leq \frac{\|\boldsymbol{g}^k\|}{\eta + \lambda_{\min}}. \tag{8}$$

4. *Bounded second moment:*

   By part 1, we have
$$\begin{aligned} E[\|\boldsymbol{w}^k\|^2] &\leq E[\|(\boldsymbol{\Sigma}_{\mathcal{S}}^k + \eta \boldsymbol{I})^{-1}\boldsymbol{g}_{\mathcal{T}}^k\|^2 \\ &\leq \frac{E[\|\boldsymbol{g}_{\mathcal{T}}^k\|^2]}{(\eta+\lambda_{\min})^2} = \frac{tr(cov[\boldsymbol{g}_{\mathcal{T}}^k]) + \|\boldsymbol{g}^k\|^2}{(\eta+\lambda_{\min})^2}. \end{aligned} \tag{9}$$

The covariance matrix of $\boldsymbol{g}_{\mathcal{T}}^k$ is proportional to the covariance matrix of the set of individual (data-point based) gradient contributions, and for problems of form (4) *these contributions lie in the convex hull of the data, so in particular the trace of the covariance must be finite. Taken together, these results show all hypotheses of [37, Proposition 3] are satisfied, and the result follows.*

Theorem (1) applies to any stochastic and semistochastic variant of the method. Note that two independent data samples $\mathcal{T}$ and $\mathcal{S}$ are required to prove (6). Computational complexity motivates different strategies for selecting choose $\mathcal{T}$ and $\mathcal{S}$. In particular, it is natural to use larger mini-batches to estimate the gradient, and smaller mini-batch sizes for the estimation of the second-order curvature term. Algorithms of this kind have been explored in the context of stochastic Hessian methods [33]. We describe our implementation details in Section 4.

3.2 Rates of Convergence for Logistic Regression

The structure of objective (4) allows for a much stronger convergence theory. We first present a lemma characterizing strong convexity and Lipschitz constant for (4). Both of these properties are crucial to the convergence theory.

Lemma 3 *The objective $\mathcal{L}_\eta$ in* (4) *has a gradient that is uniformly norm bounded, and Lipschitz continuous.*

Proof 4 *The function $\mathcal{L}_\eta$ has a Lipschitz continuous gradient if there exists an L such that*
$$\|\nabla\mathcal{L}_\eta(\boldsymbol{\theta}^1) - \nabla\mathcal{L}_\eta(\boldsymbol{\theta}^0)\| \leq L\|\boldsymbol{\theta}^1 - \boldsymbol{\theta}^0\|$$
holds for all $(\boldsymbol{\theta}^1, \boldsymbol{\theta}^0)$. Any uniform bound for trace$(\nabla^2\mathcal{L}_\eta)$ is a Lipschitz bound for $\nabla\mathcal{L}_\eta$. Define
$$a_{y,j} := h_{x_j}(y)\exp(\boldsymbol{\theta}^\top \mathbf{f}_{x_j}(y))\ ,$$
and note $a_{y,j} \geq 0$. Let $\boldsymbol{p}_j$ be the empirical density where the probability of observing y is given by $\frac{a_{y,j}}{\sum_y a_{y,j}}$. The gradient of (4) *is given by*
$$\frac{1}{T}\sum_{j=1}^{T}\left(\left(\sum_y \frac{a_{y,j}\mathbf{f}_{x_j}(y)}{\sum_y a_{y,j}}\right) - \mathbf{f}_{x_j}(y_j)\right) + \eta\boldsymbol{\theta} = \frac{1}{T}\sum_{j=1}^{T}\left(E_{\boldsymbol{p}_j}[\mathbf{f}_{x_j}(\cdot)] - \mathbf{f}_{x_j}(y_j)\right) + \eta\boldsymbol{\theta} \tag{10}$$
It is straightforward to check that the Hessian is given by
$$\nabla^2\mathcal{L}_\eta = \frac{1}{T}\sum_{j=1}^{T} cov_{\boldsymbol{p}_j}[\mathbf{f}_{x_j}(\cdot)] + \eta\boldsymbol{I} \tag{11}$$

where $\mathrm{cov}_{\mathbf{p}_j}[\cdot]$ *denotes the covariance matrix with respect to the empirical density function* $\mathbf{p}_j$*. Therefore a global bound for the Lipschitz constant* L *is given by* $\max_{y,j}\|\mathbf{f}_{x_j}(y)\|^2 + \eta\boldsymbol{I}$*, which completes the proof.*

Corollary 1 *The function* $\mathcal{L}_0$ *is strongly convex exactly when* $\sum_{j,y}\mathbf{f}_{x_j}(y)\mathbf{f}_{x_j}(y)^T$ *is positive definite, and* $\mathcal{L}_\eta$ *is strongly convex for any positive* η*.*

We now present a convergence rate result, using results from [27, Theorem 2.2].

Theorem 2 *There exist* $\mu, L > 0$ *such that*

$$\begin{aligned}\|\nabla\mathcal{L}_\eta(\boldsymbol{\theta}_1) - \nabla\mathcal{L}_\eta(\boldsymbol{\theta}_2)\|_{**} &\le L\|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\|_* \\ \mathcal{L}_\eta(\boldsymbol{\theta}_2) \ge \mathcal{L}_\eta(\boldsymbol{\theta}_1) + (\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)^T\nabla\mathcal{L}_\eta(\boldsymbol{\theta}_1) &+ \frac{1}{2}\rho\|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\|_*\end{aligned} \tag{12}$$

where $\|\boldsymbol{\theta}\|_* = \sqrt{\boldsymbol{\theta}^T(\boldsymbol{\Sigma}_{\mathcal{S}}^k + \eta\boldsymbol{I})\boldsymbol{\theta}}$ *and* $\|\boldsymbol{\theta}\|_{**}$ *is the corresponding dual norm* $\sqrt{\boldsymbol{\theta}^T(\boldsymbol{\Sigma}_{\mathcal{S}}^k + \eta\boldsymbol{I})^{-1}\boldsymbol{\theta}}$*. Furthermore, take* $\alpha_k = \frac{1}{L}$ *in* (5)*, and define* $B_k = \|\nabla\mathcal{L}_\eta^k - \boldsymbol{g}_{\mathcal{T}}^k\|^2$*, the square error incurred in the gradient at iteration* k*. Provided a batch growth schedule with* $\lim_{k\to\infty}\frac{B_{k+1}}{B_k} \le 1$*, for each iteration* (5) *we have (for any* $\epsilon > 0$*)*

$$\mathcal{L}_\eta(\boldsymbol{\theta}^k) - \mathcal{L}_\eta(\boldsymbol{\theta}^*) \le \left(1 - \frac{\rho}{L}\right)^k[\mathcal{L}_\eta(\boldsymbol{\theta}^0) - \mathcal{L}_\eta(\boldsymbol{\theta}^*)] + \mathcal{O}(C_k)\ , \tag{13}$$

with $C_k = \max\{B_k, (1 - \frac{\rho}{L} + \epsilon)^k\}$*.*

Proof 5 *Let* $\tilde{L}$ *denote the bound on the Lipschitz constant of* g *is provided in* (10)*. By the conclusions of Lemma 2, we can take* $L = \frac{1}{\sqrt{\eta+\lambda_{\min}}}\tilde{L}$*. Let* $\tilde{\rho}$ *denote the minimum eigenvalue of* (11) *(note that* $\tilde{\rho} \ge \eta$*). Then take* $\rho = \frac{1}{\sqrt{\eta+\lambda_{\max}}}\tilde{\rho}$*. The result follows immediately by [27, Theorem 2.2].*

4 Implementation details

In this section, we briefly discuss important implementation details as well as describe the comparator methods we use for our algorithm.

4.1 Efficient inexact solvers

The linear system we have to invert in iteration (5) has very special structure. The matrix $\boldsymbol{\Sigma}$ returned by Algorithm 1 may be written as $\boldsymbol{\Sigma} = \boldsymbol{S}\boldsymbol{S}^T$, where each column of $\boldsymbol{S}$ is proportional to one of the vectors $(\mathbf{f}_{x_j}(y) - \mathbf{r})$ computed by the bound. When the dimensions of $\boldsymbol{\theta}$ are large, it is not practical to compute the $\boldsymbol{\Sigma}$ explicitly. Instead, to compute the update in iteration (5), we take advantage of the fact that

$$\boldsymbol{\Sigma}\boldsymbol{x} = \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{x}),$$

and use $\boldsymbol{S}$ (computed with a simple modification to the bound method) to implement the action of $\boldsymbol{\Sigma}$. When $\boldsymbol{S} \in \mathbb{R}^{d\times k}$ (k is a mini-batch size), the action of the transpose on a vector can be computed in $O(dk)$, which is very efficient for small k. The action of the regularized curvature approximation $\boldsymbol{\Sigma} + \eta\boldsymbol{I}$ follows immediately. Therefore, it is efficient to use iterative minimization schemes, such as lsqr, conjugate gradient, or others to compute the updates. Moreover, using only a few iterations of these methods further regularizes the subproblems [38].

It is interesting to note that even when $\eta = 0$, and $\boldsymbol{\Sigma}_{\mathcal{T}}$ is not invertible, it makes sense to consider inexact updates. To justify this approach, we first present a range lemma.

Lemma 4 *For any* $\mathcal{T}$*, we have* $\boldsymbol{\mu}_{\mathcal{T}} \in \mathcal{R}(\boldsymbol{\Sigma}_{\mathcal{T}})$*.*

Proof 6 *The matrix* $\boldsymbol{\Sigma}_{\mathcal{T}}$ *is formed by a sum of weighted outer products* $(\mathbf{f}_{x_j}(y) - \boldsymbol{r})(\mathbf{f}_{x_j}(y) - \boldsymbol{r})^\top$*. We can therefore write*

$$\boldsymbol{\Sigma}_{\mathcal{T}} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$$

where $\boldsymbol{L} = [\boldsymbol{l}_1, \ldots, \boldsymbol{l}_{|\Omega|\cdot|\mathcal{T}|}]$, $\boldsymbol{l}_k = \mathbf{f}_{x_j}(y_k) - \boldsymbol{r}^k$ (k is the current iteration of the bound computation), and $\boldsymbol{D}$ is a diagonal matrix with weights $\boldsymbol{D}_{kk} = \frac{1}{|\mathcal{T}|}\frac{\tanh(\frac{1}{2}\log(\alpha_k/z_k))}{2\log(\alpha_k/z_k)}$, where the quantities α_k, z_k correspond to iterations in Algorithm (1). Since $\boldsymbol{\mu}$ is in the range of $\boldsymbol{L}$ by construction, it must also be the range of $\boldsymbol{\Sigma}_{\mathcal{T}}$.

Lemma 4 tells us that there is always a solution to the linear system $\boldsymbol{\Sigma}_{\mathcal{T}}\boldsymbol{\Delta}\theta = \boldsymbol{\mu}_{\mathcal{T}}$, even if $\boldsymbol{\Sigma}_{\mathcal{T}}$ is singular. In particular, a minimum norm solution can be found using the Moore-Penrose pseudoinverse, or by simply applying lsqr or cg, which is useful in practice when the dimension d is large. For many problems, using a small number of cg iterations both speeds up the algorithm and serves as additional regularization at the earlier iterations, since the (highly variable) initially small problems are not fully solved.

4.2 Mini-batches selection scheme

In our experiments, we use a simple linear interpolation scheme to grow the batch sizes for both the gradient and curvature term approximations. In particular, each batch size (as a function of iteration k) is given by

$$b^k = \min(b^{\text{cap}}, b^1 + \mathsf{round}((k-1)\gamma)),$$

where b^{cap} represents the cap on the maximum allowed size, b^1 is the initial batch size, and γ gives the rate of increase. In order to specify the selections chosen, we will simply give values for each of $(b^1_{\boldsymbol{\mu}}, b^1_{\boldsymbol{\Sigma}}, \gamma_{\boldsymbol{\mu}}, \gamma_{\boldsymbol{\Sigma}})$. For all experiments, the cap $b^{\text{cap}}_{\boldsymbol{\mu}}$ on the gradient computation was the full training set, the cap $b^{\text{cap}}_{\boldsymbol{\Sigma}}$ for the curvature term was taken to be 200, initial $b^1_{\boldsymbol{\mu}}$ and $b^1_{\boldsymbol{\Sigma}}$ were both set to 5. At each iteration of SQB, the parameter vector is updated as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\xi}^k,$$

where $\boldsymbol{\xi}^k = \alpha(\boldsymbol{\Sigma}^k_{\mathcal{S}} + \eta\boldsymbol{I})^{-1}(\boldsymbol{\mu}^k_{\mathcal{T}} + \eta\boldsymbol{\theta}^k)$ (α is the step size; we use constant step size for SQB in our experiments). Notice that $\boldsymbol{\xi}^k$ is the solution to the linear system $(\boldsymbol{\Sigma}^k_{\mathcal{S}} + \eta\boldsymbol{I})\boldsymbol{\xi}^k = \boldsymbol{\mu}^k_{\mathcal{T}} + \eta\boldsymbol{\theta}^k$ and can be efficiently computed using the lsqr solver (or any other iterative solver). For all experiments, we ran a small number (l) iterations of lsqr, where l was chosen from the set $\{5, 10, 20\}$, before updating the parameter vector (this technique may be viewed as performing conjugate gradient on the bound), and chose l with the best performance (the fastest and most stable convergence).

4.3 Step size

One of the most significant disadvantages of standard stochastic gradient methods [9, 10] is the choice of the step size. Stochastic gradient algorithms can achieve dramatic convergence rate if the step size is badly tuned [39, 11]. An advantage of computing updates using approximated curvature terms is that the inversion also establishes a scale for the problem, and requires minimal tuning. This is well known (phenomenologically) in inverse problems. In all experiments below, we used a constant step size; for well conditioned examples we used step size of 1, and otherwise 0.1.

4.4 Comparator methods

We compared SQB method with the variety of competitive state-of-the-art methods which we list below:

- **L-BFGS**: limited-memory BFGS method (quasi-Newton method) tuned for log-linear models which uses both first- and second-order information about the objective function (for L-BFGS this is gradient and approximation to the Hessian); we use the competitive implementation obtained from http://www.di.ens.fr/ mschmidt/Software/minFunc.html
- **SGD**: stochastic gradient descent method with constant step size; we use the competitive implementation obtained from http://www.di.ens.fr/ mschmidt/Software/SAG.html which is analogous to L. Bottou implementation but with pre-specified step size
- **ASGD**: averaged stochastic gradient descent method with constant step size; we use the competitive implementation obtained from http://www.di.ens.fr/ mschmidt/Software/SAG.html which is analogous to L. Bottou implementation but with pre-specified step size

- **SAG**: stochastic average gradient method using the estimate of Lipschitz constant L_k at iteration k set constant to the global Lipschitz constant; we use the competitive implementation of [11] obtained from http://www.di.ens.fr/ mschmidt/Software/SAG.html
- **SAGls**: stochastic average gradient method with line search, we use the competitive implementation of [11] obtained from http://www.di.ens.fr/ mschmidt/Software/SAG.html; the algorithm adaptively estimates Lipschitz constant L with respect to the logistic loss function using line-search

  Since our method uses the constant step size we chose to use the same scheme for the competitor methods like SGD, ASGD and SAG. For those methods we tuned the step size to achieve the best performance (the fastest and most stable convergence). Remaining comparators (L-BFGS and SAGls) use line-search.

5 Experiments

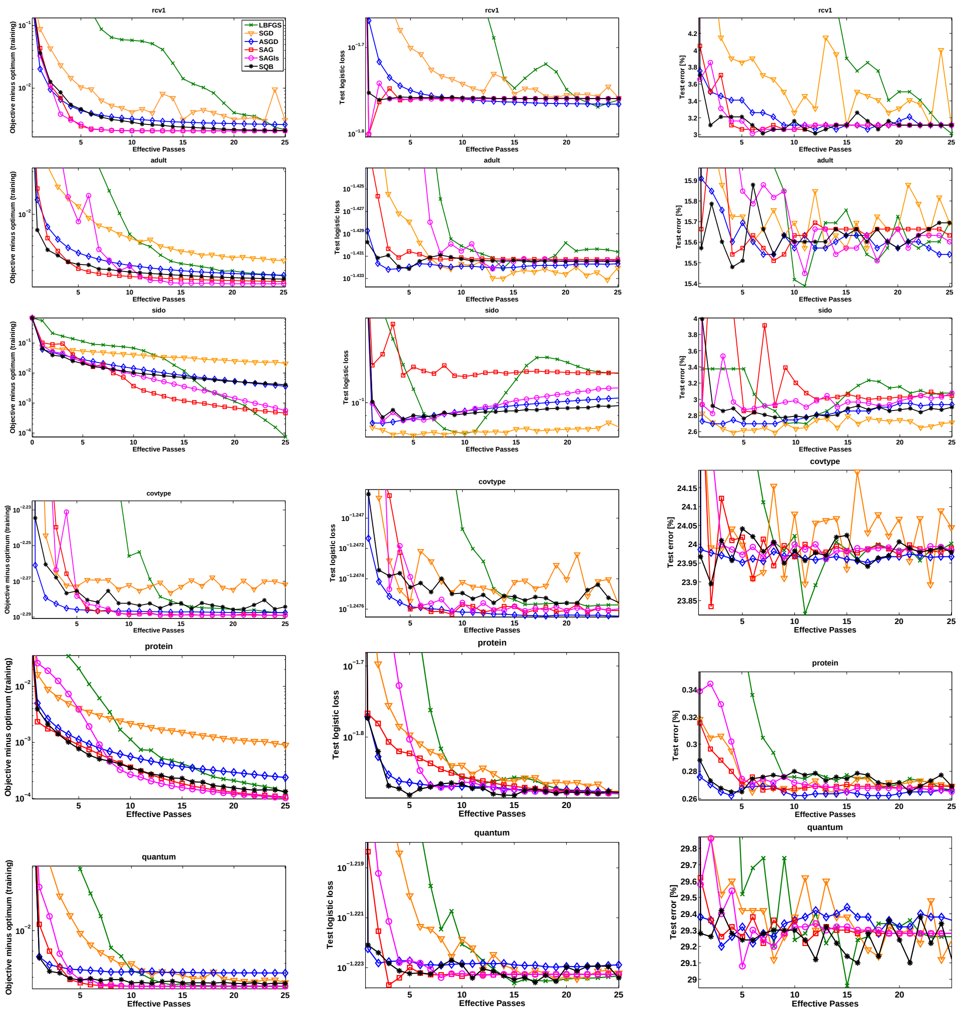

Figure 1: Comparison of optimization strategies for l_2-regularized logistic regression. From left two right: training excess cost, testing cost and testing error. From top to bottom: *rcv1* ($\alpha_{\text{SGD}} = 10^{-1}$, $\alpha_{\text{ASGD}} = 1$, $\alpha_{\text{SQB}} = 10^{-1}$), *adult* ($\alpha_{\text{SGD}} = 10^{-3}$, $\alpha_{\text{ASGD}} = 10^{-2}$, $\alpha_{\text{SQB}} = 1$), *sido* ($\alpha_{\text{SGD}} = 10^{-3}$, $\alpha_{\text{ASGD}} = 10^{-2}$, $\alpha_{\text{SQB}} = 1$), *covtype* ($\alpha_{\text{SGD}} = 10^{-4}$, $\alpha_{\text{ASGD}} = 10^{-3}$, $\alpha_{\text{SQB}} = 10^{-1}$), *protein* ($\alpha_{\text{SGD}} = 10^{-3}$, $\alpha_{\text{ASGD}} = 10^{-2}$, $\alpha_{\text{SQB}} = 1$) and *quantum* ($\alpha_{\text{SGD}} = 10^{-4}$, $\alpha_{\text{ASGD}} = 10^{-2}$, $\alpha_{\text{SQB}} = 10^{-1}$) datasets. This figure is best viewed in color.

We performed experiments with l_2-regularized logistic regression on binary classification task with regularization parameter $\eta = \frac{1}{T}$. We report the results for six datasets. The first three are sparse:

rcv1 ($T = 20242$, $d = 47236$; SQB parameters: $l = 5$, $\gamma_{\boldsymbol{\mu}} = 0.005$, $\gamma_{\boldsymbol{\Sigma}} = 0.0003$), *adult* ($T = 32561$, $d = 123$; SQB parameters: $l = 5$, $\gamma_{\boldsymbol{\mu}} = 0.05$, $\gamma_{\boldsymbol{\Sigma}} = 0.001$) and *sido* ($T = 12678$, $d = 4932$; SQB parameters: $l = 5$, $\gamma_{\boldsymbol{\mu}} = 0.01$, $\gamma_{\boldsymbol{\Sigma}} = 0.0008$). The remaining datasets are dense: *covtype* ($T = 581012$, $d = 54$; SQB parameters: $l = 10$, $\gamma_{\boldsymbol{\mu}} = 0.0005$, $\gamma_{\boldsymbol{\Sigma}} = 0.0003$), *protein* ($T = 145751$, $d = 74$; SQB parameters: $l = 20$, $\gamma_{\boldsymbol{\mu}} = 0.005$, $\gamma_{\boldsymbol{\Sigma}} = 0.001$) and *quantum* ($T = 50000$, $d = 78$; SQB parameters: $l = 5$, $\gamma_{\boldsymbol{\mu}} = 0.001$, $\gamma_{\boldsymbol{\Sigma}} = 0.0008$). Each dataset was split to training and testing datasets such that 90% of the original datasets was used for training and the remaining part for testing. Only *sido* and *protein* were split in half to training and testing datasets due to large disproportion of the number of datapoints belonging to each class. The experimental results we obtained are shown in Figure 1. We report the training and testing costs as well as the testing error as a function of the number of effective passes through the data and thus the results do not rely on the implementation details. We would like to emphasize however that under current implementation the average running time for the bound method across the datasets is comparable to that of the competitor methods. All codes are released and are publicly available at www.columbia.edu/ aec2163/NonFlash/Papers/Papers.html.

6 Conclusions

We have presented a new semistochastic quadratic bound (SQB) method, together with convergence theory and several numerical examples. The convergence theory is divided into two parts. First, we proved convergence to stationarity of the method under weak hypotheses (in particular, convexity is not required). Second, for the logistic regression problem, we provided a stronger convergence theory, including a rate of convergence analysis.

The main contribution of this paper is to apply SQB methods in a semi-stochastic large-scale setting. In particular, we developed and analyzed a flexible framework that allows sample-based approximations of the bound from [35] that are appropriate in the large-scale setting, computationally efficient, and competitive with state-of-the-art methods.

References

[1] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research, 1999.

[2] Y. Nesterov. *Introductory lectures on convex optimization : a basic course*. Applied optimization. Kluwer Academic Publ., Boston, Dordrecht, London, 2004.

[3] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[4] L. Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998.

[5] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2(4):285–318, April 1988.

[6] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[7] N. N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-newton method for online convex optimization. In *AISTATS*, 2007.

[8] S. V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML*, 2006.

[9] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[10] L. Bottou and Y. LeCun. Large scale online learning. In *NIPS*, 2003.

[11] N. Le Roux, M. W. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, 2012.

[12] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Math. Program.*, 120(1):221–259, 2009.

[13] P. Tseng. An incremental gradient(-projection) method with momentum term and adaptive stepsize rule. *SIAM J. on Optimization*, 8(2):506–531, February 1998.

[14] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, July 1992.

[15] S. Shalev-Shwartz and T. Zhang. Proximal stochastic dual coordinate ascent. *CoRR*, abs/1211.2717, 2012.

[16] S. Shalev-Shwartz and T. Zhang. Accelerated mini-batch stochastic dual coordinate ascent. *CoRR*, abs/1305.2581, 2013.

[17] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *NIPS*, pages 315–323. 2013.

[18] C. Wang, X. Chen, A. Smola, and E. Xing. Variance reduction for stochastic gradient optimization. In *NIPS*, pages 181–189. 2013.

[19] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *CoRR*, abs/1209.1873, 2012.

[20] J. Mairal. Optimization with first-order surrogate functions. In *ICML (3)*, pages 783–791, 2013.

[21] J. Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *NIPS*, pages 2283–2291. 2013.

[22] N. N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *ICANN*, 1999.

[23] N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14:2002, 2002.

[24] A. Shaprio Y. Wardi. Convergence analysis of stochastic algorithms. *Mathematics of Operations Research*, 21(3):615–628, 1996.

[25] Alexander Shapiro, Tito Homem de mello, and Pii S. On the rate of convergence of optimal solutions of monte carlo approximations of stochastic programs. *SIAM Journal on Optimization*, 11:70–86, 2000.

[26] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. on Optimization*, 12(2):479–502, February 2002.

[27] M. P. Friedlander and M. Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM J. Scientific Computing*, 34(3), 2012.

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

[29] A. Aravkin, M. P. Friedlander, F. Herrmann, and T. van Leeuwen. Robust inversion, dimensionality reduction, and randomized sampling. *Mathematical Programming*, 134(1):101–125, 2012.

[30] Y. Le Cun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science LNCS 1524. Springer Verlag, 1998.

[31] A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *J. Mach. Learn. Res.*, 10:1737–1754, December 2009.

[32] J. Martens. Deep learning via hessian-free optimization. In *ICML*, 2010.

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

[34] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng. On optimization methods for deep learning. In *ICML*, 2011.

[35] T. Jebara and A. Choromanska. Majorization for CRFs and latent likelihoods. In *NIPS*, 2012.

[36] A. Agarwal, P. L. Bartlett, P. D. Ravikumar, and M. J. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Transactions on Information Theory*, (5):3235–3249.

[37] D. P. Bertsekas and J. N. Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM J. on Optimization*, 10(3):627–642, July 1999.

[38] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[39] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. on Optimization*, 19(4):1574–1609, January 2009.