STOCHASTIC STEFFENSEN METHOD

MINDA ZHAO, ZEHUA LAI, AND LEK-HENG LIM

Abstract. Is it possible for a first-order method, i.e., only first derivatives allowed, to be quadratically convergent? For univariate loss functions, the answer is yes — the *Steffensen method* avoids second derivatives and is still quadratically convergent like Newton method. By incorporating a specific step size we can even push its convergence order beyond quadratic to $1 + \sqrt{2} \approx 2.414$. While such high convergence orders are a pointless overkill for a deterministic algorithm, they become rewarding when the algorithm is randomized for problems of massive sizes, as randomization invariably compromises convergence speed. We will introduce two adaptive learning rates inspired by the Steffensen method, intended for use in a stochastic optimization setting and requires no hyperparameter tuning aside from batch size. Extensive experiments show that they compare favorably with several existing first-order methods. When restricted to a quadratic objective, our stochastic Steffensen methods reduce to randomized Kaczmarz method — note that this is not true for SGD or SLBFGS — and thus we may also view our methods as a generalization of randomized Kaczmarz to arbitrary objectives.

1. Introduction

In minimizing a *univariate* function f with an iteration $x_{k+1} = x_k - f'(x_k)/g(x_k)$, possibilities for g include

$$\text{gradient:} \quad g(x_k) = 1, \qquad \text{Newton:} \quad g(x_k) = f''(x_k),$$
$$\text{secant:} \quad g(x_k) = \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}, \qquad \text{Steffensen:} \quad g(x_k) = \frac{f'(x_k + f'(x_k)) - f'(x_k)}{f'(x_k)},$$

with different orders of convergence q, i.e., $|x_{k+1} - x^*| \leq c|x_k - x^*|^q$. Gradient descent has $q = 1$, secant method $q = (1 + \sqrt{5})/2$, Newton and Steffensen methods both have $q = 2$.

Steffensen method [41, 42] is a surprise. Not only does it not require second derivatives (like Newton) to achieve quadratic convergence, it also does not achieve its superior convergence through the use of multisteps (like secant). In other words, the kth Steffensen iterate only depends on x_k but not x_{k-1}, x_{k-2}, etc.

Nevertheless, while the other three methods have widely used multivariate generalizations (secant method has several, as quasi-Newton methods, as Barzilai–Borwein step size, etc), all existing multivariate generalizations of Steffensen method [1, 9, 13, 15, 16, 22, 25, 26, 28, 29, 30, 31, 32] involve multivariate *divided differences* that require $O(n^2)$ function evaluations and are no less expensive than using the full Hessian. Furthermore these multivariate generalizations are no longer one-step methods. As a result they have not found widespread use.

Our contributions are as follows:

(i) We show that by incorporating a carefully chosen step size parameter the convergence of Steffensen method may be further improved beyond quadratic to $q = 1 + \sqrt{2}$.
(ii) We extend Steffensen method to a multivariate setting as an adaptive learning rate, avoiding divided differences, requiring just two gradient evaluations, and remaining a one-step method.
(iii) We show that when used in a randomized setting, our methods outperform SGD, SVRG, and SLBFGS on a variety of standard machine learning tasks on real data sets.

The performance in (iii) is measured in actual running time. But aside from speed, our methods have two advantages over SLBFGS, which has become a gold standard in machine learning:

(a) Quasi-Newton methods may involve matrix-vector product, a two-loop recursion with $O(d^2)$ computation. Although deterministic LBFGS does not form matrix-vector product explicitly, stochastic LBFGS does. Our multivariate Steffensen methods, whether deterministic or stochastic, are free of such products.

(b) Quasi-Newton methods come in two flavors: Hessian or inverse Hessian updates. The latter seems a nobrainer as it avoids matrix inversion but this is a fallacy. It is common knowledge among practitioners [11, Section 4.5.2.2] that the inverse Hessian version often conceals an ill-conditioned approximate Hessian; one should instead update the Cholesky factors of the approximate Hessian in order to detect ill-conditioning. By its design, LBFGS inevitably uses the inverse Hessian version. Our multivariate Steffensen methods are not quasi-Newton methods and do not involve approximate Hessians, avoiding this issue entirely.

A theoretical counterpart to (iii) is that when measured by the number of stochastic gradient evaluations, we have the following complexity estimates:

$$\text{SSM/SSBB:} \qquad O\big((n+\kappa^2)\log(1/\varepsilon)\big),$$

$$\text{SVRG–BB:} \qquad O\big((n+\kappa^3)\log(1/\varepsilon)\big),$$

$$\text{SLBFGS:} \qquad O\big((n+\kappa^{2+2(d+h)})\log(1/\varepsilon)\big),$$

to minimize a d-variate convex function of the form $f = f_1 + \cdots + f_n$ to ε-accuracy. So our proposed methods SSM and SSBB are at least an order of magnitude faster than SVRG–BB and SLBFGS in terms of the condition number κ. Here h refers to the 'history size' of SLBFGS. The algorithms and quantities involved will all be explained in due course.

Johan Steffensen first proposed his eponymous method [41] in 1933. See [4] for an informative history of the method and a biography of its inventor. The method was described in the classic books of Henrici [13, pp. 91–95] and Householder [14, p. 164] but has remained more of a textbook curiosity. One reason, as we mentioned above and will elaborate in Section 2.2, is that there has been no viable multivariate version.

Another reason, as we will speculate, is that much like the Kaczmarz method [18, 19] for iterative solution of linear systems had lingered in relative obscurity until it was randomized [43], Steffensen method is also most effective in a randomized setting. This is in fact more than an analogy; we will show in Section 2.4 that the stochastic Steffensen method we propose reduces to randomized Kaczmarz method when applied to a quadratic objective — not true for SGD, SVRG, or SLBFGS. So one may also view our stochastic Steffensen method as a generalization of randomized Kaczmarz method to arbitrary differentiable objective functions.

In Section 3 we will supply proofs of linear convergence of our methods and a theoretical comparison with other existing methods. In Section 4 we show how to adapt our methods for nonsmooth functions. In the numerical experiments in Section 5, we will see that our stochastic Steffensen method compares favorably with SGD, SVRG (with or without Barzilai–Borwein step size), and SLBFGS across different tasks in the `LIBSVM` datasets: ridge regression, logistic regression, and support vector machines with squared hinge loss.

Background. As in the usual setting for stochastic gradient descent and its variants, our goal is to minimize an objective function of the form

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{1.1}$$

where $x \in \mathbb{R}^d$ is the model parameter. Such functions are ubiquitous in machine learning, arising from the empirical risk minimization (ERM) problem where f_i takes the form

$$f_i(x) = \ell(w_i^{\mathsf{T}} x; y_i) + \lambda R(x),$$

with $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ the loss function, $R : \mathbb{R}^d \to \mathbb{R}_+$ the regularizer, $\lambda \geq 0$ the regularization parameter, and $\{(w_i, y_i) \in \mathbb{R}^d \times \mathbb{R} : i = 1, \dots, n\}$ the training set with labels. Different choices of ℓ and R give l^2-regularized logistic regression, lasso regression, soft-margin support vector machine, etc.

The challenge here is that the dimension d and sample size n are extremely large in modern situations, mandating the use of *first-order methods* that rely only on first derivatives. But when n is large, even computing the full gradient of all $f_1, \dots, f_n$ is intractable, and we need *stochastic optimization methods* that update x only after processing a small subset of data, permitting progress in the time deterministic methods make only a single step. Consequently, stochastic first-order methods have become the method of choice, with stochastic gradient descent (SGD) [37] and its many variants [39, 7, 17] and various stochastic quasi-Newton methods [21, 6, 46] ruling the day.

Stochastic optimization has grown into a vast subject. We have limited our comparison in this article to stochastic variants of classical methods that rely primarily on *gradients*. We did not include more sophisticated stochastic optimization algorithms that bring in additional features like *moments* [8, 20] or *momentum* [24, 33, 35, 36] for two reasons. Firstly these more sophisticated algorithms invariably require heavy tuning compared to purely gradient-based methods. Secondly we view them as enhancements to gradients-based methods and our proposed stochastic Steffensen methods likewise lend themselves to such enhancements. As such, the most appropriate and equitable comparisons for us would be the aforementioned gradient-based methods.

Convention. In this article, we use the terms *learning rate* and *step size* slightly differently. Take for example our Steffensen–Barzilai–Borwein iteration in (2.9):

$$x_{k+1} = x_k - \frac{\beta_k \|\nabla f(x_k)\|^2}{[\nabla f(x_k + \beta_k \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)} \nabla f(x_k),$$

the coefficient

$$\eta_k^{\mathsf{SBB}} := \frac{\beta_k \|\nabla f(x_k)\|^2}{[\nabla f(x_k + \beta_k \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}$$

will be called a learning rate whereas the coefficient

$$\beta_k := \frac{\|x_k - x_{k-1}\|^2}{[\nabla f(x_k) - \nabla f(x_{k-1})]^{\mathsf{T}} (x_k - x_{k-1})}$$

will be called a step size. In general, the term 'learning rate' will be used exclusively to refer to the coefficient of a search direction, which may be a gradient, a stochastic gradient, a variance-reduced stochastic gradient, etc. The term 'step size' will be used for coefficients in other contexts like β_k in the definition of the learning rate η_k^{SBB}.

We will use η_k to denote a general learning rate. For the learning rate of a particular algorithm, we will indicate the algorithm in superscript. For example, η_k^{SBB} above is the learning rate of Steffensen–Barzilai–Borwein method (SBB). The Barzilai–Borwein step size above will always be denoted β_k throughout.

2. Stochastic Multivariate Steffensen Methods

Our three-step strategy is to (a) push the convergence order of the univariate Steffensen method to its limit, (b) extend the resulting method to a multivariate setting, and then (c) randomize the multivariate algorithm. For (a), we are led naturally to the Barzilai–Borwein step size; for (b), we emulate the multivariate extension of secant method into quasi-Newton method; and for (c), we draw inspiration from stochastic gradient descent and its various derivatives.

2.1. **Deterministic univariate setting.** As we saw in Section 1, univariate Steffensen method:

$$x_{k+1} = x_k - \frac{f'(x_k)^2}{f'(x_k + f'(x_k)) - f'(x_k)} \tag{2.1}$$

avoids second-order derivatives and yet preserves quadratic convergence with the use of two first-order derivatives $f'(x_k + f'(x_k))$ and $f'(x_k)$. With modern hindsight, it is clear that we may obtain an immediate improvement in (2.1), one that is essentially free, by incorporating a coefficient β_k that only depends on quantities already computed. The analysis in the next two results will lead us to an appropriate choice of β_k. Note that although the algorithms require only first derivatives of f, the convergence results assume that f has a higher degree of smoothness.

Proposition 2.1 (Convergence order of Steffensen method)**.** Let f be a function that is C^3 in a neighborhood of a stationary point x^* with $f'(x^*) = 0$ and $f''(x^*) \neq 0$. Let $\alpha \in \mathbb{R}$ be a nonzero constant parameter and

$$x_{k+1} \coloneqq x_k - \frac{\alpha f'(x_k)^2}{f'\big(x_k + \alpha f'(x_k)\big) - f'(x_k)}$$

for $k = 0, 1, 2, \dots$. If $\lim_{k\to\infty} x_k = x^*$, then

$$\lim_{k\to\infty} \frac{|\varepsilon_{k+1}|}{|\varepsilon_k^2|} = \frac{1}{2}\left|\frac{f'''(x^*)}{f''(x^*)}\right| \big|1 + \alpha f''(x^*)\big|,$$

where $\varepsilon_k \coloneqq x_k - x^*$ denotes the error in iteration k.

Proof. Let $\varepsilon_k = x_k - x^*$. Subtracting x^* from both sides, we get

$$\varepsilon_{k+1} = \varepsilon_k - \frac{\alpha f'(x_k)^2}{f'(x_k + \alpha f'(x_k)) - f'(x_k)}.$$

Taylor expanding $f'(x_k + \alpha f'(x_k))$ about x_k, we get

$$f'(x_k + \alpha f'(x_k)) = f'(x_k) + f''(x_k)\alpha f'(x_k) + \frac{f'''(\xi_k)}{2}\alpha^2 f'(x_k)^2$$

for some ξ_k between x_k and $x_k + \eta f'(x_k)$. Combining the previous two equations, we have

$$\varepsilon_{k+1} = \varepsilon_k - \frac{f'(x_k)}{f''(x_k) + \frac{f'''(\xi_k)}{2}\alpha f'(x_k)} = \frac{-f'(x_k) + f''(x_k)\varepsilon_k + \frac{1}{2}f'''(\xi_k)\alpha f'(x_k)\varepsilon_k}{f''(x_k) + \frac{1}{2}f'''(\xi_k)\alpha f'(x_k)}. \tag{2.2}$$

Taylor expanding f' about x_k, we get

$$0 = f'(x^*) = f'(x_k) - f''(x_k)\varepsilon_k + \frac{f'''(\xi_k^*)}{2}\varepsilon_k^2$$

for some ξ_k^* between x_k and x^*. Plugging $f'(x_k)$ into (2.2) gives us

$$\varepsilon_{k+1} = \frac{f'''(\xi_k^*)\varepsilon_k^2 + \alpha f'''(\xi_k)f''(x_k)\varepsilon_k^2 - \frac{\alpha}{2}f'''(\xi_k)f'''(\xi_k^*)\varepsilon_k^3}{2f''(x_k) + f'''(\xi_k)\alpha f'(x_k)}.$$

Taking limit $k \to \infty$ and using continuity of f', f'', and f''' at x^*, we have

$$\begin{aligned}\lim_{k\to\infty} \frac{|\varepsilon_{k+1}|}{|\varepsilon_k^2|} &= \lim_{k\to\infty}\left|\frac{f'''(\xi_k^*) + \alpha f'''(\xi_k)f''(x_k) - \frac{\alpha}{2}f'''(\xi_k)f'''(\xi_k^*)\varepsilon_k}{2f''(x_k) + f'''(\xi_k)\alpha f'(x_k)}\right| \\ &= \frac{1}{2}\left|\frac{f'''(x^*)}{f''(x^*)}\right||1 + \alpha f''(x^*)|\end{aligned}$$

as required. □

We next show that with an appropriate choice of α, we can push Steffensen method into the superquadratically convergent regime. The quadratic convergence in Proposition 2.1 is independent of the value α and we may thus choose a different α at every step. Of course if we simply set $\alpha_k = -1/f''(x_k)$ in Proposition 2.1, we will obtain a cubically convergent algorithm. However since we want a first-order method whose learning rate depends only on previously computed quantities, we set $\alpha_k = -(x_k - x_{k-1})/[f'(x_k) - f'(x_{k-1})]$ to be the finite difference to avoid second derivatives — as it turns out, this improves convergence order to $1 + \sqrt{2}$.

Theorem 2.2 (Convergence order of Steffensen method with Barzilai–Borwein step size)**.** Let f be a function that is C^4 in a neighborhood of a stationary point x^* with $f'(x^*) = 0$ and $f''(x^*) \neq 0$. Let

$$\beta_k = -\frac{x_k - x_{k-1}}{f'(x_k) - f'(x_{k-1})}$$

and

$$x_{k+1} = x_k - \frac{\beta_k f'(x_k)^2}{f'\big(x_k + \beta_k f'(x_k)\big) - f'(x_k)} \tag{2.3}$$

for $k = 0, 1, 2, \dots$. If $\lim_{k\to\infty} x_k \to x^*$, then

$$\lim_{k\to\infty} \frac{|\varepsilon_{k+1}|}{|\varepsilon_k^2 \varepsilon_{k-1}|} = \Big(\frac{f'''(x^*)}{2f''(x^*)}\Big)^2.$$

In particular, the order of convergence of (2.3) is superquadratic with $1 + \sqrt{2} \approx 2.414$.

Proof. Taylor expanding $f'(x_k + \beta_k f'(x_k))$ at x_k, we get

$$f'(x_k + \beta_k f'(x_k)) = f'(x_k) + f''(x_k)\beta_k f'(x_k) + \frac{f^{(3)}(x_k)}{2}\beta_k^2 f'(x_k)^2 + \frac{f^{(4)}(\xi_k)}{6}\beta_k^3 f'(x_k)^3$$

for some ξ_k between x_k and $x_k + \eta_k f'(x_k)$. Let $\varepsilon_k = x_k - x^*$, we have

$$\begin{aligned} \varepsilon_{k+1} &= \varepsilon_k - \frac{f'(x_k)}{f''(x_k) + \frac{1}{2}f^{(3)}(x_k)\beta_k f'(x_k) + \frac{1}{6}f^{(4)}(\xi_k)\beta_k^2 f'(x_k)^2} \\ &= \frac{-f'(x_k) + f''(x_k)\varepsilon_k + \frac{1}{2}f^{(3)}(x_k)\beta_k f'(x_k)\varepsilon_k + \frac{1}{6}f^{(4)}(\xi_k)\beta_k^2 f'(x_k)^2\varepsilon_k}{f''(x_k) + \frac{1}{2}f^{(3)}(x_k)\beta_k f'(x_k) + \frac{1}{6}f^{(4)}(\xi_k)\beta_k^2 f'(x_k)^2}. \end{aligned} \tag{2.4}$$

Taylor expanding $f'(x^*)$ at x_k to 4th, 3th, and 2nd order, we get

$$\begin{aligned} 0 = f'(x^*) &= f'(x_k) - f''(x_k)\varepsilon_k + \frac{f^{(3)}(x_k)}{2}\varepsilon_k^2 - \frac{f^{(4)}(\xi_k^*)}{6}\varepsilon_k^3, \\ 0 = f'(x^*) &= f'(x_k) - f''(x_k)\varepsilon_k + \frac{f^{(3)}(\xi_k')}{2}\varepsilon_k^2, \\ 0 = f'(x^*) &= f'(x_k) - f''(\xi_k^\dagger)\varepsilon_k. \end{aligned}$$

Plugging these into (2.4) and defining

$$\begin{aligned} A_k &\coloneqq f''(x_k) + \frac{f^{(3)}(x_k)}{2}\beta_k f'(x_k) + \frac{f^{(4)}(\xi_k)\beta_k^2 f'(x_k)^2}{6}, \\ B_k &\coloneqq \frac{f^{(4)}(\xi_k)}{6}\beta_k^2 f''(\xi_k^\dagger)^2\varepsilon_k^3 - \frac{f^{(4)}(\xi_k^*)}{6}\varepsilon_k^3 - \frac{f^{(3)}(x_k)}{4}f^{(3)}(\xi_k')\beta_k\varepsilon_k^3, \end{aligned}$$

we obtain

$$\varepsilon_{k+1} = \frac{\frac{1}{2}f^{(3)}(x_k)\varepsilon_k^2(f''(x_k)\beta_k + 1) + B_k}{A_k}.$$

Since $\beta_k = -(x_k - x_{k-1})/(f'(x_k) - f'(x_{k-1}))$, we may Taylor expand $f'(x_{k-1})$ at x_k to get

$$f'(x_{k-1}) = f'(x_k) + f''(x_k)(\varepsilon_{k-1} - \varepsilon_k) + \frac{f^{(3)}(\xi_k^\ddagger)}{2}(\varepsilon_{k-1} - \varepsilon_k)^2$$

for some $\xi_k^\ddagger$ between x_{k-1} and x_k. Plugging it into

$$\beta_k = -\frac{1}{f''(x_k) + \frac{1}{2}f^{(3)}(\xi_k^\ddagger)(\varepsilon_{k-1} - \varepsilon_k)}$$

gives us

$$\varepsilon_{k+1} = \frac{\dfrac{f^{(3)}(x_k)f^{(3)}(\xi_k^\ddagger)\varepsilon_k^2(\varepsilon_{k-1} - \varepsilon_k)}{2(2f''(x_k) + f^{(3)}(\xi_k^\ddagger)(\varepsilon_{k-1} - \varepsilon_k))} + B_k}{A_k}.$$

We deduce that

$$\lim_{k\to\infty} \frac{|\varepsilon_k|}{|\varepsilon_{k-1}|} = 0, \qquad \lim_{k\to\infty} \frac{|B_k|}{|\varepsilon_k^2\varepsilon_{k-1}|} = 0,$$

and therefore

$$\lim_{k\to\infty} \frac{|\varepsilon_{k+1}|}{|\varepsilon_k^2\varepsilon_{k-1}|} = \Big(\frac{f^{(3)}(x^*)}{2f^{(2)}(x^*)}\Big)^2.$$

Hence the convergence order is $1 + \sqrt{2}$. □

The choice of β_k above is exactly the Barzilai–Borwein (BB) step size for a univariate function [3]. In the multivariate setting, β_k will be replaced by the multivariate BB step size. Theorem 2.2 provides the impetus for a first-order method with Steffensen updates and BB step size, namely, it is superquadratically convergent for univariate functions. Such a high convergence order is clearly an overkill for a deterministic algorithm but our experiments in Section 5 show that they are rewarding when the algorithm is randomized, as randomization inevitably compromises convergence speed. For easy comparison, we tabulate the convergence order, i.e., the largest q such that $|\varepsilon_{k+1}| \le c|\varepsilon_k|^q$ for some $c > 0$ and all k sufficiently large, of various methods below:

Method	Convergence	Derivatives	Steps
Steepest descent	1	1st	single step
Secant = Barzilai–Borwein = quasi-Newton	$(1+\sqrt{5})/2$	1st	mutltistep
Newton	2	2nd	single step
Steffensen	2	1st	single step
Steffensen–Barzilai–Borwein	$1+\sqrt{2}$	1st	multistep

Note that for a univariate function, Barzilai–Borwein step size and any quasi-Newton method with Broyden class updates (including BFGS, DFP, SR1) reduce to the secant method. In particular, they are all two-step methods, i.e., its iterate at step k depends on both x_k and x_{k-1}. As a result Steffensen–Barzilai–Borwein method is also a two-step method as it involves the Brazlai–Borwein step size but Steffensen method is a one-step method.

2.2. **Deterministic multivariate setting.** There have been no shortage of proposals for extending Steffensen method to a multivariate or even infinite-dimensional setting [1, 9, 13, 15, 16, 22, 25, 26, 28, 29, 30, 31, 32]. However all of them rely on various multivariate versions of *divided differences* that require evaluation and storage of $O(n^2)$ first derivatives in each step. Although they do avoid second derivatives, computationally they are just as expensive as Newton method and are unsuitable for modern large scale applications like training deep neural networks.

We will propose an alternative class of multivariate Steffensen methods that use only $O(n)$ first derivatives, by emulating quasi-Newton methods [5, 10, 12, 40] and Barzilai–Borwein method [3] respectively. Our observation is that expensive multivariate divided differences can be completely avoided if we just use the ideas in Section 2.1 to *define learning rates*. Another advantage is

that these learning rates could be readily used in conjunction with existing stochastic optimization methods, as we will see in Section 2.3.

The key idea behind quasi-Newton method is the extension of univariate secant method to a multivariate objective function $f : \mathbb{R}^d \to \mathbb{R}$ by replacing the finite difference approximation of $f''(x_k)$, i.e., $h_k = [f'(x_k) - f'(x_{k-1})]/(x_k - x_{k-1})$, with the *secant equation* $H_k s_k = y_k$ or

$$B_k y_k = s_k \tag{2.5}$$

where $s_k = x_k - x_{k-1}$ and $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$, avoiding the need to divide vectorial quantitites. Here H_k (resp. B_k) is the approximate (resp. inverse) Hessian.

We use the same idea to extend Steffensen method to a multivariate setting, solving (2.5) with

$$s_k = \nabla f(x_k), \qquad y_k = \nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k).$$

Note that with these choices, (2.5) roughly says that "$B_k = s_k/y_k = \nabla f(x_k)/[\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k)]$," which gives us $f'(x_k)/[f'(x_k + f'(x_k)) - f'(x_k)]$ as in the univariate Steffensen method when $d = 1$ but is of course meaningless when $d > 1$. Nevertheless we may pick a minimum-norm solution to (2.5), which is easily seen to be given by the rank-one matrix

$$B_k = \operatorname*{argmin}_{By_k = s_k} \|B\| = \frac{s_k y_k^{\mathsf{T}}}{y_k^{\mathsf{T}} y_k}$$

regardless of whether $\|\cdot\|$ is the Frobenius or spectral norm. Hence we obtain a multivariate analogue of Steffensen method (2.1) as

$$x_{k+1} = x_k - B_k \nabla f(x_k) = x_k - \frac{[\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}{\|\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k)\|^2} \nabla f(x_k). \tag{2.6}$$

We will call this *quasi-Steffensen method* in analogy with quasi-Newton methods.

The key idea behind the Barzilai–Borwein method [3] is an alternative way of treating the secant equation (2.5), whereby the approximate Hessian B_k is assumed to take the form $B_k = \sigma_k I$ for some scalar $\sigma_k > 0$. Since in general it is not possible to find σ_k so that (2.5) holds exactly with $B_k = \sigma_k I$, a best approximation is used instead. We seek σ_k so that the residual of the secant equation $\|y_k - (1/\sigma_k)s_k\|^2$ or $\|\sigma_k y_k - s_k\|^2$ is minimized. The first minimization problem gives us

$$\sigma_k = \operatorname*{argmin}_{\sigma > 0} \|y_k - (1/\sigma)s_k\|^2 = \frac{s_k^{\mathsf{T}} s_k}{s_k^{\mathsf{T}} y_k} = \frac{\|\nabla f(x_k)\|^2}{[\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}, \tag{2.7}$$

and the second minimization gives the same expression as (2.6). We will call the resulting iteration

$$x_{k+1} = x_k - \frac{\|\nabla f(x_k)\|^2}{[\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)} \nabla f(x_k)$$

Steffensen method since it most resembles the univariate Steffensen method in (2.1). Note that the Barzilai–Borwein step size derived in [3] is

$$\beta_k = \frac{\|x_k - x_{k-1}\|^2}{[\nabla f(x_k) - \nabla f(x_{k-1})]^{\mathsf{T}} (x_k - x_{k-1})} \tag{2.8}$$

and differs significantly from (2.7). In particular, $x_{k+1} = x_k - \beta_k \nabla f(x_k)$ is a multistep method whereas $x_{k+1} = x_k - \sigma_k \nabla f(x_k)$ remains a single step method.

Both (2.6) and (2.7) reduce to (2.1) when f is univariate. Motivated by the univariate discussion before Theorem 2.2, we combine features from (2.7) and (2.8) to obtain a *Steffensen–Barzilai–Borwein method* in analogy with the univariate case (2.3):

$$x_{k+1} = x_k - \frac{\beta_k \|\nabla f(x_k)\|^2}{[\nabla f(x_k + \beta_k \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)} \nabla f(x_k). \tag{2.9}$$

Note that (2.9) reduces to (2.3) when f is univariate. The stochastic version of (2.9) will be our method of choice, supported by extensive empirical evidence some of which we will present in Section 5.

In summary, we have four plausible learning rates.

$$\text{quasi-Steffensen:} \qquad \eta_k^{\mathsf{qS}} = \frac{[\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}{\|\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k)\|^2},$$

$$\text{quasi-Steffensen–Barzilai–Borwein:} \qquad \eta_k^{\mathsf{qSBB}} = \frac{\beta_k[\nabla f(x_k + \beta_k \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}{\|\nabla f(x_k + \beta_k \nabla f(x_k)) - \nabla f(x_k)\|^2},$$

$$\text{Steffensen:} \qquad \eta_k^{\mathsf{S}} = \frac{\|\nabla f(x_k)\|^2}{[\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)},$$

$$\text{Steffensen–Barzilai–Borwein:} \qquad \eta_k^{\mathsf{SBB}} = \frac{\beta_k \|\nabla f(x_k)\|^2}{[\nabla f(x_k + \beta_k \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}.$$

Here β_k is the Barzilai–Borwein step size in (2.8). For a univariate function, the iterations with η_k^{qS} and η_k^{S} reduce to (2.1) whereas those with η_k^{qSBB} and η_k^{SBB} reduce to (2.3). The computational costs of all four learning rates are the same: two gradient evaluations and two inner products.

Note that our muiltivariate Steffensen and quasi-Steffensen methods are one-step methods — η_k^{S} and η_k^{qS} depend only on x_k — just like the univariate Steffensen method. Steffensen–Barzilai–Borwein and quasi-Steffensen–Barzilai–Borwein are inevitably two-step methods because they involve the Barzilai–Borwein step size β_k, which has a two-step formula.

The main difference between our multivariate Steffensen methods and those in the literature [1, 9, 13, 15, 16, 22, 25, 26, 28, 29, 30, 31, 32] is that ours are encapsulated as learning rates and avoid expensive multivariate divided differences. Recall that for $g = (g_1, \dots, g_n) : \mathbb{R}^n \to \mathbb{R}^n$, its divided difference [34] at $x, y \in \mathbb{R}^n$ is the matrix $[\![x, y]\!] \in \mathbb{R}^{n \times n}$ whose (i, j)th entry is

$$[\![x, y]\!]_{ij} \coloneqq \begin{cases} \dfrac{g_i(x_1, \dots, x_j, y_{j+1}, \dots, y_n) - g_i(x_1, \dots, x_{j-1}, y_j, \dots, y_n)}{x_j - y_j} & x_j \neq y_j, \\ \dfrac{\partial g_i}{\partial x_j}(x_1, \dots, x_j, y_{j+1}, \dots, y_n) & x_j = y_j, \end{cases}$$

for $i, j = 1, \dots, n$.

In a stochastic setting, the learning rates $\eta_k^{\mathsf{S}}, \eta_k^{\mathsf{qS}}, \eta_k^{\mathsf{SBB}}, \eta_k^{\mathsf{qSBB}}$ share the same upper and lower bounds in Lemma 3.6 and as a result, the linear convergence conclusion in Theorem 3.9 applies alike to all four of them. Our experiments also indicate that η_k^{qS} and η_k^{S} have similar performance and likewise for η_k^{qSBB} and η_k^{SBB}, although there is a slight difference between η_k^{S} and η_k^{SBB}. One conceivable advantage of the 'quasi' variants is that for a given $\nabla f(x_k)$, the denominator vanishes only at a single point, e.g., when $\nabla f(x_k + \nabla f(x_k)) = \nabla f(x_k)$, as opposed to a whole hyperplane, e.g., whenever $\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k) \perp \nabla f(x_k)$. Nevertheless, in all our experiments on their stochastic variants, this has never been an issue.

We prefer the slightly simpler expressions of the Steffensen and Steffensen–Barzilai–Borwein methods and will focus our subsequent discussions on them. Their 'quasi' variants may be taken as nearly equivalent alternatives for users who may have some other reasons to favor them.

2.3. **Stochastic multivariate setting.** Encapsulating Steffensen method in the form of learning rates offers an additional advantage — it is straightforward to incorporate them into many stochastic optimization algorithms, which we will do next.

Standard gradient descent applied to (1.1) requires the evaluation of n gradients. The stochastic gradient descent (SGD), instead of using the full gradient $\nabla f(x_k)$, relies on an unbiased estimator g_k with $\mathbb{E}[g_k] = \nabla f(x_k)$ [37]. One common randomization is to draw $i_k \in \{1, \dots, n\}$ randomly and

set $g_k = \nabla f_{i_k}(x_k)$, giving the update:

$$x_{k+1} = x_k - \eta_k \nabla f_{i_k}(x_k).$$

Note that $\mathbb{E}[\nabla f_{i_k}(x_k) \mid x_k] = \nabla f(x_k)$ and its obvious advantage is that each step relies only on a single gradient ∇f_{i_k}, resulting in a computational cost that is $1/n$ that of the standard gradient descent. While we could adopt this procedure to randomize our Steffensen and Steffensen–Barzilai–Borwein iterations, we will use a slightly more sophisticated variant with *variance reduction* and *minibatching*.

The price of randomization is paid in the form of variance, as the stochastic gradient $\nabla f_{i_k}(x_k)$ equals the gradient $\nabla f(x_k)$ only in expectation but each $\nabla f_{i_k}(x_k)$ is different. Of the many variance reduction strategies, one of the best known and simplest is the stochastic variance reduced gradient method (SVRG) [17], based on the tried-and-tested notion of control variates in Monte Carlo methods. We will emulate SVRG to randomize (2.6) and (2.9).

The basic idea of SVRG is to compute the full gradient once every m iterations for some fixed m and use it to generate stochastic gradients with lower variance in the next m iterations:

$$x_{k+1} = x_k - \eta_k\big(\nabla f_{i_k}(x_k) - \nabla f_{i_k}(\widetilde{x}) + \nabla f(\widetilde{x})\big).$$

Here $\widetilde{x}$ denotes the point where full gradient is computed. Notice that when $k \to \infty$, x_k and $\widetilde{x}$ are very close to the optimal point x^*. As x_k and $\widetilde{x}$ are highly correlated, the variability of the stochastic gradient is reduced as a result [17].

We may similarly randomize multivariate Steffensen method. Our stochastic Steffensen method (SSM) in Algorithm 1 operates in two nested loops. In the kth iteration of the outer loop, we compute two full gradients $\nabla f(x_k)$ and $\nabla f(x_k + \nabla f(x_k))$. Note that x_k plays the role of $\widetilde{x}$ in the above paragraph. These two terms are used for computing the Steffensen learning rate:

$$\eta_k^{\mathsf{SS}} = \frac{1}{\sqrt{m}} \cdot \frac{\|\nabla f(x_k)\|^2}{[\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}. \tag{2.10}$$

In the $(t+1)$th iteration of the inner loop, we use $\nabla f(x_k)$ to generate the stochastic gradient with lower variance

$$v_{k,t} = \nabla f_{i_t}(x_{k,t}) - \nabla f_{i_t}(x_k) + \nabla f(x_k),$$

with $i_t \in \{1, \dots, n\}$ sampled uniformly. The updating rule takes the form

$$x_{k,t+1} = x_{k,t} - \eta_k^{\mathsf{SS}} v_{k,t}$$

where the search direction is known as the *variance-reduced stochastic gradient*. Note that the learning rate η_k given by (2.10) has an extra $1/\sqrt{m}$ factor to guarantee the linear convergence. The $1/\sqrt{m}$ factor may be replaced by $1/m^p$ for any $p \in (0, 1)$, although not $p = 1$, while preserving linear convergence. The optimal complexity, as measured by the number of stochasic gradient evaluations, is achieved when $p = 1/2$, with details to follow in Section 3.

Aside from variance reduction, we include another common enhancement called *minibatching*. Minibatched SGD is a trade-off between SGD and gradient descent (GD) where the cost function (and therefore its gradient) is averaged over a small number of samples. SGD has a batch size of one whereas GD has a batch size that includes all training samples. In each iteration, we sample a minibatch $S_k \subseteq \{1, \dots, n\}$ with $|S_k| = b$ a small number and update

$$x_{k+1} = x_k - \eta_k \frac{1}{|S_k|} \sum_{j \in S_k} \nabla f_j(x_k) =: x_k - \eta_k \nabla f_{S_k}(x_k).$$

Minibatched SGD smooths out some of the noise in SGD but maintains the ability to escape local minima. The minibatch size b is kept small, thus preserving the cost-saving benefits of SGD. As in the discussion after (2.10), the coefficient $1/\sqrt{m}$ may be replaced by $1/m^p$ for any $p \in (0, 1)$ while preserving linear convergence, but $p = 1/2$ gives the minimum number of stochastic gradient evaluations, as we will see in Section 3.

Algorithm 1 Stochastic Steffensen Method (SSM)

Input: initial state x_0, inner loop size m, data size n.
for $k = 0, 1, \dots$ **do**
  Compute full gradients $\nabla f(x_k)$ and $\nabla f(x_k + \nabla f(x_k))$.
  Compute stochastic Steffensen learning rate

$$\eta_k^{\mathsf{SS}} = \frac{1}{\sqrt{m}} \cdot \frac{\|\nabla f(x_k)\|^2}{[\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}.$$

  Set $x_{k,0} = x_k$.
  for $t = 0$ **to** $m - 1$ **do**
    Sample $i_t \in \{1, \dots, n\}$ uniformly.
    Compute variance-reduced stochastic gradient

$$v_{k,t} = \nabla f_{i_t}(x_{k,t}) - \nabla f_{i_t}(x_k) + \nabla f(x_k).$$

    Update $x_{k,t+1} = x_{k,t} - \eta_k^{\mathsf{SS}} v_{k,t}$.
  end for
  Set $x_{k+1} = x_{k,i}$ for uniformly chosen $i \in \{0, \dots, m - 1\}$.
end for

Upon incorporating (i) a Barzilai–Borwein step size, (ii) variance reduction, and (iii) minibatching, we arrive at the stochastic Steffensen–Barzilai–Borwein method (SSBB) in Algorithm 2. This is our method of choice in this article.

Although we did not include minibatching in Algorithm 1's pseudocode to avoid clutter, we will henceforth assume that it is also minibatched. The randomization, variance reduction, and minibatching all apply verbatim when the learning rates in Algorithms 1 and 2 are replaced respectively by the quasi-Steffensen and quasi-Steffensen–Barzilai–Borwein learning rates on p. 8. Nevertheless, as we have mentioned, our numerical experiments do not show that the resulting algorithms differ in performance from that of Algorithms 1 and 2.

2.4. **Randomized Kaczmarz method as a special case.** Given $A \in \mathbb{R}^{m \times n}$ of full row rank with *row* vectors $a_1, \dots, a_m \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ in the image of A, the Kaczmarz method [18, 19] solves the consistent linear system $Ax = b$ via

$$x_{k+1} = x_k + \frac{b_i - a_i^{\mathsf{T}} x_k}{\|a_i\|^2} a_i,$$

with $i = k \mod m$, $i = 1, \dots, m$. The iterative method has remained relatively obscure, almost unheard of in numerical linear algebra, until it was randomized in [43], which essentially does

$$x_{k+1} = x_k + \frac{b_{i_k} - a_{i_k}^{\mathsf{T}} x_k}{\|a_{i_k}\|^2} a_{i_k},$$

where $i_k \in \{1, \dots, m\}$ is now *sampled* with probability $\|a_{i_k}\|^2 / \|A\|^2$.

We will see that randomized Kaczmarz method is equivalent to applying stochastic Steffensen method, with or without Barzilai–Borwein step size, to minimize the quadratic function $f : \mathbb{R}^n \to \mathbb{R}$,

$$f(x) \coloneqq \frac{1}{2} \sum_{i=1}^m f_i(x) = \frac{1}{2} \sum_{i=1}^m (a_i^{\mathsf{T}} x - b_i)^2.$$

While it is sometimes claimed that SGD has this property, this is not quite true. Suppose $i_k \in \{1, \dots, m\}$ is the random row index sampled at the kth step, the update rule in SGD gives

$$x_{k+1} = x_k - \eta_k (a_{i_k}^{\mathsf{T}} x_k - b_i) a_{i_k},$$

Algorithm 2 Stochastic Steffensen–Barzilai–Borwein Method (SSBB)

Input: initial state x_0, inner loop size m, minibatch size b, data size n, Barzilai-Borwein step size $\beta_0 = -1$.
for $k = 0, 1, \dots$ **do**
  Compute full gradient $\nabla f(x_k)$.
  if $k > 0$ **then**
    Set $s_k = x_k - x_{k-1}$ and $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$.
    Compute Barzilai–Borwein step size

$$\beta_k = -\frac{\|s_k\|^2}{s_k^{\mathsf{T}} y_k}.$$

  end if
  Compute the stochastic Steffensen–Barzilai–Borwein learning rate

$$\eta_k^{\mathsf{SSBB}} = \frac{1}{\sqrt{m}} \cdot \frac{\beta_k \|\nabla f(x_k)\|^2}{[\nabla f(x_k + \beta_k \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}.$$

  Set $x_{k,0} = x_k$.
  for $t = 0$ **to** $m - 1$ **do**
    Sample minibatch $S_{k,t} \subseteq \{1, \dots, n\}$ uniformly with $|S_{k,t}| = b$.
    Compute variance-reduced stochastic gradient

$$v_{k,t} = \nabla f_{S_{k,t}}(x_{k,t}) - \nabla f_{S_{k,t}}(x_{k,t}) + \nabla f(x_{k,t}).$$

    Update $x_{k,t+1} = x_{k,t} - \eta_k^{\mathsf{SSBB}} v_{k,t}$.
  end for
  Set $x_{k+1} = x_{k,i}$ for uniformly chosen $i \in \{0, \dots, m-1\}$.
end for

and the update rule in SLBFGS is even further from this. So one needs to impose further assumptions [23] on the learning rate to get randomized Kaczmarz method, which requires that $\eta_k = 1/\|a_{i_k}^2\|$. If we use the Steffensen method, we get from (2.9) that

$$\eta_k^{\mathsf{S}} = \frac{\|\nabla f_{i_k}(x_k)\|^2}{[\nabla f_{i_k}\big(x_k + \nabla f_{i_k}(x_k)\big) - \nabla f_{i_k}(x_k)]^{\mathsf{T}} \nabla f_{i_k}(x_k)} = \frac{1}{\|a_{i_k}\|^2};$$

and using Steffensen–Barzilai–Borwein method makes no difference:

$$\eta_k^{\mathsf{SBB}} = \frac{\beta_k \|\nabla f_{i_k}(x_k)\|^2}{[\nabla f_{i_k}\big(x_k + \beta_k \nabla f_{i_k}(x_k)\big) - \nabla f_{i_k}(x_k)]^{\mathsf{T}} \nabla f_{i_k}(x_k)} = \frac{1}{\|a_{i_k}\|^2},$$

as $\beta_k = \|x_k - x_{k-1}\|^2/(x_k - x_{k-1})^{\mathsf{T}}[\nabla f_{i_k}(x_k) - \nabla f_{i_k}(x_{k-1})] = 1/\|a_{i_k}\|^2$.

3. Convergence Analysis

In this section, we establish the linear convergence of our stochastic Steffensen methods Algorithm 1 (SSM) and Algorithm 2 (SSBB) for solving (1.1) under standard assumptions. We would like to stress that these convergence results are intended to provide a minimal theoretical guarantee and do not really do justice to the actual performance of SSBB. The experiments in Section 5 indicate that the convergence of SSBB is often superior to other existing methods like SGD and SVRG, with or without Barzilai–Borwein step size, or even SLBFGS. However, we are unable to prove this theoretically, only that it is linearly convergent like the other methods.

For easy reference, we reproduce the minibatched SVRG algorithm in [2, Algorithm 1] as Algorithm 3. We need to establish the linear convergence of Algorithm 3 for our own convergence results

Algorithm 3 Minibatched SVRG

Input: initial state x_0, inner loop size m, minibatch size b, data size n.
for $k = 0, 1, \dots$ **do**
  Compute full gradient $\nabla f(x_k)$.
  Set $x_{k,0} = x_k$.
  for $t = 0$ **to** $m - 1$ **do**
    Sample minibatch $S_{k,t} \subseteq \{1, \dots, n\}$ uniformly with $|S_{k,t}| = b$.
    Compute variance-reduced stochastic gradient

$$v_{k,t} = \nabla f_{S_{k,t}}(x_{k,t}) - \nabla f_{S_{k,t}}(x_k) + \nabla f(x_k).$$

    Update $x_{k,t+1} = x_{k,t} - \eta_k v_{k,t}$.
  end for
  Set $x_{k+1} = x_{k,i}$ for uniformly chosen $i \in \{0, \dots, m-1\}$.
end for

in Sections 3.1 and 3.2 but we are unable to find such a result in the literature. In particular, the convergence results in [2, Propositions 2–4] and [45, Theorem 1] are for more sophisticated variants of Algorithm 3. So we will provide a version following the same line of arguments in [45, Theorem 1] but tailored to our own requirements.

Our linear convergence proofs for SSM and SSBB are a combination of the proofs in [27, 45] adapted for our purpose. In particular, we quote [27, Lemma A] and prove a simplied version of [45, Lemma 3] for easy reference.

Lemma 3.1 (Nitanda)**.** Let $\xi_1, \dots, \xi_n \in \mathbb{R}^d$ and $\bar{\xi} := \frac{1}{n}\sum_{i=1}^n \xi_i$. Let S be a b-element subset chosen uniform randomly from all b-element subsets of $\{1, 2, \dots, n\}$. Then

$$\mathbb{E}_S \Big\| \frac{1}{b} \sum\nolimits_{i \in S} \xi_i - \bar{\xi} \Big\|^2 = \frac{n-b}{b(n-1)} \mathbb{E}_i \big\| \xi_i - \bar{\xi} \big\|^2.$$

Here $\mathbb{E}_S$ denotes expectation of the random subset $S \subseteq \{1, \dots, n\}$ and $\mathbb{E}_i$ that of the uniform random variable $i \in \{1, \dots, n\}$. More specifically, if $S = \{i_1, \dots, i_b\}$, then

$$\mathbb{E}_S \Big\| \frac{1}{b} \sum_{i \in S} \xi_i - \bar{\xi} \Big\|^2 = \frac{b!(n-b)!}{n!} \sum_{S \subseteq \{1,\dots,n\}} \Big\| \frac{1}{b} \sum_{j=1}^{b} \xi_{i_j} - \bar{\xi} \Big\|^2, \qquad \mathbb{E}_i \big\| \xi_i - \bar{\xi} \big\|^2 = \frac{1}{n} \sum_{j=1}^{n} \big\| \xi_j - \bar{\xi} \big\|^2.$$

For the rest of this section, we will need to assume, as is customary in such proofs of linear convergence, that our objective f is μ-strongly convex, the gradient of each additive component f_i is L-Lipschitz continuous (and therefore so is ∇f), and that all iterates are well-defined (the denominators appearing in our learning rates η_k are nonzero).

Assumption 3.2. Assume that the function f in (1.1) satisfies

$$f(w) \geq f(v) + \nabla f(v)^{\mathsf{T}}(w - v) + \frac{\mu}{2}\|v - w\|^2,$$
$$\|\nabla f_i(v) - \nabla f_i(w)\| \leq L\|v - w\|$$

for any $v, w \in \mathbb{R}^d$, $i = 1, \dots, n$.

Applying Lemma 3.1 with $\xi_i = v_i^{k,t}$ and [45, Corollary 3], we may bound the variance of a minibatched variance-reduced gradient as follows.

Lemma 3.3. Let f be as in Assumption 3.2 with $x^* \coloneqq \operatorname{argmin}_x f(x)$. Let

$$v_i^{k,t} = \nabla f_i(x_{k,t}) - \nabla f_i(x_k) + \nabla f(x_k), \qquad v_{k,t} = \frac{1}{b}\sum_{i\in S_{k,t}} v_i^{k,t}.$$

Then

$$\mathbb{E}\|v_{k,t} - \nabla f(x_{k,t})\|^2 \le \frac{4L}{b}\big[f(x_{k,t}) - f(x^*) + f(x_k) - f(x^*)\big].$$

The next lemma, a simplified version of [45, Lemma 3], gives a lower bound of the optimal value $f(x^*)$ useful in our proof of linear convergence.

Lemma 3.4. Let $\Delta_{k,t} \coloneqq v_{k,t} - \nabla f(x_{k,t})$ and η_k be a learning rate with $0 < \eta_k \le 1/L$. Then with the same assumptions and notations in Lemma 3.3, we have

$$f(x^*) \ge f(x_{k,t+1}) + v_{k,t}^\mathsf{T}(x^* - x_{k,t}) + \frac{\eta_k}{2}\|v_{k,t}\|^2 + \frac{\mu}{2}\|x^* - x_{k,t}\|^2 + \Delta_{k,t}^\mathsf{T}(x_{k,t+1} - x^*).$$

Proof. By the strong convexity of f, we have

$$f(x^*) \ge f(x_{k,t}) + \nabla f(x_{k,t})^\mathsf{T}(x^* - x_{k,t}) + \frac{\mu}{2}\|x^* - x_{k,t}\|^2.$$

By the smoothness of f, we have

$$f(x_{k,t}) \ge f(x_{k,t+1}) - \nabla f(x_{k,t+1})^\mathsf{T}(x_{k,t+1} - x_{k,t}) - \frac{L}{2}\|x_{k,t+1} - x_{k,t}\|^2.$$

Summing the two inequalities, we get

$$f(x^*) \ge f(x_{k,t+1}) + \nabla f(x_{k,t})^\mathsf{T}(x^* - x_{k,t+1}) + \frac{\mu}{2}\|x^* - x_{k,t}\|^2 - \frac{L\eta_k^2}{2}\|v_{k,t}\|^2.$$

The second term on the right simplifies as

$$\begin{aligned}\nabla f(x_{k,t})^\mathsf{T}(x^* - x_{k,t+1}) &= \nabla f(x_{k,t})^\mathsf{T}(x^* - x_{k,t+1}) + (v_{k,t} - v_{k,t})^\mathsf{T}(x^* - x_{k,t+1})\\ &= v_{k,t}^\mathsf{T}(x^* - x_{k,t+1}) + (v_{k,t} - \nabla f(x_{k,t}))^\mathsf{T}(x_{k,t+1} - x^*)\\ &= v_{k,t}^\mathsf{T}(x^* - x_{k,t+1}) + \eta_k\|v_{k,t}\|^2.\end{aligned}$$

If the learning rate satisfies $0 < \eta_k \le 1/L$, then

$$\begin{aligned}f(x^*) &\ge f(x_{k,t+1}) + v_{k,t}^\mathsf{T}(x^* - x_{k,t}) + \frac{\eta_k}{2}(2 - L\eta_k)\,\|v_{k,t}\|^2 + \frac{\mu}{2}\,\|x^* - x_{k,t}\|^2 + \Delta_{k,t}^\mathsf{T}(x_{k,t+1} - x^*)\\ &\ge f(x_{k,t+1}) + v_{k,t}^\mathsf{T}(x^* - x_{k,t}) + \frac{\eta_k}{2}\,\|v_{k,t}\|^2 + \frac{\mu}{2}\,\|x^* - x_{k,t}\|^2 + \Delta_{k,t}^\mathsf{T}(x_{k,t+1} - x^*),\end{aligned}$$

as required. □

Theorem 3.5 (Linear convergence of Algorithm 3)**.** Let f be as in Assumption 3.2 with $x^* \coloneqq \operatorname{argmin}_x f(x)$. For the $(k+1)$th iteration of outer loop in Algorithm 3,

$$\mathbb{E}[f(x_{k+1}) - f(x^*)] \le \left[\frac{b}{m\mu\eta_k(b - 4L\eta_k)} + \frac{4(m+1)L\eta_k}{m(b - 4L\eta_k)}\right][f(x_k) - f(x^*)].$$

If m, η_k, and b are chosen so that

$$\rho_k = \frac{b}{m\mu\eta_k(b - 4L\eta_k)} + \frac{4(m+1)L\eta_k}{m(b - 4L\eta_k)} \le \rho < 1, \qquad \eta_k < \min\Big(\frac{b}{4L}, \frac{1}{L}\Big),$$

then Algorithm 3 converges linearly in expectation with

$$\mathbb{E}[f(x_k) - f(x^*)] \le \rho^k[f(x_0) - f(x^*)].$$

Proof. For the iteration in the inner loop, we apply Lemma 3.4 to get

$$\begin{aligned}\|x_{k,t+1} - x^*\|^2 &= \|x_{k,t} - x^*\|^2 - 2\eta_k v_{k,t}^{\mathsf{T}}(x_{k,t} - x^*) + \eta_k^2 \|v_{k,t}\|^2 \\ &\leq \|x_{k,t} - x^*\|^2 + 2\eta_k[f(x^*) - f(x_{k,t+1})] - 2\eta_k \Delta_{k,t}^{\mathsf{T}}(x_{k,t+1} - x^*).\end{aligned} \tag{3.1}$$

Lemma 3.4 requires that the learning rate $\eta_k \leq 1/L$. Let $\bar{x}_{k,t+1} \coloneqq x_{k,t} - \eta_k \nabla f(x_{k,t})$. Then the last term in (3.1) may be written as

$$\begin{aligned}-2\eta_k \Delta_{k,t}^{\mathsf{T}}(x_{k,t+1} - x^*) &= -2\eta_k \Delta_{k,t}^{\mathsf{T}}(x_{k,t+1} - \bar{x}_{k,t+1}) - 2\eta_k \Delta_{k,t}^{\mathsf{T}}(\bar{x}_{k,t+1} - x^*) \\ &= 2\eta_k^2 \|\Delta_{k,t}\|^2 - 2\eta_k \Delta_{k,t}^{\mathsf{T}}(\bar{x}_{k,t+1} - x^*).\end{aligned}$$

Plugging this into (3.1) and taking expectations on both sides conditioned on $x_{k,t}$ and x_k respectively, we get

$$\begin{aligned}\mathbb{E}\|x_{k,t+1} - x^*\|^2 &\leq \|x_{k,t} - x^*\|^2 + 2\eta_k[\eta_k \mathbb{E}\|\Delta_{k,t}\|^2 - \mathbb{E}[\Delta_{k,t}^{\mathsf{T}}(\bar{x}_{k,t+1} - x^*)] - (f(x_{k,t+1}) - f(x^*))] \\ &= \|x_{k,t} - x^*\|^2 + 2\eta_k[\eta_k \mathbb{E}\|\Delta_{k,t}\|^2 - (f(x_{k,t+1}) - f(x^*))],\end{aligned}$$

where the last equality follows from $\mathbb{E}[\Delta_{k,t}] = 0$. Set $\gamma \coloneqq 8L\eta_k^2/b$. By Lemma 3.3, we have

$$\mathbb{E}\|x_{k,t+1} - x^*\|^2 \leq \|x_{k,t} - x^*\|^2 + \gamma[f(x_{k,t}) - f(x^*) + f(x_k) - f(x^*)] - 2\eta_k \mathbb{E}[f(x_{k,t+1}) - f(x^*)].$$

For $t = 0, \ldots, m-1$, we have

$$\mathbb{E}\,\|x_{k,t+1} - x^*\|^2 + 2\eta_k \mathbb{E}[f(x_{k,t+1}) - f(x^*)] \leq \|x_{k,t} - x^*\|^2 + \gamma[f(x_{k,t}) - f(x^*) + f(x_k) - f(x^*)].$$

Summing this inequality over all $t = 0, \ldots, m-1$, the left hand side becomes

$$\text{LHS} = \sum_{t=0}^{m-1} \mathbb{E}\,\|x_{k,t+1} - x^*\|^2 + 2\eta_k \sum_{t=0}^{m-1} \mathbb{E}[f(x_{k,t+1}) - f(x^*)],$$

and the right hand side becomes

$$\text{RHS} = \sum_{t=0}^{m-1} \|x_{k,t} - x^*\|^2 + \gamma \sum_{t=0}^{m-1} \mathbb{E}[f(x_{k,t}) - f(x^*)] + \gamma m \mathbb{E}[f(x_k) - f(x^*)].$$

By the definition of x_{k+1} in Algorithm 3,

$$\mathbb{E}[f(x_{k+1})] = \frac{1}{m} \sum_{t=1}^{m} f(x_{k,t}),$$

and so, bearing in mind that LHS $\leq$ RHS,

$$\begin{aligned}&\mathbb{E}\|x_{k,m} - x^*\|^2 + 2\eta_k m \mathbb{E}[f(x_{k+1}) - f(x^*)] \\ &\quad\leq \mathbb{E}\,\|x_{k,0} - x^*\|^2 + \gamma m \mathbb{E}[f(x_k) - f(x^*)] + \gamma \sum_{t=0}^{m-1} \mathbb{E}[f(x_{k,t}) - f(x^*)] \\ &\quad= \mathbb{E}\,\|x_{k,0} - x^*\|^2 + \gamma m \mathbb{E}[f(x_k) - f(x^*)] + \gamma m \mathbb{E}[f(x_{k+1}) - f(x^*)] + \gamma[f(x_k) - f(x^*)],\end{aligned}$$

where the last step follows by replacing $\sum_{t=0}^{m-1}$ with $\sum_{t=0}^{m}$, which preserves inequality. Thus

$$\begin{aligned}2\eta_k m \mathbb{E}[f(x_{k+1}) - f(x^*)] &\leq 2\eta_k m \mathbb{E}[f(x_{k+1}) - f(x^*)] + \mathbb{E}\,\|x_{k,m} - x^*\|^2 \\ &\leq \mathbb{E}\,\|x_k - x^*\|^2 + \gamma(m+1)\mathbb{E}[f(x_k) - f(x^*)] + \gamma m \mathbb{E}[f(x_{k+1}) - f(x^*)].\end{aligned}$$

Rearranging terms and applying strong convexity of f, we have

$$\left(2\eta_k - \frac{8L\eta_k^2}{b}\right) m\mathbb{E}[f(x_{k+1}) - f(x^*)] \leq \mathbb{E}\,\|x_k - x^*\|^2 + \frac{8(m+1)L\eta_k^2}{b}\mathbb{E}[f(x_k) - f(x^*)]$$
$$\leq \frac{2}{\mu}[f(x_k) - f(x^*)] + \frac{8(m+1)L\eta_k^2}{b}\mathbb{E}[f(x_k) - f(x^*)].$$

Here we require that $2\eta_k > 8L\eta_k^2/b$ and thus $\eta_k < b/(4L)$, leading to

$$\mathbb{E}[f(x_{k+1}) - f(x^*)] \leq \rho_k[f(x_k) - f(x^*)]$$

with

$$\rho_k \coloneqq \frac{b}{m\mu\eta_k(b - 4L\eta_k)} + \frac{4(m+1)L\eta_k}{m(b - 4L\eta_k)}.$$

Choose m, η_k so that $\rho_k \leq \rho < 1$ and apply the last inequality recursively, we get

$$\mathbb{E}[f(x_k) - f(x^*)] \leq \rho^k[f(x_0) - f(x^*)]$$

as required. □

3.1. **Linear convergence of stochastic Steffensen method.** With Theorem 3.5, we may deduce the linear convergence of Algorithm 1 as a special case of Algorithm 3 with $b = 1$ (no minibatching) and $\eta_k = \eta_k^{\mathsf{SS}}$ (SSM learning rate).

Lemma 3.6. Let f be as in Assumption 3.2. Then the stochastic Steffensen learning rate

$$\eta_k^{\mathsf{SS}} = \frac{1}{\sqrt{m}} \cdot \frac{\|\nabla f(x_k)\|^2}{\nabla f(x_k)^{\mathsf{T}}(\nabla f(x_k + \nabla f(x_k)) - \nabla f(x_k))}$$

satisfies

$$\frac{1}{\sqrt{m}L} \leq \eta_k^{\mathsf{SS}} \leq \frac{1}{\sqrt{m}\mu}.$$

Proof. Since ∇f is L-Lipschitz, a lower bound is given by

$$\eta_k^{\mathsf{SS}} \geq \frac{1}{\sqrt{m}} \cdot \frac{\|\nabla f(x_k)\|^2}{L\|\nabla f(x_k)\|^2} = \frac{1}{\sqrt{m}L}.$$

The required upper bound follows the μ-strong convexity of f. □

Corollary 3.7 (Linear convergence of SSM)**.** Let f be as in Assumption 3.2 with $x^* \coloneqq \operatorname{argmin}_x f(x)$. If m is chosen so that

$$\rho \coloneqq \frac{(5 + 4/m)\kappa}{\sqrt{m} - 4\kappa} < 1,$$

where $\kappa = L/\mu$ is the condition number, then Algorithm 1 converges linearly in expectation with

$$\mathbb{E}[f(x_k) - f(x^*)] \leq \rho^k[f(x_0) - f(x^*)].$$

Proof. By Theorem 3.5, we have

$$\mathbb{E}[f(x_{k+1}) - f(x^*)] \leq \left[\frac{1}{m\mu\eta_k^{\mathsf{SS}}(1 - 4L\eta_k^{\mathsf{SS}})} + \frac{4(m+1)L\eta_k^{\mathsf{SS}}}{m(1 - 4L\eta_k^{\mathsf{SS}})}\right] \mathbb{E}[f(x_k) - f(x^*)]$$

as long as $\eta_k^{\mathsf{SS}} < 1/(4L)$. Lemma 3.6 shows that this holds for $m > 16\kappa^2$, which follows from $\rho < 1$. The upper and lower bounds in Lemma 3.6 also give

$$\rho_k^{\mathsf{SS}} = \frac{1}{m\mu\eta_k^{\mathsf{SS}}(1 - 4L\eta_k^{\mathsf{SS}})} + \frac{4(m+1)L\eta_k^{\mathsf{SS}}}{m(1 - 4L\eta_k^{\mathsf{SS}})}$$
$$\leq \frac{1}{m\mu\frac{1}{\sqrt{m}L}(1 - 4L\frac{1}{\sqrt{m}\mu})} + \frac{4(m+1)L\frac{1}{\sqrt{m}\mu}}{m(1 - 4L\frac{1}{\sqrt{m}\mu})} = \frac{(5 + 4/m)\kappa}{\sqrt{m} - 4\kappa}.$$

Hence if m is chosen so that $\rho < 1$, we have

$$\mathbb{E}[f(x_k) - f(x^*)] \leq \rho^k [f(x_0) - f(x^*)]$$

as required. □

3.2. **Linear convergence of stochastic Steffensen–Barzilai–Borwein.** The linear convergence of Algorithm 2 likewise follows from Theorem 3.5 with $\eta_k = \eta_k^{\mathsf{SSBB}}$.

Lemma 3.8. Let f be as in Assumption 3.2. Then the stochastic Steffensen–Barzilai–Borwein learning rate

$$\eta_k^{\mathsf{SSBB}} = \frac{1}{\sqrt{m}} \cdot \frac{\beta_k \|\nabla f(x_k)\|^2}{[\nabla f(x_k + \beta_k \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}$$

satisfies

$$\frac{1}{\sqrt{m}L} \leq \eta_k^{\mathsf{SSBB}} \leq \frac{1}{\sqrt{m}\mu}.$$

Proof. Similar to that of Lemma 3.6. □

Corollary 3.9 (Linear convergence of SSBB)**.** Let f be as in Assumption 3.2 with $x^* \coloneqq \operatorname{argmin}_x f(x)$. If m and b are chosen so that

$$\rho \coloneqq \frac{(b + 4/m + 4)\kappa}{\sqrt{m}b - 4\kappa} < 1,$$

where $\kappa = L/\mu$ is the condition number, then Algorithm 2 converges linearly in expectation with

$$\mathbb{E}[f(x_k) - f(x^*)] \leq \rho^k [f(x_0) - f(x^*)].$$

Proof. Because SSBB is a special case of Algorithm 3, then we can easily get

$$\mathbb{E}[f(x_{k+1}) - f(x^*)] \leq \left[\frac{b}{m\mu\eta_k^{\mathsf{SSBB}}(b - 4L\eta_k^{\mathsf{SSBB}})} + \frac{4(m+1)L\eta_k^{\mathsf{SSBB}}}{m(b - 4L\eta_k^{\mathsf{SSBB}})} \right] \mathbb{E}[f(x_k) - f(x^*)]$$

when $\eta_k^{\mathsf{SSBB}} < b/(4L)$ and $\eta_k^{\mathsf{SSBB}} < 1/L$. From Lemma 3.8, this is valid for $m > \max(\kappa^2, 16\kappa^2/b^2)$, which holds because $\rho < 1$. Also, from Lemma 3.8, we have

$$\begin{aligned}
\rho_k^{\mathsf{SSBB}} &= \frac{b}{m\mu\eta_k^{\mathsf{SSBB}}(b - 4L\eta_k^{\mathsf{SSBB}})} + \frac{4(m+1)L\eta_k^{\mathsf{SSBB}}}{m(b - 4L\eta_k^{\mathsf{SSBB}})} \\
&\leq \frac{b}{m\mu\frac{1}{\sqrt{m}L}(b - 4L\frac{1}{\sqrt{m}\mu})} + \frac{4(m+1)L\frac{1}{\sqrt{m}\mu}}{m(b - 4L\frac{1}{\sqrt{m}\mu})} = \frac{(b + 4/m + 4)\kappa}{\sqrt{m}b - 4\kappa}.
\end{aligned}$$

Hence if m and b are chosen so that $\rho < 1$, we have

$$\mathbb{E}[f(x_k) - f(x^*)] \leq \rho^k [f(x_0) - f(x^*)]$$

as required. □

3.3. **Optimal number of stochastic gradient evaluations.** Observe that in the proof of Corollary 3.7 and 3.9, we may replace $1/\sqrt{m}$ by $1/m^p$ for any $p \in (0, 1)$ without affecting the linear convergence conclusion. More precisely, to reach ε-accuracy, the proofs of Corollaries 3.7 and 3.9 show that when we set $b = O(1)$ and $m = O\big(\max(\kappa^{1/p}, \kappa^{1/(1-p)})\big)$, then both SSM and SSBB require evaluation of

$$O\big((n + \max(\kappa^{1/p}, \kappa^{1/(1-p)})) \log(1/\varepsilon)\big)$$

stochastic gradients. Clearly, this is minimized when $p = 1/2$. It follows that for $p = 1/2$, $b = O(1)$, and $m = O(\kappa^2)$, both SSM and SSBB require evaluation of

$$\text{SSM/SSBB:} \qquad O\big((n + \kappa^2) \log(1/\varepsilon)\big),$$

stochastic gradients to reduce to ε-accuracy.

3.4. **Comparison with other methods.** For a theoretical comparison with other methods on equal footing, we will have to limit ourselves to the ones that do not leave the step size η_k unspecified. This automatically excludes SGD and SVRG, which treat η_k as a hyperparameter to be tuned separately. A standard choice is to choose η_k to be the Barzilai–Borwein step size, resulting in the SVRG–BB method [44], which requires evaluation of

$$\text{SVRG–BB:} \qquad O\big((n+\kappa^3)\log(1/\varepsilon)\big)$$

stochastic gradients to achieve ε-accuracy when κ is sufficiently large. On the other hand, the SLBFGS method [21] requires evaluation of

$$\text{SLBFGS:} \qquad O\big((n+\kappa^{2+2(d+h)})\log(1/\varepsilon)\big),$$

stochastic gradients where h is 'history size,' the number of previous updates kept in LBFGS. Evidently both SVRG–BB and SLBFGS are at least an order of magnitude slower than SSM and SSBB as measured by the condition number κ. It is worth noting that for a d-variate objective, the number of stochastic gradients required by SLBFGS depends on d. Our methods, like SVRG–BB, are free of such dependence.

4. Proximal variant

As shown in [45], SGD and SVRG may be easily extended to cover nondifferentiable objective functions of the form

$$F(x) = f(x) + R(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + R(x), \tag{4.1}$$

where f satisfies Assumption 3.2 and R is a nondifferentiable function such as $R(x) = \|x\|_1$. In this section we will see that SSBB may likewise be extended, and the linear convergence is preserved.

To solve (4.1), the proximal gradient method does

$$x_k = \operatorname{prox}_{\eta R}\big(x_{k-1} - \eta\nabla f(x)\big),$$

with a *proximal map* defined by

$$\operatorname{prox}_R(y) = \operatorname*{argmin}_{x\in\mathbb{R}^d}\left\{\frac{1}{2}\|x-y\|^2 + R(x)\right\}.$$

As in [45], we replace the update rule $x_{k,t+1} = x_{k,t} - \eta_k^{\mathsf{SSBB}} v_{k,t}$ in Algorithm 2 by

$$x_{k,t+1} = \operatorname{prox}_{\eta_k^{\mathsf{SSBB}} R}\big(x_{k,t} - \eta_k^{\mathsf{SSBB}} v_{k,t}\big). \tag{4.2}$$

We will see that the resulting algorithm, which we will call prox-SSBB, remains linearly convergent as long as the following assumption holds for some $\mu > 0$.

Assumption 4.1. The function R is μ-strongly convex in the sense that

$$R(y) \geq R(x) + g(x)^{\mathsf{T}}(y-x) + \frac{\mu}{2}\|y-x\|^2$$

for all $x \in \operatorname{dom}(R)$, $g(x) \in \partial R(x)$, $y \in \mathbb{R}^d$, and $R(y) \coloneqq +\infty$ whenever $y \notin \operatorname{dom}(R)$. Here $\partial R(x)$ denotes subgradient at x.

It is a standard fact [38, p. 340] that if R is a closed convex function on $\mathbb{R}^d$, then

$$\|\operatorname{prox}_R(x) - \operatorname{prox}_R(y)\| \leq \|x-y\| \tag{4.3}$$

for all $x, y \in \operatorname{dom}(R)$. We will write μ_f for the convexity parameter of f in Assumption 3.2 and μ_R for that of R in Assumption 4.1. This implies that the overall objective function F is strongly convex with $\mu \geq \mu_f + \mu_R$.

To establish linear convergence for prox-SSBB, we need an analogue of Lemma 3.4, which is provided by [45, Lemma 3], reproduced here for easy reference.

Algorithm 4 Proximal Stochastic Steffensen–Barzilai–Borwein Method (prox-SSBB)

Input: initial state x_0, inner loop size m, minibatch size b, data size n, Barzilai-Borwein step size $\beta_0 = -1$.
for $k = 0, 1, \dots$ **do**
  Compute full gradient $\nabla f(x_k)$.
  if $k > 0$ **then**
    Set $s_k = x_k - x_{k-1}$ and $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$.
    Compute Barzilai–Borwein step size

$$\beta_k = -\frac{\|s_k\|^2}{s_k^{\mathsf{T}} y_k}.$$

  end if
  Compute the stochastic Steffensen–Barzilai–Borwein learning rate

$$\eta_k^{\mathsf{SSBB}} = \frac{1}{\sqrt{m}} \cdot \frac{\beta_k \|\nabla f(x_k)\|^2}{[\nabla f(x_k + \beta_k \nabla f(x_k)) - \nabla f(x_k)]^{\mathsf{T}} \nabla f(x_k)}.$$

  Set $x_{k,0} = x_k$.
  for $t = 0$ **to** $m - 1$ **do**
    Sample minibatch $S_{k,t} \subseteq \{1, \dots, n\}$ uniformly with $|S_{k,t}| = b$.
    Compute variance-reduced stochastic gradient

$$v_{k,t} = \nabla f_{S_{k,t}}(x_{k,t}) - \nabla f_{S_{k,t}}(x_{k,t}) + \nabla f(x_{k,t}).$$

    Update $x_{k,t+1} = \operatorname{prox}_{\eta_k^{\mathsf{SSBB}} R}\big(x_{k,t} - \eta_k^{\mathsf{SSBB}} v_{k,t}\big)$.
  end for
  Set $x_{k+1} = x_{k,i}$ for uniformly chosen $i \in \{0, \dots, m-1\}$.
end for

Lemma 4.2 (Xiao–Zhang)**.** Let f be as in Assumption 3.2, R as in Assumptions 3.2, and $F = f + R$ with $x^* \coloneqq \operatorname{argmin}_x F(x)$. Let $\Delta_{k,t} \coloneqq v_{k,t} - \nabla f(x_{k,t})$ and

$$g_{k,t} \coloneqq \frac{1}{\eta_k}(x_{k,t} - x_{k,t+1}) = \frac{1}{\eta_k}\big(x_{k,t} - \operatorname{prox}_{\eta_k R}(x_{k,t} - \eta_k v_{k,t})\big).$$

If $0 < \eta_k < 1/L$, then

$$\begin{aligned} F(x^*) \geq\ & F(x_{k,t+1}) + g_{k,t}^{\mathsf{T}}(x^* - x_{k,t}) + \frac{\eta_k}{2}\|g_{k,t}\|^2 \\ & + \frac{\mu_f}{2}\|x_{k,t} - x^*\|^2 + \frac{\mu_R}{2}\|x_{k,t+1} - x^*\|^2 + \Delta_{k,t}^{\mathsf{T}}(x_{k,t+1} - x^*). \end{aligned}$$

Corollary 4.3 (Linear convergence of prox-SSBB)**.** Let F and x^* be as in Lemma 4.2 and $\eta_k = \eta_k^{\mathsf{SSBB}}$. Then Corollary 3.9 holds verbatim with F in place of f.

Proof. To apply Lemma 4.2, we need $\eta_k \leq 1/L$ and this holds as we have $m \geq (L/\mu)^2 = \kappa^2$ among the assumptions of Lemma 3.8. In the notations of Lemma 4.2, the update (4.2) is equivalent to $x_{k,t+1} = x_{k,t} - \eta_k g_{k,t}$. So

$$\|x_{k,t+1} - x^*\|^2 = \|x_{k,t} - x^*\|^2 - 2\eta_k g_{k,t}^{\mathsf{T}}(x_{k,t} - x^*) + \eta_k^2 \|g_{k,t}\|^2.$$

By Lemma 4.2, we have

$$-g_{k,t}^{\mathsf{T}}(x_{k,t}-x^*)+\frac{\eta_k}{2}\|g_{k,t}\|^2 \\ \leq F(x^*)-F(x_{k,t+1})-\frac{\mu_f}{2}\|x_{k,t}-x^*\|^2-\frac{\mu_R}{2}\|x_{k,t+1}-x^*\|^2-\Delta_{k,t}^{\mathsf{T}}(x_{k,t+1}-x^*).$$

Therefore,

$$\|x_{k,t+1}-x^*\|^2 \leq \|x_{k,t}-x^*\|^2-2\eta_k\Delta_{k,t}^{\mathsf{T}}(x_{k,t+1}-x^*)+2\eta_k[F(x^*)-F(x_{k,t+1})].$$

We bound the middle term on the right. Let $\bar{x}_{k,t+1} \coloneqq \operatorname{prox}_{\eta_k R}(x_{k,t}-\eta_k\nabla f(x_{k,t}))$. Then

$$\begin{aligned}-2\eta_k\Delta_{k,t}^{\mathsf{T}}(x_{k,t+1}-x^*) &= -2\eta_k\Delta_{k,t}^{\mathsf{T}}(x_{k,t+1}-\bar{x}_{k,t+1})-2\eta_k\Delta_{k,t}^{\mathsf{T}}(\bar{x}_{k,t+1}-x^*)\\ &\leq 2\eta_k\|\Delta_{k,t}\|\|x_{k,t+1}-\bar{x}_{k,t+1}\|-2\eta_k\Delta_{k,t}^{\mathsf{T}}(\bar{x}_{k,t+1}-x^*)\\ &\leq 2\eta_k\|(x_{k,t}-\eta_k v_{k,t})-(x_{k,t}-\eta_k\nabla f(x_{k,t}))\|-2\eta_k\Delta_{k,t}^{\mathsf{T}}(\bar{x}_{k,t+1}-x^*)\\ &= 2\eta_k^2\|\Delta_{k,t}\|^2-2\eta_k\Delta_{k,t}^{\mathsf{T}}(\bar{x}_{k,t+1}-x^*),\end{aligned}$$

where the first inequality is Cauchy–Schwarz and the second follows from Lemma 4.3. The remaining steps are as in the proofs of Theorem 3.5 and Corollary 3.9 with F in place of f. □

5. Numerical Experiments

As mentioned earlier, for smooth objectives, our method of choice is Algorithm 2, the stochastic Steffensen–Barzilai–Borwein method (SSBB) with minibatching. We will compare it with several benchmarking algorithms: stochastic gradient descent (SGD), stochastic variance reduced gradient (SVRG) [17], stochastic LBFGS [21], and the first two with Barzilai–Borwein step size (SGD–BB and SVRG–BB) [44]. For nonsmooth objectives, we compare Algorithm 4, the proximal stochastic Steffensen–Barzilai–Borwein method (prox-SSBB), with proximal variants of the previously mentioned algorithms: prox-SGD, prox-SVRG [45], prox-SLBFGS, and prox-SVRG–BB.

We test these algorithms on popular empirical risk minimization problems — ridge regression, logistic regression, support vector machines with squared hinge loss, l^1-regularized logistic regression — on standard datasets in `LIBSVM`.[1] The parameters involved are summarized in Table 1. Our experiments show that SSBB and prox-SSBB compare favorably with these benchmark algorithms. All our codes are available at https://github.com/Minda-Zhao/stochastic-steffensen.

Data set	Loss function	n	d	m	b	λ_2	λ_1
synthetic	squared loss	10000	100	$4n$	4	10^{-5}	0
`w6a`	logistic loss	17188	300	$2n$	16	10^{-4}	0
`a6a`	squared hinge loss	11220	123	$2n$	16	10^{-3}	0
`w6a`	l^1-regularized logistic loss	17188	300	$2n$	32	10^{-4}	10^{-4}

Table 1. Sample size n, dimension d, batch size b, l_2-regularization parameter λ_2, l_1-regularization parameter λ_1.

For a fair comparison, all algorithms are minibatched. We set a batch size of $b = 4$ for ridge regression, $b = 16$ for logistic loss and squared hinge loss, $b = 32$ for l^1-regularized logistic loss. The inner loop size is set at $m = 2n$ or $4n$. The learning rates in SGD, SVRG, and SLBFGS are hyperparameters that require separate tuning; we pick the best possible values with a grid search. SLBFGS requires more hyperparameters: As suggested by the authors of [21], we set the Hessian update interval to be $L = 10$, Hessian batch size to be $b_H = Lb$, and memory length to be $M = 10$.

[1] https://github.com/cjlin1/libsvm

All experiments are initialized with $x_0 = 0$. We repeat every experiment ten times and report average results.

In all figures, we present the convergence trajectory of each method. The vertical axis represents in log scale the value $f(x_k) - f(x^*)$ where we estimate $f(x^*)$ by running full gradient descent or Newton method multiple times. The horizontal axis represents computational cost as measured by either number of gradient computations divided by n or the actual running time — we present both. In all experiments, we note that the convergence trajectories of SSBB and prox-SSBB agree with the linear convergence established in Sections 3 and 4.

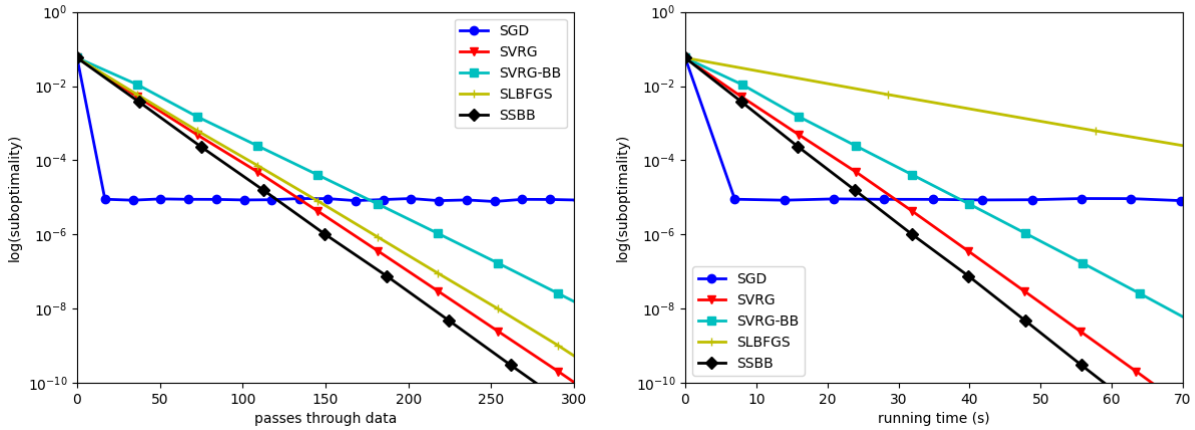

FIGURE 1. Ridge regression on synthetic dataset regularized with $\lambda_2 = 10^{-5}$. *Left:* number of passes through data. *Right:* running time.

5.1. **Ridge Regression.** Figure 1 shows a simple ridge regression on a synthetic dataset generated in a controlled way to give us the true global solution. We generate $x^* \in \mathbb{R}^d$ with $x_i^* \sim \mathcal{N}(0,1)$ and $A \in \mathbb{R}^{n\times d}$ with row vectors $a_1, \dots, a_n \in \mathbb{R}^d$ and entries $a_{ij} \sim \mathcal{N}(0,1)$. We form $y = Ax^* + b$ with b an n-dimensional standard normal variate. We then attempt to recover x^* from A and y by optimizing, with $\lambda_2 = 10^{-5}$,

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}(y_i - a_i^{\mathsf{T}}x)^2 + \frac{\lambda_2}{2}\|x\|_2^2.$$

5.2. **Logistic Regression.** Figure 2 shows the results of a binary classification problem on the `w6a` dataset from `LIBSVM` using an l^2-regularized binary logistic regression. The associated optimization problem with regularization $\lambda_2 = 10^{-4}$ and labels $y_i \in \{-1, +1\}$ is

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}\log\big(1 + e^{-y_i(a_i^{\mathsf{T}}x)}\big) + \frac{\lambda_2}{2}\|x\|_2^2.$$

5.3. **Squared Hinge Loss.** Figure 3 shows the results of a support vector machine classifier with l^2-regularized squared hinge loss and $\lambda_2 = 10^{-3}$ on the `a6a` dataset from `LIBSVM`. The optimization problem in this case is

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}[(1 - y_i a_i^{\mathsf{T}}x)_+]^2 + \frac{\lambda_2}{2}\|x\|_2^2.$$

The results are clear: SSBB solves the problems to high levels of accuracy and is the fastest, whether measured by running time or by number of passes through data, in all experiments. When measured by running times, SLBFGS performs relatively poorly because of the additional computational cost of its matrix-vector products that other methods avoid.

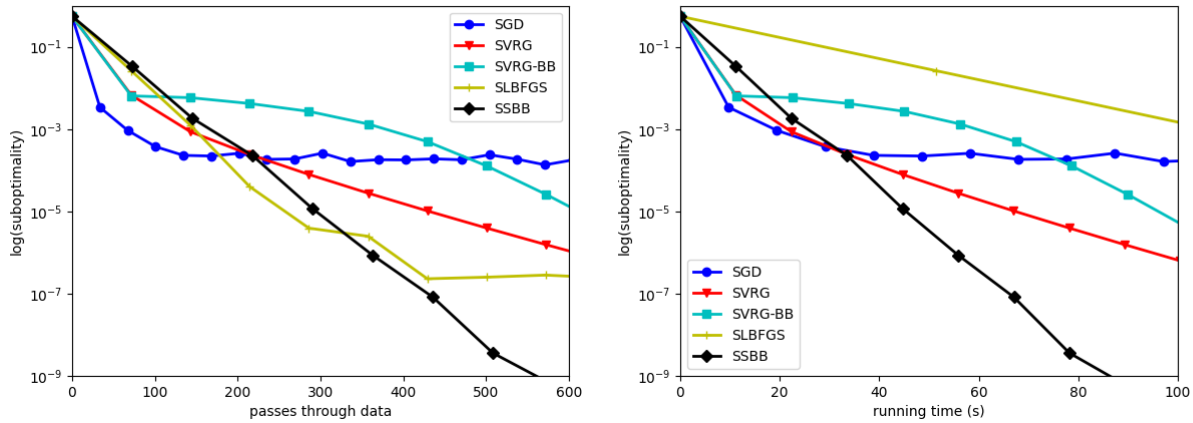

FIGURE 2. l^2-regularized logistic regression on `w6a` dataset from `LIBSVM` regularized with $\lambda_2 = 10^{-4}$. *Left:* number of passes through data. *Right:* running time.

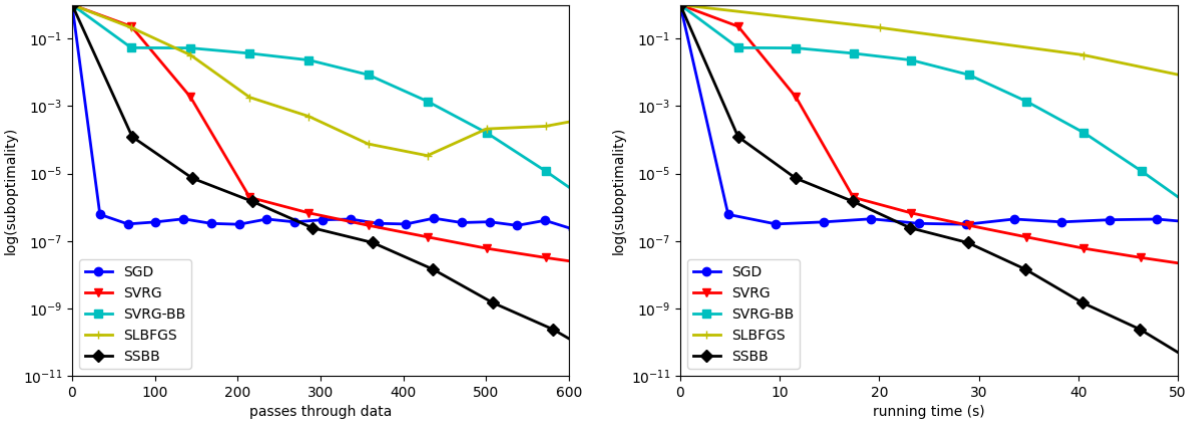

FIGURE 3. l^2-regularized squared hinge loss on `a6a` from `LIBSVM` regularized with $\lambda_2 = 10^{-3}$. *Left:* number of passes through data. *Right:* running time.

5.4. **Proximal Variant.** Figure 4 shows the results of a binary classification problem on the `w6a` dataset from `LIBSVM` using a binary logistic regression with both l^2- and l^1-regularizations, a problem considered in [45]:

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \log\bigl(1+e^{-y_i(a_i^\top x)}\bigr) + \frac{\lambda_2}{2}\|x\|_2^2 + \lambda_1\|x\|_1.$$

We set regularization parameters to be $\lambda_2 = 10^{-4}$, $\lambda_1 = 10^{-4}$.

The results obtained are consistent with those in Sections 5.1–5.3, demonstrating that prox-SSBB solves the problem to high levels of accuracy and is the fastest among all algorithms compared, whether measured by running time or by the number of passes through data.

6. Conclusion

The stochastic Steffensen methods introduced in this article are (i) simple to implement, (ii) efficient to compute, (iii) easy to incorporate, (iv) tailored for massive data and high dimensions, have (v) minimal memory requirements and (vi) a negligible number of hyperparameters to tune. The last point is in contrast to more sophisticated methods involving moments [8, 20] or momentum

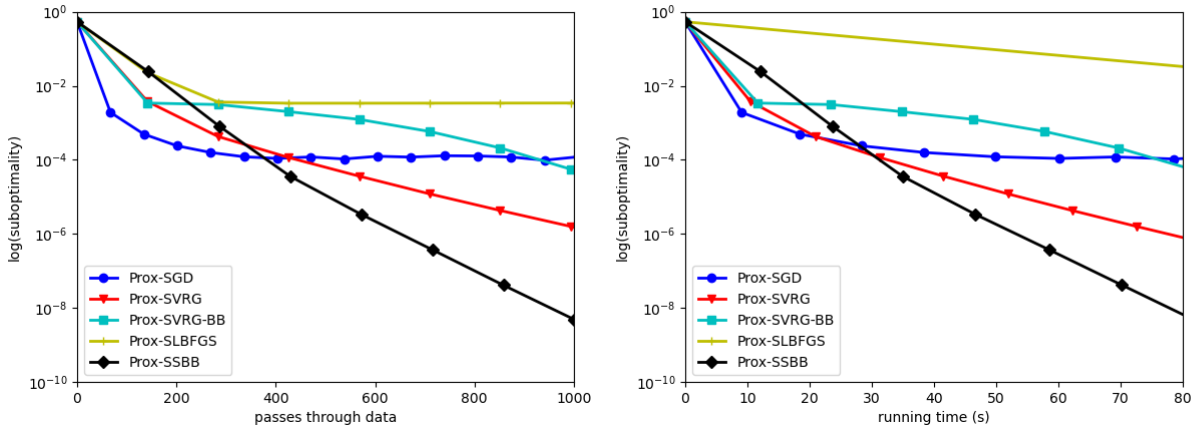

FIGURE 4. Logistic regression with l^2 and l^1 regularizations on `w6a` dataset from `LIBSVM` regularized with $\lambda_2 = 10^{-4}$ and $\lambda_1 = 10^{-4}$. *Left:* number of passes through data. *Right:* running time.

[24, 33, 35, 36], which require heavy tuning of many more hyperparameters. SSM and SSBB require just two — minibatch size b and inner loop size m.

The point (iii) also deserves special mention. Since SSM and SSBB are ultimately encapsulated in the respective learning rates η_k^{SS} and η_k^{SSBB}, they are versatile enough to be incorporated into other methods such as those in [8, 20, 24, 33, 35, 36], assuming that we are willing to pay the price in hyperparameters tuning. We hope to explore this in future work.

Acknowledgment. This work is partially supported by DARPA HR00112190040, NSF DMS-1854831, NSF ECCS-2216912, ONR N000142312863, and the Eckhardt Faculty Fund. LHL thanks Junjie Yue for helpful discussions.

Data availability and conflict of interest. We do not analyze or generate any datasets. The numerical experiments in Section 5 rely on standard datasets in `LIBSVM` that is publicly available from `https://github.com/cjlin1/libsvm`. The authors declare no conflict of interest.

REFERENCES

[1] S. Amat, J. A. Ezquerro, and M. A. Hernández-Verón. On a Steffensen-like method for solving nonlinear equations. *Calcolo*, 53(2):171–188, 2016.
[2] R. Babanezhad, M. O. Ahmed, A. Virani, M. Schmidt, J. Konečný, and S. Sallinen. Stopwasting my gradients: Practical SVRG. In *Advances in Neural Information Processing Systems 28*, pages 2251–2259, 2015.
[3] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA J. Numer. Anal.*, 8(1):141–148, 1988.
[4] C. Brezinski and M. Redivo-Zaglia. *Extrapolation and rational approximation—the works of the main contributors*. Springer, Cham, 2020.
[5] C. G. Broyden. The convergence of a class of double-rank minimization algorithms. II. The new algorithm. *J. Inst. Math. Appl.*, 6:222–231, 1970.
[6] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM J. Optim.*, 26(2):1008–1031, 2016.
[7] A. Defazio, F. R. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27*, pages 1646–1654, 2014.
[8] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.
[9] J. A. Ezquerro, M. A. Hernández-Verón, M. J. Rubio, and A. I. Velasco. An hybrid method that improves the accessibility of Steffensen's method. *Numer. Algorithms*, 66(2):241–267, 2014.
[10] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, 1970.
[11] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. SIAM, 2019.

[12] D. Goldfarb. A family of variable-metric methods derived by variational means. *Math. Comp.*, 24:23–26, 1970.
[13] P. Henrici. *Elements of numerical analysis*. John Wiley & Sons, Inc., New York-London-Sydney, 1964.
[14] A. S. Householder. *The numerical treatment of a single nonlinear equation*. International Series in Pure and Applied Mathematics. McGraw-Hill, New York, 1970.
[15] H. Y. Huang. Unified approach to quadratically convergent algorithms for function minimization. *J. Optim. Theory Appl.*, 5:405–423, 1970.
[16] L. W. Johnson and D. R. Scholz. On Steffensen's method. *SIAM J. Numer. Anal.*, 5:296–302, 1968.
[17] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pages 315–323, 2013.
[18] S. Kaczmarz. Angenäherte auflösung von systemen linearer gleichungen. *Bull. Internat. Acad. Polon. Sci. A*, 57(6):355–357, 1937.
[19] S. Kaczmarz. Approximate solution of systems of linear equations. *Internat. J. Control*, 57(6):1269–1271, 1993.
[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.
[21] P. Moritz, R. Nishihara, and M. I. Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 249–258, 2016.
[22] G. H. Nedzhibov. An approach to accelerate iterative methods for solving nonlinear operator equations. In *Applications of mathematics in engineering and economics (AMEE'11)*, volume 1410 of *AIP Conf. Proc.*, pages 76–82. Amer. Inst. Phys., Melville, NY, 2011.
[23] D. Needell, N. Srebro, and R. Ward. Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. *Math. Program.*, 155(1-2, Ser. A):549–573, 2016.
[24] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269(3):543–547, 1983.
[25] Y. Nievergelt. Aitken's and Steffensen's accelerations in several variables. *Numer. Math.*, 59(3):295–310, 1991.
[26] Y. Nievergelt. The condition of Steffensen's acceleration in several variables. *J. Comput. Appl. Math.*, 58(3):291–305, 1995.
[27] A. Nitanda. Accelerated stochastic gradient descent for minimizing finite sums. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 195–203, 2016.
[28] T. Noda. The Aitken-Steffensen method in the solution of simultaneous nonlinear equations. *Sūgaku*, 33(4):369–372, 1981.
[29] T. Noda. The Aitken-Steffensen method in the solution of simultaneous nonlinear equations. II. *Sūgaku*, 38(1):83–85, 1986.
[30] T. Noda. The Aitken-Steffensen method in the solution of simultaneous nonlinear equations. III. *Proc. Japan Acad. Ser. A Math. Sci.*, 62(5):174–177, 1986.
[31] T. Noda. The Aitken-Steffensen formula for systems of nonlinear equations. IV. *Proc. Japan Acad. Ser. A Math. Sci.*, 66(8):260–263, 1990.
[32] T. Noda. The Aitken-Steffensen formula for systems of nonlinear equations. V. *Proc. Japan Acad. Ser. A Math. Sci.*, 68(2):37–40, 1992.
[33] B. T. Poljak. Some methods of speeding up the convergence of iterative methods. *Ž. Vyčisl. Mat i Mat. Fiz.*, 4:791–803, 1964.
[34] F. A. Potra. On an iterative algorithm of order $1.839\cdots$ for solving nonlinear operator equations. *Numer. Funct. Anal. Optim.*, 7(1):75–106, 1984/85.
[35] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.
[36] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of Adam and beyond. *arXiv:1904.09237*, 2019.
[37] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Statistics*, 22:400–407, 1951.
[38] R. T. Rockafellar. *Convex analysis*. Princeton Landmarks in Mathematics. Princeton University Press, Princeton, NJ, 1997.
[39] N. L. Roux, M. Schmidt, and F. R. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems 25*, pages 2672–2680, 2012.
[40] D. F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Math. Comp.*, 24:647–656, 1970.
[41] J. F. Steffensen. Remarks on iteration. *Skand. Aktuarietidskr.*, 1:64–72, 1933.
[42] J. F. Steffensen. Further remarks on iteration. *Skand. Aktuarietidskr.*, 28:44–55, 1945.
[43] T. Strohmer and R. Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *J. Fourier Anal. Appl.*, 15(2):262–278, 2009.
[44] C. Tan, S. Ma, Y. Dai, and Y. Qian. Barzilai-borwein step size for stochastic gradient descent. In *Advances in Neural Information Processing Systems 29*, pages 685–693, 2016.

[45] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM J. Optim.*, 24(4):2057–2075, 2014.
[46] R. Zhao, W. B. Haskell, and V. Y. Tan. Stochastic L-BFGS: Improved convergence rates and practical acceleration strategies. *IEEE Transactions on Signal Processing*, 66(5):1155–1169, 2018.

Computational and Applied Mathematics, University of Chicago, Chicago, IL 60637
Email address: mindazhao@uchicago.edu, laizehua@uchicago.edu, lekheng@uchicago.edu