STOCHASTIC GRADIENT METHODS WITH PRECONDITIONED UPDATES

Anonymous authors
Paper under double-blind review

ABSTRACT

This work considers non-convex finite sum minimization. There are a number of algorithms for such problems, but existing methods often work poorly when the problem is badly scaled and/or ill-conditioned, and a primary goal of this work is to introduce methods that alleviate this issue. Thus, here we include a preconditioner that is based upon Hutchinson's approach to approximating the diagonal of the Hessian, and couple it with several gradient based methods to give new 'scaled' algorithms: `Scaled SARAH` and `Scaled L-SVRG`. Theoretical complexity guarantees under smoothness assumptions are presented, and we prove linear convergence when both smoothness and the PL-condition is assumed. Because our adaptively scaled methods use approximate partial second order curvature information, they are better able to mitigate the impact of badly scaled problems, and this improved practical performance is demonstrated in the numerical experiments that are also presented in this work.

1 INTRODUCTION

This work considers the following, possibly nonconvex, finite-sum optimization problem:

$$\min_{w\in\mathbb{R}^d} \left\{ P(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) \right\}, \tag{1}$$

where $w \in \mathbb{R}^d$ is the model/weight parameter and the loss functions $f_i : \mathbb{R}^d \to \mathbb{R}\ \forall i \in [n] := \{1 \dots n\}$ are smooth and twice differentiable. Throughout this work it is assumed that (1) has an optimal solution, with a corresponding optimal value, denoted by w^*, and $P^* = P(w^*)$, respectively.

Problems of the form (1) cover a plethora of applications, including empirical risk minimization, deep learning, and supervised learning tasks such as regularized least squares or logistic regression (Shalev-Shwartz & Ben-David, 2014). This minimization problem can be difficult to solve, particularly when the number of training samples n, or problem dimension d, is large, or if the problem is nonconvex.

Stochastic Gradient Descent (`SGD`) is one of the most widely known methods for problem (1), and its origins date back to the 1950s with the work of Robbins & Monro (1951). The explosion of interest in machine learning has led to an immediate need for reliable and efficient algorithms for solving (1). Motivated by, and aiming to improve upon, vanilla `SGD`, many novel methods have already been developed for convex and/or strongly convex instances of (1), including `SAG`/`SAGA` (Le Roux et al., 2012; Defazio et al., 2014), `SDCA` (Shalev-Shwartz & Zhang, 2013), `SVRG` (Johnson & Zhang, 2013; Xiao & Zhang, 2014), `S2GD` (Konečný & Richtárik, 2017) and `SARAH` (Nguyen et al., 2017), to name just a few. In general, these methods are simple, have low per iteration computational costs, and are often able to find an ε-optimal solution to (1) quickly, when $\varepsilon > 0$ is not too small. However, they often have several hyper-parameters that can be difficult to tune, they can struggle when applied to ill-conditioned problems, and many iterations may be required to find a high accuracy solution.

Non-convex instances of the optimization problem (1) (for example, arising from deep neural networks (DNNS)) have been diverting the attention of researchers of late, and new algorithms are being developed to fill this gap (Ghadimi & Lan, 2013; Ghadimi et al., 2016; Lei et al., 2017; Li et al., 2021b). Of particular relevance to this work is the `PAGE` algorithm presented in Li et al. (2021a). The algorithm is conceptually simple, involving only one loop, and a small number of parameters, and can be applied to non-convex problems (1). The main update involves either a minibatch `SGD` direction, or the previous gradient with a small adjustment (similar to that in `SARAH` (Nguyen et al., 2017)). The `Loopless SVRG` (`L-SVRG`) method (Hofmann et al., 2015; Qian et al., 2021), is also of particular interest here. It is a simpler 'loopless' variant of `SVRG`, which, unlike for `PAGE`, involves

an unbiased estimator of the gradient, and it can be applied to non-convex instances of problem (1).

For problems that are poorly scaled and/or ill-conditioned, second order methods that incorporate curvature information, such as Newton or quasi-Newton methods (Dennis Jr. & Moré, 1977; Fletcher, 1987; Nocedal & Wright, 2006), can often outperform first order methods. Unfortunately, they can also be prohibitively expensive, in terms of both computational and storage costs. There are several works that have tried to reduce the potentially high cost of second order methods by using only approximate, or partial curvature information. Some of these stochastic second order, and quasi-Newton (Jahani et al., 2021a; 2020) methods have shown good practical performance for some machine learning problems, although, possibly due to the noise in the Hessian approximation, sometimes they perform similarly to first order variants.

An alternative approach to enhancing search directions is to use a *preconditioner*. There are several methods for problems of the form (1), which use what we call a 'first order preconditioner' — a preconditioner built using gradient information — including `Adagrad` (Duchi et al., 2011), RMSProp (Tieleman et al., 2012), and `Adam` (Kingma & Ba, 2015). `Adagrad` (Duchi et al., 2011) incorporates a diagonal preconditioner that is built using accumulated gradient information from the previous iterates. The preconditioner allows every component of the current gradient to be scaled adaptively, but it has the disadvantage that the elements of the preconditioner tend to grow rapidly as iterations progress, leading to a quickly decaying learning rate. A method that maintains the ability to adaptively scale elements of the gradient, but overcomes the drawback of a rapidly decreasing learning rate, is RMSProp. It does this by including a momentum parameter, β_2 in the update for the diagonal preconditioner. In particular, at each iteration the updated diagonal preconditioner is taken to be a convex combination (using a momentum parameter β_2) of the (square) of the previous preconditioner and the Hadamard product of the current gradient with itself. So, gradient information from all the previous iterates is included in the preconditioner, but there is a preference for more recent information. `Adam` (Kingma & Ba, 2015) combines the positive features of `Adagrad` and RMSProp, but it also uses a first moment estimate of the gradient, providing a kind of additional momentum. `Adam` preforms well in practice, and is among the most popular algorithms for DNN.

Recently, second order preconditioners that use approximate and/or partial curvature information have been developed and studied. The approach in `AdaHessian` (Yao et al., 2020) was to use a diagonal preconditioner that was motivated by Hutchinson's approximation to the diagonal of the Hessian (Bekas et al., 2007), but that also stayed close to some of the approximations used in existing methods such as `Adam` (Kingma & Ba, 2015) and `Adagrad` (Duchi et al., 2011). Because of this, the approximation often differed markedly from the true diagonal of the Hessian, and therefore it did not always capture good enough curvature information to be helpful. The work `OASIS` (Jahani et al., 2021b), proposed a preconditioner that was closely based upon Hutchinson's approach, and provided a more accurate estimation of the diagonal of the Hessian, and correspondingly led to improved numerical behaviour in practice. The preconditioner presented in Jahani et al. (2021b) is adopted here.

1.1 Notation and Assumptions

Given a Positive Definite (PD) matrix $D \in \mathbb{R}^{d\times d}$, the weighted Euclidean norm is defined to be $\|x\|_D^2 = x^T Dx$, where $x \in \mathbb{R}^d$. The symbol $\odot$ denotes the Hadamard product, and $\mathtt{diag}(x)$ denotes the $d \times d$ diagonal matrix whose diagonal entries are the components of the vector $x \in \mathbb{R}^d$.

Recall that problem (1) is assumed to have an optimal (probably not a unique) solution w^*, with corresponding optimal value $P^* = P(w^*)$. As is standard for stochastic algorithms, the convergence guarantees presented in this work will develop a bound on the number of iterations T, required to push the expected squared norm of the gradient below some error tolerance $\varepsilon > 0$, i.e., to find a $\hat{w}_T$ satisfying

$$\mathbb{E}[\|\nabla P(\hat{w}_T)\|_2^2] \leq \varepsilon^2. \tag{2}$$

A point $\hat{w}_T$ satisfying (2) is referred to as an ε-optimal solution. Importantly, $\hat{w}_T$ is *some* iterate generated in the first T iterations of each algorithm, but it is *not necessarily* the Tth iterate.

Throughout this work we assume that each $f_i : \mathbb{R}^d \to \mathbb{R}$ and $P : \mathbb{R}^d \to \mathbb{R}$ are twice differentiable and also L-smooth. This is formalized in the following assumption.

Assumption 1.1 (L-smoothness)**.** For all $i \in [n]$ f_i and P are assumed to be twice differentiable and L-smooth, i.e., $\forall i \in [n], \forall w, w' \in \mathrm{dom}(f_i)$ we have $\|\nabla f_i(w) - \nabla f_i(w')\| \leq L\|w - w'\|$, and $\forall w, w' \in \mathrm{dom}(P)$ we have $\|\nabla P(w) - \nabla P(w')\| \leq L\|w - w'\|$.

For some of the results in this work, it will also be assumed that the function P satisfies the PL-condition. Note that the PL-condition does not imply convexity (see Footnote 1 in Li et al. (2021a)).

Assumption 1.2 (Polyak-Łojasiewicz-condition)**.** A function $P : \mathbb{R}^d \to \mathbb{R}$ satisfies the PL-condition if there exists $\mu > 0$, such that $\|\nabla P(w)\|^2 \geq 2\mu(P(w) - P^*),\ \forall w \in \mathbb{R}^d$.

1.2 Contributions

The main contributions of this work are stated below, and are summarized in Tables 1 and 2.

Table 1: Comparison of scaled stochastic methods for non-convex problems. *Notation:* ε denotes solution accuracy (2). The 'Tuning of β_2' column shows whether it is easy ('+'), or difficult ('−') to tune β_2. Our preconditioner (5) works with any β_2. `Adam` only supports certain large $\beta \approx 1$ (Défossez et al., 2020; Reddi et al., 2019).

Method	Reference	Convergence	Tuning of β_2
`Adagrad`	(Duchi et al., 2011) (Zou et al., 2018) (Défossez et al., 2020)	ε^{-4}	+
`RMSProp`	(Tieleman et al., 2012)	no theory	
`Adam`	(Kingma & Ba, 2015) (Défossez et al., 2020)	ε^{-4}	−
`AdaHessian`	(Yao et al., 2020)	no theory	
`OASIS`	(Jahani et al., 2021b)	ε^{-4}	+
`Scaled SARAH`	This work	ε^{-2}	+
`Scaled L-SVRG`	This work	ε^{-2}	+

Table 2: A summary of the main results of this work. Complexities for `Scaled SARAH` and `Scaled L-SVRG` are given for non-convex problems (first row), and under the PL assumption (second row). *Notation:* L = smoothness constant, μ = PL constant, ε = solution accuracy (2), $\Delta_0 = P(w^0) - P^*$, n = data size, and Γ, α are upper and lower bounds of the Hessian approximation.

	`Scaled SARAH`	`Scaled L-SVRG`
NC	$\mathcal{O}\left(\frac{\Gamma}{\alpha}\frac{\sqrt{n}L\Delta_0}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{\Gamma}{\alpha}\frac{n^{2/3}L\Delta_0}{\varepsilon^2}\right)$
PL	$\mathcal{O}\left(\max\left\{n, \frac{\Gamma}{\alpha}\sqrt{n}\frac{L}{\mu}\right\}\log\frac{\Delta_0}{\varepsilon}\right)$	$\mathcal{O}\left(\max\left\{n, \frac{\Gamma}{\alpha}n^{2/3}\frac{L}{\mu}\right\}\log\frac{\Delta_0}{\varepsilon}\right)$

• **Scaled SARAH.** We present a new algorithm called `Scaled SARAH`, which is a combination of the `SARAH` (Nguyen et al., 2017) and `PAGE` (Li et al., 2021a) algorithms, coupled with the diagonal preconditioner from Jahani et al. (2021b). The inclusion of the preconditioner results in adaptive scaling of every element of the search direction (negative gradient), which leads to improved practical performance, particularly on ill-conditioned and poorly scaled problems. The algorithm is simple (a single loop) and is easy to tune (Section 3).

• **Scaled L-SVRG.** The `Scaled L-SVRG` algorithm is also presented, which is similar to `L-SVRG`, but with the addition of the diagonal preconditioner (Jahani et al., 2021b). Again, the preconditioner allows all elements of the gradient to be scaled adaptively, the algorithm uses a single loop structure, and for this algorithm an unbiased estimate of the gradient is used. The inclusion of adaptive local curvature information via the preconditioner leads to improvments in practical performance (Sections 4).

• **Convergence guarantees.** Theoretical guarantees show that both `Scaled SARAH` and `Scaled L-SVRG` converge and we present an explicit bound for the number of iterations required by each algorithm to obtain an iterate that is ϵ-optimal. Convergence is guaranteed for both `Scaled SARAH` and `Scaled L-SVRG` under a smoothness assumption on the functions f_i. If both smoothness and the PL-condition hold, then improved iteration complexity results for `Scaled SARAH` and `Scaled L-SVRG` are obtained, which show that expected function value gap converges to zero at a linear rate (see Theorems 3.2 and 4.2.) Our scaled methods achieve the best known rates for non-convex stochastic problems, and `Scaled SARAH` and `Scaled L-SVRG` are the first preconditioned methods that achieve a linear rate of convergence under the PL assumption. (See Sections 4.)

• **Numerical experiments.** Extensive numerical experiments were performed (Sections 5 and A) under various parameter settings to investigate the practical behaviour of our new scaled algorithms. The inclusion of preconditioning in `Scaled SARAH` and `Scaled L-SVRG` led to improvements in performance compared with no preconditioning in several of the experiments, and `Scaled SARAH` and `Scaled L-SVRG` were competitive with, and often outperformed, `Adam`.

Paper outline. This paper is organised as follows. In Section 2 we describe the diagonal preconditioner that will be used in this work. In Section 3, we describe a new `Scaled PAGE` algorithm and present theoretical convergence guarantees. In Section 4, we introduce the `Scaled L-SVRG` algorithm, which adapts the `L-SVRG` algorithm to include a preconditioner. We present numerical experiments demonstrating the practical performance of our proposed methods in Section 5.

and concluding remarks are given in Section 6. All proofs (Sections C and D), additional numerical experiments (Section A), and further details and discussion can be found in the appendix.

2 DIAGONAL PRECONDITIONER

In this section we describe the diagonal preconditioner that is used in this work. The paper of Bekas et al. (2007) described Hutchinson's approximation to the diagonal of the Hessian, and this provided motivation for the diagonal preconditioner proposed in Jahani et al. (2021b), which is adopted here. In particular, given an initial approximation D_0, (to be described soon), and Hessian approximation momentum parameter $\beta \in (0, 1)$ (equivalent to the second moment hyperparameter, β_2 in `Adam` (Kingma & Ba, 2015)), for all $t \geq 1$,

$$D_t = \beta D_{t-1} + (1-\beta)\texttt{diag}\left(z_t \odot \nabla^2 P_{\mathcal{J}_t}(w_t) z_t\right), \tag{3}$$

where z_t is a random vector with Rademacher distribution[1], $\mathcal{J}_t$ is an index set randomly sampled from $[n]$, and

$$\nabla^2 P_{\mathcal{J}_t}(w_t) = \frac{1}{|\mathcal{J}_t|} \sum_{j \in \mathcal{J}_t} \nabla^2 f_j(w_t). \tag{4}$$

Finally, for $\alpha > 0$ (where the parameter $\alpha > 0$ is equivalent to the parameter ϵ in `Adam` (Kingma & Ba, 2015) and `AdaHessian` (Yao et al., 2020)), the diagonal preconditioner is:

$$\left(\hat{D}_t\right)_{i,i} = \max\{\alpha, |D_t|_{i,i}\}. \tag{5}$$

The expression (5) ensures that the preconditioner $\hat{D}_t$ is always PD, so it is well-defined and results in a descent direction. The absolute values are necessary because the objective function is potentially nonconvex, so the batch Hessian approximation could be indefinite. In fact, even if the Hessian is PD, D_t in (3) may still contain negative elements due to the sampling strategy used.

The preconditioner (5) is a good estimate of the diagonal of the (batch) Hessian because Hutchinson's updating formula (3) is used (see also Figures 12, 13). Hence, it captures accurate curvature information, which is helpful for poorly scaled and ill-conditioned problems. Because the preconditioner is diagonal it is easy and inexpensive to apply it's inverse, and the associated storage costs are low.

The preconditioner (3)+(5) depends on the parameter β: if $\beta = 1$ then the preconditioner is fixed for all iterations, whereas if $\beta = 0$ then the preconditioner is simply a kind of sketched batch Hessian. Taking $0 < \beta < 1$ gives a convex combination of the previous approximation and the current approximation, thereby ensuring that the entire history is included in the preconditioner, but is damped by β, and the most recent information is also present. See Section E for a detailed discussion of the choice of β.

The main computational cost of the approximation in (3) is the (batch) Hessian-vector product $\nabla^2 P_{\mathcal{J}_t}(w_t) z_t$. Fortunately, this can be efficiently calculated using two rounds of back propagation. Moreover, the preconditioner is matrix-free, simply needing an oracle to return the Hessian vector product, but it does not need explicit access to the batch Hessian itself; see Appendix B in Jahani et al. (2021b). Therefore, the costs (both computational and storage) for this preconditioner are not burdensome.

As previously mentioned, the approximation (3) requires an initial estimate D_0 of the diagonal of the Hessian, and this is critical to the success of the preconditioner. In particular, one must take

$$D_0 = \frac{1}{m} \sum_{j=1}^{m} \texttt{diag}\left(z_j \odot \nabla^2 P_{\mathcal{J}_j}(w_0) z_j\right), \tag{6}$$

where $\mathcal{J}_j$ denotes sampled batches and the vectors z_j are generated from a Radermacher distribution. This ensures that $\hat{D}_t$ does indeed approximate the diagonal of the Hessian; see Section 3.3 in Jahani et al. (2021b).

The following remark confirms that the diagonal preconditioner is both PD and bounded.

Lemma 2.1 (See Remark 4.10 in Jahani et al. (2021b)). *For any $t \geq 1$, we have $\alpha I \preccurlyeq \hat{D}_t \preccurlyeq \Gamma I$, where $0 < \alpha \leq \Gamma = \sqrt{d}L$.*

[1] i.e., the components of the z_t are ± 1 with equal probability.

Note that Remark 4.10 is proved incorrectly in Jahani et al. (2021b).

3 Scaled SARAH

Here we propose a new algorithm, Scaled SARAH, for finite sum optimization (1). Our algorithm is similar to the SARAH algorithm (Nguyen et al., 2017) and the PAGE algorithm (Li et al., 2021a), but a key difference is that Scaled SARAH includes the option of a preconditioner, $\hat{D}_t$ for all $t \geq 0$, with a preconditioned approximate gradient step. Scaled SARAH is presented now as Algorithm 1.

Algorithm 1 Scaled SARAH

1: **Input:** initial point w_0, learning rate η, preconditioner $\hat{D}_0$, probability p
2: $v_0 = \nabla P(w_0)$
3: **for** $t = 0, 1, 2, \ldots$ **do**
4: $\quad w_{t+1} = w_t - \eta \hat{D}_t^{-1} v_t$
5: $\quad$ Generate independently batches i_{t+1} for v_{t+1} and $\mathcal{J}_t$ for $\hat{D}_{t+1}$
6: $\quad v_{t+1} = \begin{cases} \nabla P(w_{t+1}), & \text{with probability } p \\ v_t + \nabla f_{i_{t+1}}(w_{t+1}) - \nabla f_{i_{t+1}}(w_t), & \text{with probability } 1-p \end{cases}$
7: $\quad$ Update the preconditioner $\hat{D}_{t+1}$
8: **end for**
9: **Output:** $\hat{w}_T$ chosen uniformly from $\{w_t\}_{t=0}^T$

In each iteration of Algorithm 1 an update is computed in Step 4. The point w_t is adjusted by taking a step in the direction $\hat{D}_t^{-1} v_t$, of fixed step size η. The vector v_t approximates the gradient, and the preconditioner scales that direction. A key difference between Scaled SARAH and PAGE/SARAH is the inclusion of the preconditioner $\hat{D}_t^{-1}$ in this step.

Step 6 defines the next gradient estimator v_{t+1}, for which there are two options. With probability p the full gradient is used. Alternately, with probability $1-p$, the new gradient estimate is the previous gradient approximation v_t, with an adjustment term that involves the difference between the gradient of f_i evaluated at w_{t+1} and at w_t. The search direction computed in Scaled SARAH contains gradient information, while the preconditioner described in Section 2 contains approximate second order information. When this preconditioner is applied to the gradient estimate, each dimension is scaled adaptively depending on the corresponding curvature. Intuitively, this amplifies dimensions with low curvature (shallow loss surfaces), while damping directions with high curvature (sharper loss surfaces). The aim is for $\hat{D}_t^{-1} v_t$ to point in a better, adjusted direction, compared with v_t.

Scaled SARAH is a single loop algorithm so it is conceptually simple. If $p = 1$ then the algorithm always picks the first option in Step 6, so that Scaled SARAH reduces to a preconditioned GD method. On the other hand, if $p = 0$, then only the second option in Step 6 is used.

Notice that Scaled SARAH is a combination of both the PAGE and SARAH algorithms, coupled with a preconditioner. SARAH (Nguyen et al., 2017) is a double loop algorithm, where the inner loop is defined in the same way as update in Step 6. PAGE (Li et al., 2021a) is based upon SARAH, but PAGE uses a single loop structure, and allows for minibatches to be used in the gradient approximation v_{t+1} (rather than the single component as in Step 6).[2] Scaled SARAH shares the same single loop structure as PAGE, but also shares the same single component update for the gradient estimator as SARAH (no minibatches). However, different from both PAGE and SARAH, Scaled SARAH uses a preconditioner in Step 4.

In the remainder of this work we focus on a particular instance of Scaled SARAH, which uses a fixed probability $p_t = p$, and uses the diagonal preconditioner presented in Section 2. These choices have been made because a central goal of this work is to understand the impact that a well chosen preconditioner has on poorly scaled problems. Convergence guarantees and the results of numerical experiments, will be presented using this set up.

Theoretical results for Scaled SARAH are presented now. In particular, we present complexity bounds on the number of iterations required by Scaled SARAH to obtain an ε-optimal solution for the non-convex problem (1) (recall Section 1.1 and (2)). The first result holds under Assumption 1.1,

[2] Note that, while PAGE allows minibatches for either option in the update Step 6, most of the theoretical results presented in Li et al. (2021a) require the full gradient to be computed as the first option in Step 6.

while the second theorem holds under both smoothness *and* PL assumptions. First, we define the following step-size bound:

$$\bar{\eta} = \frac{\alpha}{L\left(1+\sqrt{\frac{1-p}{p}}\right)}. \tag{7}$$

Theorem 3.1. *Suppose that Assumption 1.1 holds, let $\epsilon > 0$, let p denote the probability, and let the step-size satisfy $\eta \leq \bar{\eta}$ (7). Then, the number of iterations performed by* `Scaled SARAH`*, starting from an initial point $w_0 \in \mathbb{R}^d$ with $\Delta_0 = P(w_0) - P^*$, required to obtain an ε-approximate solution of the non-convex finite-sum problem (1) can be bounded by*

$$T = \mathcal{O}\left(\frac{\Gamma}{\alpha}\frac{\Delta_0 L}{\varepsilon^2}\left(1+\sqrt{\frac{1-p}{p}}\right)\right).$$

Theorem 3.2. *Suppose that Assumptions 1.1 and 1.2 hold, let $\epsilon > 0$, and let the step-size satisfy $\eta \leq \bar{\eta}$ (7). Then the number of iterations performed by* `Scaled SARAH` *sufficient for finding an ε-approximate solution of non-convex finite-sum problem (1) can be bounded by*

$$T = \mathcal{O}\left(\max\left\{\frac{1}{p}, \frac{L}{\mu}\frac{\Gamma}{\alpha}\left(1+\sqrt{\frac{1-p}{p}}\right)\right\}\log\frac{\Delta_0}{\varepsilon}\right).$$

Note that this last theorem shows that `Scaled SARAH` exhibits a linear rate of convergence under both the smoothness assumption and the PL-condition.

4 `Scaled L-SVRG`

`SVRG` (Johnson & Zhang, 2013; Xiao & Zhang, 2014) is a variance reduced stochastic gradient method that is very popular for finite sum optimization problems. However, the algorithm has a double loop structure, and careful tuning of hyper-parameters is required for good practical performance.

Recently, Hofmann et al. (2015) proposed a Loopless `SVRG` (`L-SVRG`) variant, that has a simpler, single loop structure, which can be applied to problem (1) in the convex and smooth case. This was extended in Qian et al. (2021) to cover the composite case with an arbitrary sampling scheme. With its single loop structure, and consequently fewer hyperparameters to tune, coupled with the fact that, unlike for `PAGE` (recall Section 1), `L-SVRG` uses an *unbiased* estimate of the gradient, `L-SVRG` is a versatile and competitive algorithm for problems of the form (1).

However, as for the other previously mentioned gradient based methods, `L-SVRG` can perform poorly when the problem is badly scaled and/or ill-conditioned. This provides the motivation for the `Scaled L-SVRG` method that we propose in this work. Our `Scaled L-SVRG` algorithm combines the positive features of `L-SVRG`, with a preconditioner, to give a method that is loopless, has few hyperparameters to tune, uses an unbiased estimate of the gradient, and adaptively scales the search direction depending upon the local curvature. The `Scaled L-SVRG` method is presented now as Algorithm 2.

Algorithm 2 `Scaled L-SVRG`

Input: initial point w_0, learning rate η, preconditioner $\hat{D}_0$, probability p
$z_0 = w_0$, $v_0 = \nabla P(w_0)$
for $t = 0, 1, 2, \ldots$ **do**
  $w_{t+1} = w_t - \eta \hat{D}_t^{-1} v_t$
  $z_{t+1} = \begin{cases} z_t, & \text{with probability } p \\ w_t, & \text{with probability } 1-p \end{cases}$
  Generate independently batches i_{t+1} for v_{t+1} and $\mathcal{J}_t$ for $\hat{D}_{t+1}$
  $v_{t+1} = \nabla f_{i_{t+1}}(w_{t+1}) - \nabla f_{i_{t+1}}(z_{t+1}) + \nabla P(z_{t+1})$
  Update the preconditioner $\hat{D}_{t+1}$
end for
Output: $\hat{w}_T$ chosen uniformly from $\{w_t\}_{t=0}^T$

`Scaled L-SVRG` can be described, in words, as follows. The algorithm is initialized with an initial point w_0, a learning rate η, an initial preconditioner $\hat{D}_0$, and probability p. At each iteration $t \geq 0$ of `Scaled L-SVRG` (Algorithm 2) a search direction v_t is generated. This is made up of the full

gradient plus a small adjustment. Next, the new point w_{t+1} is taken to be a step from w_t in the *scaled* direction $\hat{D}_t^{-1} v_t$, of size η. The new point z_{t+1} is either the (unchanged) previous point z_t with probability p, or the scaled approximate gradient step w_{t+1} with probability $1-p$. Finally, the preconditioner is updated and the next iterate begins. The output, denoted by $\hat{w}_T$ is chosen uniformly from the points w_t, for $t = 0, \ldots, T$, generated by `Scaled L-SVRG` (Algorithm 2).

Note that a key difference between `L-SVRG` (Hofmann et al., 2015; Qian et al., 2021) and our new `Scaled L-SVRG` is the inclusion of the preconditioner in Step 4; recall that a competitive preconditioner is described in Section 2.

The following theorem presents a complexity bound on the number of iterations required by `Scaled L-SVRG` to obtain an ε-optimal solution for the non-convex problem (1).

Theorem 4.1. *Suppose that Assumption 1.1 holds, let $\epsilon > 0$, let p denote the probability and let the step-size satisfy $\eta \leq \min\left\{\frac{\alpha}{4L}, \frac{\sqrt{p}\alpha}{\sqrt{24}L}, \frac{p^{2/3}}{144^{2/3}}\frac{\alpha}{L}\right\}$. Given an initial point $w_0 \in \mathbb{R}^d$, let $\Delta_0 = P(w_0) - P^*$. Then the number of iterations performed by `Scaled L-SVRG`, starting from w_0, required to obtain an ε-approximate solution of non-convex finite-sum problem (1) can be bounded by*

$$T = \mathcal{O}\left(\frac{\Gamma}{\alpha}\frac{L\Delta_0}{p^{2/3}\varepsilon^2}\right).$$

While the previous theorem held under a smoothness assumption, here we prove a complexity result for `Scaled SARAH` under both smoothness *and* PL assumptions.

Theorem 4.2. *Suppose that Assumptions 1.1 and 1.2 hold, let $\epsilon > 0$, let p denote the probability and let the step-size satisfy $\eta \leq \min\left\{\frac{p\Gamma}{6\mu}, \frac{1}{4}\frac{\alpha}{L}, \left(\frac{p}{6}\right)^{1/2}\frac{\alpha}{L}, \left(\frac{p}{6}\right)^{2/3}\frac{\alpha}{L}\right\}$. Then the number of iterations performed by `Scaled L-SVRG` sufficient for finding an ε-approximate solution of non-convex finite-sum problem (1) can be bounded by*

$$T = \mathcal{O}\left(\max\left\{\frac{1}{p}, \frac{\Gamma}{\alpha}\frac{L}{p^{2/3}\mu}\right\}\right).$$

5 Numerical Experiments

The purpose of these numerical experiments is to study the practical performance of our new `Scaled SARAH` and `Scaled L-SVRG` algorithms, and hence, to understand the advantages of using the proposed diagonal preconditioner on `SARAH` and `L-SVRG`. These results will also be compared with `SGD`, both with and without the preconditioner described in Section 2, as well as the state-of-the-art (first order) preconditioned optimizer `Adam`.

We test these algorithms on problem (1) with two loss functions: (1) *logistic regression* loss function, which is convex, and (2) *non-linear least squares* loss function, which is nonconvex. The loss functions are described in details below. For further details and experimental results that support the findings of this section, please see Appendix A. Note that all the experiments were initialized at the point $w_0 = 0$, and each experiment was run for 10 different random seeds.

5.1 Loss Functions

Let $P(w)$ be the empirical risk on a dataset $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$. Then, the *logistic regression loss* is

$$P_{\text{logistic}}(w) = \frac{1}{n}\sum_{i=1}^n \log(1 + e^{-y_i x_i^T w}) \tag{8}$$

whereas for $y_i \in \{0, 1\}$ the *non-linear least squares loss (NLLSQ)* is

$$P_{\text{nllsq}}(w) = \frac{1}{n}\sum_{i=1}^n (y_i - 1/(1 + e^{-x_i^T w}))^2 \tag{9}$$

We consider two different loss functions to test our algorithms on both convex and nonconvex settings.

5.2 Binary Classification on LibSVM Datasets

We train the optimizers on three binary classification LibSVM datasets[3], namely `w8a`, `rcv1`, and `real-sim`. We also consider feature-scaled versions of these datasets, where the scaling is done as

[3] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

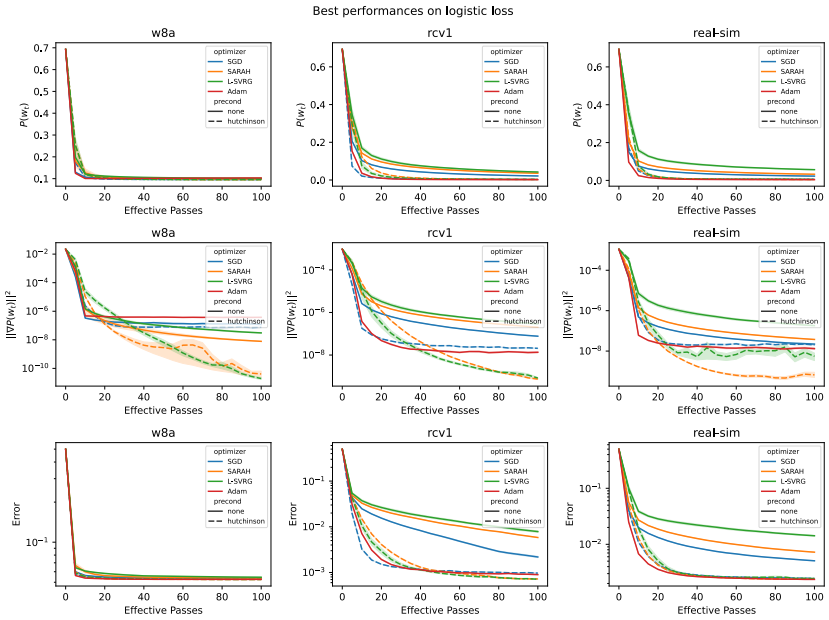

Figure 1: Best performances of the optimizers, including `Adam`, on the (unscaled) LibSVM datasets using the logistic loss. The `Scaled` variants are shown as dashed lines sharing the same color.

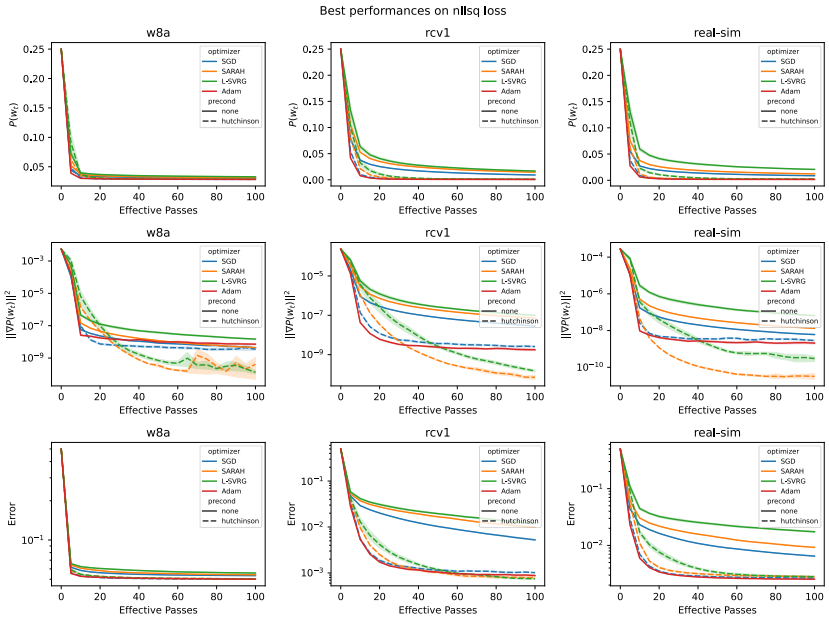

Figure 2: Best performances on the unscaled LibSVM datasets using the NLLSQ loss.

follows: we choose a minimum exponent k_{min} and a maximum exponent k_{max}, and scale the features by values ranging from $10^{k_{min}}$ to $10^{k_{max}}$ in equal steps in the exponent according to the number of features and in random order. The setting $(k_{min}, k_{max}) = (0, 0)$ corresponds to the original, unscaled version of the datasets. We consider combinations of $k_{min} = 0, -3$ and $k_{max} = 0, 3$. This scaling is done to check the robustness and overall effectiveness of the diagonal preconditioner in comparison with `Adam`. Figure 1 shows the results of the first experiment, and presents three types of line plots for each of the datasets of interest where the loss function is the logistic regression loss (8). Figure 2 shows the same for the NLLSQ loss (9). The first row corresponds to the loss, the second is the squared norm of the gradient, and the third is the error. Tuning was performed in

order to select the best hyperparameters (that minimize either the loss, gradient norm squared, or the error). The hyperparameter search grid is reported in Appendix, Table 3, and a thorough discussion is below. We fixed the batch size to be 128, in order to narrow the fine-tuned variables down to η, β, and α. Figure 1 shows the performances when minimizing the error on the unscaled datasets, $(k_{min}, k_{max}) = (0, 0)$. Experiments on scaled datasets can be found in Appendix A.2.

Consider the first column in Figure 1, which uses the `w8a` dataset. While preconditioned `SGD` performs better than `SGD`, for `SARAH` and `L-SVRG` this is not the case. Notice that `Adam` performs well initially, but then further effective data passes lead to small improvement (for all three metrics). While `SARAH` and `L-SVRG` perform the best on this dataset, after approximately 35 passes for `Scaled L-SVRG` and 60 passes for `Scaled SARAH`, performance is better than `Adam`. For the remaining three datasets, preconditioning helps in all cases, with `Scaled L-SVRG` and `Scaled SARAH`, performing the best.

In order to understand the main factors that affect the preconditioner, we ran comparative studies, including studying the parameters β, and α, as well as studying the initialization of preconditioner D_0, including a warm-up period, and studying how the number of samples, z, impacts performance. First, recalling (6), there did not appear to be any significant improvement from averaging across more samples of z per minibatch, neither in the initialization nor in the update step. We also observed that initializing D_0 with a batch size of 100 was sufficiently good for non-sparse problems, and consistently resulted in a relative error of within 0.1 from the true diagonal. However, increasing the number of warm-up samples, proportionally to the number of features, led to observable improvements in convergence for sparse datasets.

We also investigated the role that β (3) played in algorithm performance. We found that larger values lead to slightly slower but more stable convergence. The best β highly depends on the dataset, but the value 0.999 appeared to be a good starting point in general. To ensure a fair comparison, we also optimized `Adam`'s momentum parameter β_2 over the same range.

Aside from the batch size and learning rate, we found that, for ill-conditioned problems, the choice of α (recall (5) and Remark 2.1) played an important role in determining the quality of the solution, convergence speed, and stability (which is not obvious from Figure 1). For example, if the features were scaled with $k_{min} = -3$ and $k_{max} = 0$, the best α is often around 10^{-7} (very small), whereas if we scaled with $k_{min} = 0$ and $k_{max} = 3$, the best α becomes 10^{-1} (relatively large). Therefore, finding the best α might require some additional fine-tuning, depending on the choice of η and β. However, we noticed that once we had tuned the learning rate for one scaled version of the dataset, the same learning rate transferred well to all the other scaled versions. In general, the optimal learning rate in our `Scaled` algorithms is very robust to feature scaling, given that α is chosen well, whereas `Adam`'s learning rate depends more heavily upon how ill-conditioned the problem is, so it requires fine-tuning across a potentially much wider range. In our case, tuning α and β is straightforward, so we obtained state-of-the-art performance with minimal parameter tuning.

The choice $\beta = 1 - 1/(t+1)$ was investigated in Appendices A.5 and E.3. Preliminary results suggest that this choice virtually removes the dependence on α, and is competitive with a fine-tuned β across a large number of values. This makes fine-tuning much easier, even for strong feature scaling.

6 Conclusion and Future Directions

In this paper we presented two novel algorithms for finite sum minimization problems: `Scaled SARAH` and `Scaled L-SVRG`. The algorithms combined existing stochastic reduced variance gradient based methods with a preconditioner, to give algorithms that are better able to adapt to local curvature. We presented theoretical iteration complexity guarantees to provide a bound on the number of iterations required to obtain an ε-optimal solution to (1), and we showed that the algorithms converge linearly when the PL-condition (in addition to smoothness) is assumed. Numerical experiments confirm the benefits of this approach.

Future work. The `Scaled SARAH` and `Scaled L-SVRG` algorithms are deliberately general. This allows for user flexibility in the specific implementation strategy. The analysis presented in this paper considered a specific instance of each algorithm, where a full batch is taken $b = n$, the probability is fixed $p_t = p$ and the preconditioner described in Section 2 is used. Future research directions include the development of theoretical guarantees under different parameter settings.

REFERENCES

C. Bekas, E. Kokiopoulou, and Y. Saad. An estimator for the diagonal of a matrix. *Applied Numerical Mathematics*, 57(11):1214–1229, 2007. Numerical Algorithms, Parallelism and Applications (2).

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proceedings of the 27th International Conference on Neural Information and Processing Systems*, vol. 1, pp. 1646–1654, 2014.

Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of adam and adagrad. *arXiv preprint arXiv:2003.02395*, 2020.

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

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7):2121–2159, 2011.

Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, 2 edition, 1987.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155:267–305, 2016.

Thomas Hofmann, Aurelien Lucchi, Simon Lacoste-Julien, and Brian McWilliams. Variance reduced stochastic gradient descent with neighbors. In *Proceedings of the 28th International Conference on Neural Information and Processing Systems*, pp. 2305–2313, 2015.

Majid Jahani, Mohammadreza Nazari, Sergey Rusakov, Albert Berahas, and Martin Takáč. Scaling up quasi-Newton algorithms: Communication efficient distributed SR1. In *International Conference on Machine Learning, Optimization and Data Science*, pp. 41–54, 2020. PMLR.

Majid Jahani, Mohammadreza Nazari, Rachael Tappenden, Albert Berahas, and Martin Takáč. SONIA: a symmetric blockwise truncated optimization algorithm. In *International Conference on Artificial Intelligence and Statistics*, pp. 487–495, 2021a. PMLR.

Majid Jahani, Sergey Rusakov, Zheng Shi, Peter Richtárik, Michael Mahoney, and Martin Takáč. Doubly adaptive scaled algorithm for machine learning using second-order information. *arXiv preprint arXiv:2109.05198v1*, 2021b.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proceedings of the 26th International Conference on Neural Information and Processing Systems*, vol. 1, pp. 315–323, 2013.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, pp. 2305–2313, 2015.

Jakub Konečný and Peter Richtárik. Semi-stochastic gradient descent methods. *Frontiers in Applied Mathematics and Statistics*, 3, 2017.

Nicolas Le Roux Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Proceedings of the 25th International Conference on Neural Information and Processing Systems*, vol. 2, pp. 2663–2671, 2012.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

L. Lei, C. Ju, J. Chen, and M.I. Jordan. Non-convex finite-sum optimization via SCSG methods. In *Proceedings of the 31st International Conference on Neural Information and Processing Systems*, pp. 2345–2355, 2017.

Zhize Li, Hongyan Bao, Xiangliang Zhang, and Peter Richtarik. PAGE: A simple and optimal probabilistic gradient estimator for nonconvex optimization. In *Proceedings of the 38th International Conference on Machine Learning*, PMLR 139, pp. 6286–6295, 2021a.

Zhize Li, Slavomír Hanzely, and Peter Richtárik. ZeroSARAH: Efficient nonconvex finite-sum optimization with zero full gradient computation. *arXiv preprint arXiv:2103.01447*, 2021b.

Lam M. Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning*, PMLR 70, pp. 2613–2621, 2017.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research, 2 edition, 2006.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Xun Qian, Zheng Qu, and Peter Richtárik. L-SVRG and L-katyusha with arbitrary sampling. *Journal of Machine Learning Research*, 22:1–49, 2021.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

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

Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Zhewei Yao, Amir Gholami, Sheng Shen, Kurt Keutzer, and Michael W Mahoney. ADAHESSIAN: An adaptive second order optimizer for machine learning. *arXiv preprint arXiv:2006.00719*, 2020.

Fangyu Zou, Li Shen, Zequn Jie, Ju Sun, and Wei Liu. Weighted adagrad with unified momentum. *arXiv preprint arXiv:1808.03408*, 2018.