Stochastic Ratios Tracking Algorithm for Large Scale Machine Learning Problems

Shigeng Sun
Department of Engineering Sciences and Applied Mathematics
Northwestern University
Evanston, IL, USA
shigengsun2024@u.northwestern.edu

Yuchen Xie
ycxie@u.northwestern.edu

Abstract

Many machine learning applications and tasks rely on the stochastic gradient descent (SGD) algorithm and its variants. Effective step length selection is crucial for the success of these algorithms, which has motivated the development of algorithms such as ADAM or AdaGrad. In this paper, we propose a novel algorithm for adaptive step length selection in the classical SGD framework, which can be readily adapted to other stochastic algorithms. Our proposed algorithm is inspired by traditional nonlinear optimization techniques and is supported by analytical findings. We show that under reasonable conditions, the algorithm produces step lengths in line with well-established theoretical requirements, and generates iterates that converge to a stationary neighborhood of a solution in expectation. We test the proposed algorithm on logistic regressions and deep neural networks and demonstrate that the algorithm can generate step lengths comparable to the best step length obtained from manual tuning.

1 Introduction

Optimization problems that involve millions of unknown parameters and vast datasets are common occurrences in machine learning. Addressing the computational demands of such problems necessitates highly efficient implementations of stochastic gradient methods [4, 5]. In this context, the paper proposes an algorithm that adaptively tunes the step length (or learning rate), striving to account for the presence of nonlinearity in the true objective function and stochasticity in the function and gradient approximations used in the iteration.

This algorithm is designed to minimize empirical risk,

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} f\left(w;\xi_i\right) := \frac{1}{n}\sum_{i=1}^{n} f_i(w), \tag{1}$$

where $(\xi_i)_{i=1}^n$ denote the training examples and $f(\cdot;\xi) : \mathbb{R}^d \to \mathbb{R}$ is the composition of a prediction function (parametrized by w) and a loss function. $\{\xi_k\}$ can be seen as representing a sequence of jointly independent random variables. The training problem consists of finding an optimal choice of the parameters $w \in \mathbb{R}^d$ with respect to F, i.e.,

$$\min_{w\in\mathbb{R}^d} F(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) \tag{2}$$

Preprint. Under review.

For these type of problems, the current *de facto* optimization methods are the stochastic gradient descent (SGD) method and its variants, such as Adam and AdaGrad. More concretely, a basic mini-batch stochastic gradient algorithm at iteration k involves the update

$$w_{k+1} = w_k - \alpha_k \nabla F_{S_k} \tag{3}$$

where we define

$$\nabla F_{S_k}(w_k) = \frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(w_k) \tag{4}$$

and the set $S_k \subset \{1, 2, \ldots\}$ indexes training data points. The sample S_k changes at every iteration and in the basic mini-batch stochastic algorithm, its cardinality $|S_k|$ remains the same. The step size parameter of SGD algorithm is is often difficult to determine and a large number of approaches have been proposed to address it, and none has been universally adopted.

The recent paper by Defazio et al. [10] proposes a steplength technique based on worst-case complexity. It requires the estimation of the distance D from the initial point to the solution set as well as the estimation of the Lipschitz constant G. In this paper, we take the view that the step length parameter should depend on G but also on the variance in the stochastic gradient approximations (as opposed to the distance to the solution).

Our Goals Whilst we do not purport that our algorithm is designed with the primary objective of surpassing some of the existing state-of-the-art computational algorithms [16, 11, 1, 9] in terms of efficiency or speed, our intention is to explicate the principles that govern the selection of step lengths for stochastic algorithms in the most interpretable and transparent fashion. We have also endeavored to demonstrate the intricacies of such an algorithm that incorporates these principles. Furthermore, we contend that by allowing the discretion to select search directions d_k (as expounded in the algorithmic section), this framework can be adapted to various algorithms, e.g. [12, 14], not just the standard SGD algorithm.

Stochastic Line Search or Stochastic Trust Region Methods In classical optimization, techniques like trust region and line search are used to control step lengths since near the current iterate, the step direction generated is usually a descent direction[17]. However, in stochastic optimization, the direction's quality can be questionable, and the value of exploring it is debated. Despite this, instead of relying on external inputs for step length selection, several algorithms use classical nonlinear optimization methods to dynamically determine step lengths during runtime. For instance, [6, 18, 21] have each examined stochastic line search algorithms that potentially involve multiple function evaluations on a given direction; [15] introduced a stochastic step search algorithm that modifies the search direction each time a step is refused under a relaxed Armijo condition; in terms of trust region approaches, [3, 7, 22] have each proposed remedies for adapting existing algorithms to stochasticities.

In stead of treating step lengths as exogenous quantities that may require substantial tuning or using a pre-set diminishing step length as proposed in [20], we propose to employ a progress ratio to gauge the average progress made in past iterations and differentiate the effect of noise into a separate ratio. This obviates the potential additional computation for the algorithm to backtrack multiple times before taking a step and allows for a more comprehensive assessment of the overall progress made in the previous iterations.

Contributions We propose a novel algorithm for determining the appropriate step length at the runtime, thus avoiding extensive tuning. The numerical experiments show that the method proposed in this paper—which we call the Stochastic Ratios Tracking method—produces step lengths that are comparable to the best hand-tuned step lengths while achieving a good balance between computational costs and training efficiency. Analytical results show convergence in expectation of the algorithm under mild conditions while highlighting the effect of nonlinearity and stochasticity during the design of step length selection principles.

2 The Stochastic Ratio Tracking Algorithm

When tackling stochastic optimization problems in machine learning, selecting an appropriate step length parameter for gradient-based first-order methods depends on two key factors: nonlinearity

and stochasticity. Nonlinearity, which is sometimes represented as curvature information, determines the optimal choice of step length in a given direction in the conventional non-stochastic settings. Stochasticity, on the other hand, relates to inaccuracies or noise in gradient estimates, and is often characterized by variances or second moments. To achieve optimal performance, one must take into consideration both non-linearity and stochasticity.

To motivate the proposed procedure, we cite a well-known result about the stochastic gradient method (c.f. [5], Theorem 4.6). It has been established that if the step length is given by

$$\alpha = \frac{1}{L(M_V + 1)}, \tag{5}$$

(where L is the Lipschitz constant, and M_V is a scalar defined below) then the iterates given by the stochastic gradient algorithm converge to a stationary point in expectation under certain conditions. Specifically, the iterations update $w_{k+1} = w_k - \alpha \nabla F_{S_k}$ yields

$$\mathbb{E}\left[F\left(w_k\right) - F_*\right] \leq \left(1 - \frac{1}{L(M_V + 1)}\right)^{k-1} \left(F\left(w_1\right) - F_*\right), \tag{6}$$

where F^* is the optimal function value. A result of this type can be established under various conditions (c.f. [5], Assumptions 4.1 to 4.3); one of the conditions is

$$\mathbb{E}_{\xi_k}[\|\nabla f\left(w_k, \xi_k\right)\|^2] - \|\nabla F(w_k)\|^2 \leq M_V \|\nabla F(w_k)\|^2, \tag{7}$$

here $F(w_k)$ denotes the true objective. We can approximate this condition at each given iterate k as:

$$\frac{\sum_{i \in S_k} \|\nabla f_i\left(w_k\right)\|^2 / |S_k| - \|\nabla F_{S_k}(w_k)\|^2}{\|\nabla F_{S_k}(w_k)\|^2} \leq M_V. \tag{8}$$

We can compute the value on the left-hand side using Backpack [8] within an iteration, which provides an approximation of M_V, that we shall name the *variance ratio estimate* later in this paper.

While many problems in machine learning satisfy (7) [1], and in particular in some over-parametrized models in data science [2], this assumption still can be perceived as somewhat strong. A more general condition is

$$\mathbb{E}_{\xi_k}[\|\nabla f\left(w_k, \xi_k\right)\|^2] - \|\nabla F(w_k)\|^2 \leq M_V \|\nabla F(w_k)\|^2 + M.$$

We will discuss later that the presence of a constant term $M \geq 0$ on the right-hand side would only potentially introduce an overestimation of M_V in our algorithm, which doesn't affect convergence, as the size of the overestimation is bounded above outside of a neighborhood of stationarity.

Besides using the noise ratio M_V to account for stochasticity, we shall employ another ratio estimate (*progress ratio estimate*) to allow the step to be adjusted to scale with $\frac{1}{L}$ as suggested in (5). In the subsequent section, we shall explain the computations of the two ratios above.

2.1 Step Length Selection

Progress Ratio Estimates In this work, we abstain from employing back-tracking line search or shrinking trust region within an iteration; instead, we opt to calculate a "Progress Ratio Estimate" to evaluate the effectiveness of the current step length along the intended direction. After sufficient information about past iterations is attained, we update the step length accordingly.

More rigorously, within an iteration k, we generate an index set S_k and fix it. We compute the stochastic gradient $\nabla F_{S_k}(w_k)$ and the search direction d_k (for instance, SGD has $d_k = -\nabla F_{S_k}(w_k)$; this framework also allows other choices of d_k). We then evaluate the *Progress Ratio Estimate* with step length α as $\hat{\rho}_k(\alpha)$:

$$\hat{\rho}_k(\alpha) = \frac{F_{S_k}(w_k + \alpha d_k) - F_{S_k}(w_k)}{\alpha \nabla F_{S_k}^T(w_k) d_k}. \tag{9}$$

Here the same sample S_k is used in (9). Note that the condition $\rho_k(\alpha) \geq c_1$ is the Armijo condition on F_{S_k} stating that the step length is to have provided a sufficient decrease in the objective. We

[1] This condition is sometimes referred to as 'homogeneity of minima': a minimizer of F is a minimizer of f with probability one [19].

track ratio (9) over a set of iterations and if the average value of $\hat{\rho}$ in these iterates is large enough, we increase the step length, as this suggests that more progress may be achievable. On the other hand, if that ratio tracks to a small value, we shrink α to facilitate the alignment between progress and expectation. In other words, this non-dimensional ratio $\hat{\rho}$ is employed to assess the degree of alignment between the stochastic objective function and its local linear model. A value of the ratio in proximity to 1 indicates the stochastic objective function exhibits linear behavior along the direction d_k and the progress achieved by traversing a distance of α in this direction is in accordance with the anticipated progress.

Our progress ratio tracking, therefore, determines whether the steplengths should be adjusted. We now consider how to choose the actual value of α_k.

Variance Ratio Estimates As already mentioned in the paragraph surrounding (5), the step length must depend on the error/noise in the gradient approximation, it's natural to measure this noise in the form of the mini-batch variance. Thanks to the recent advent of Backpack [8], obtaining estimates of $\hat{M}_V$ from mini-batch updates has become relatively effortless in PyTorch, with only minor computational overhead. We perform this as follows:

We re-iterate that the stochastic objective F_{S_k} is often referred to as a 'mini-batch' with cardinality $|S_k| = m$ (where $m = 16, 32, 64, ...$ are commonly used):

$$F_{S_k}(w_k) = \frac{1}{|S_k|} \sum_{i \in S_k} f_i(w_k), \quad \nabla F_{S_k}(w_k) = \frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(w_k) \tag{10}$$

This motivates an estimated empirical value $\hat{M}_{V,k}$, that would allow the satisfaction of (8) when used in place of M_V:

$$\hat{M}_{V,k} = \frac{\hat{V}_k}{\|\nabla F_{S_k}(w_k)\|^2}, \quad \text{with} \quad \hat{V}_k = \sum_{i \in S_k} \|\nabla f_i(w_k)\|^2/|S_k| - \|\nabla F_{S_k}(w_k)\|^2 \tag{11}$$

This dimensionless ratio $\hat{M}_{V,k}$ is used to quantify the magnitude of noise inherent in the estimates of stochastic gradients. As explained before, it serves as an indicator of the relative second moment of the stochastic gradients, a quantity that an ideal step length should inversely dependent upon.

We comment again on the condition in (7). A weaker assumption can be written as $\mathbb{E}_{\xi_k}[\|\nabla f(w_k, \xi_k)\|^2] - \|\nabla F(w_k)\|^2 \leq M_V \|\nabla F(w_k)\|^2 + M$. If this weaker assumption is satisfied (with $M > 0$) instead of (7), we might overestimate M_V, but the overestimation will be no more than $M/\|\nabla F_k(w_k)\|$, which is bounded above by M/ϵ if $\|\nabla F_k(w_k)\| > \epsilon$, i.e., if the iterates are outside the neighborhood of a stationary point. Hence, replacing (7) with a weaker assumption doesn't affect the convergence of the algorithm to the stationary neighborhood.

In the next section, we shall proceed to expound upon the algorithm. It is pertinent to reiterate that the incorporation of the *progress ratio estimates* $\hat{\rho}_k$ is intended to factor in nonlinearity and enable the scaling of steps with $1/L$, while the *variance ratio estimates* $\hat{M}_{V,k}$ are intended to account for the impact of noise or inaccuracies in the gradient estimates. By isolating and addressing the effects of nonlinearity and noise separately, the algorithm can leverage mechanisms designed for classical, non-stochastic non-linear optimization in the stochastic setting.

2.2 Specification of the Algorithm

In the k-th iteration, upon generating $\nabla F_{S_k}(w_k)$ and d_k, our proposed algorithm calculates the quantities $\hat{\rho}_k(\alpha)$ for a trial step size α and $\hat{M}_{V,k}$ using the expressions provided in the previous subsection. These values, $\hat{\rho}_k(\alpha)$ and $\hat{M}_{V,k}$, are then appended in buffers v_ρ and v_M, respectively. (If the size of v_ρ and v_M exceed a predetermined memory length N, we delete the oldest values from the buffer.) The algorithm then takes the step:

$$w_{k+1} = w_k + \frac{\alpha_k}{mean(v_M) + 1} d_k \tag{12}$$

Once the size of v_ρ reaches the pre-specified memory length N, the algorithm decides whether to increase or decrease α_k based on the values in v_ρ. Our strategy is to compute the mean of the values

in v_ρ (denoted as $\bar{\rho}$) and compare it with user-defined parameters c_1 and c_2, where $0 < c_1 \leq c_2 < 1$. If $\bar{\rho} < c_1$, then α is reduced, and if $\bar{\rho} > c_2$, then α is increased.

We are now ready to state our algorithm below:

Algorithm 1: Stochastic Ratio Tracking Algorithm (SRT)

Initialize w_0 and α_0, pick $0 < c_1 \leq c_2 < 1$, $\tau > 1$ and memory buffers v_ρ , v_M and length N.
for $k = 0, 1, ...$ **do**
  Generate index set S_k, evaluate ∇F_{S_k} by (10), compute d_k;
  Evaluate $\hat{M}_{V,k}$ by (11) and $\hat{\rho}_k(\alpha_k)$ by (9);
  Append $\hat{\rho}_k(\alpha_k)$ to v_ρ and $\hat{M}_{V,k}$ to and v_M; if size of a buffer exceeds N, delete oldest element;
  if size of $v_\rho = N$ **then**
    if $mean(v_\rho) > c_2$ **then**
      $\alpha_{k+1} = \tau \alpha_k$
      clear v_ρ
    else if $mean(v_\rho) < c_1$ **then**
      $\alpha_{k+1} = \alpha_k / \tau$
      clear v_ρ
    end
  end
  Update iterate by (12);
end

To illustrate the implementation of this algorithm, we provide the following schematics in Figure 1.

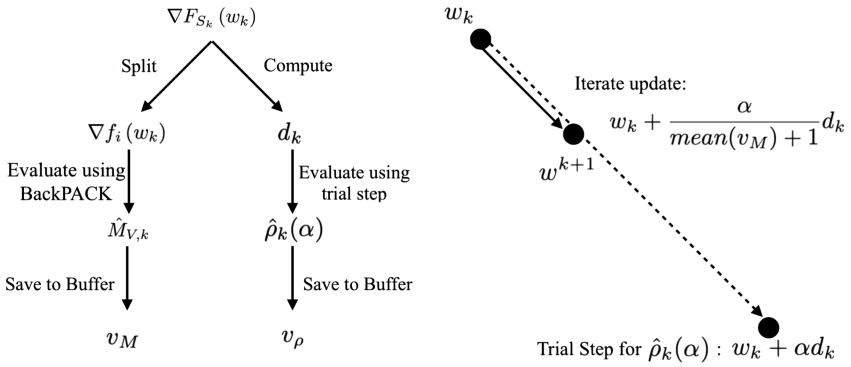

Figure 1: Schematics of the SRT algorithm. Left: computation of estimates. Right: Trial step and iteration updates.

3 Fixed Step Lengths and Batch Sizes Experiments

We first showcase the ratios $\hat{\rho}_k, \hat{M}_{V,k}$ in controlled settings before presenting convergence theory. More specifically, for Logistic regression and DNN problems, we employ the stochastic gradient algorithm with different fixed step lengths and across different batch sizes. We show that the choice of step length α has a direct impact on the estimated and $\hat{\rho}_k$, and similarly the choice of batch size $\|S_k\|$ affects the estimated and $\hat{M}_{V,k}$ as expected.

3.1 Batch size's influence on $\hat{M}_{V,k}$

As discussed in the previous sections, the values of $\hat{M}_{V,k}$ quantify the level of stochasticities in the gradient estimations, where a better estimation of the gradient in a larger-sized batch should be reflected in smaller values in the estimations of $\hat{M}_{V,k}$. We demonstrate this effect in this subsection.

We start the initial round of experiments in Figure 2 with logistic regression on the gisette dataset [13], where we run the SGD algorithm with a constant step length of 0.003 for 10 epochs. We repeat the same experiment with batch size $\|S_k\|$ of 8 and 64, and included, in each of the left and right panels, an example of typical results in Figure 2. In each of the panels, from top to bottom, we report $F_{S_k}(w_k)$, step length, $\hat{M}_{V,k}$ and $\hat{\rho}_k$, respectively.

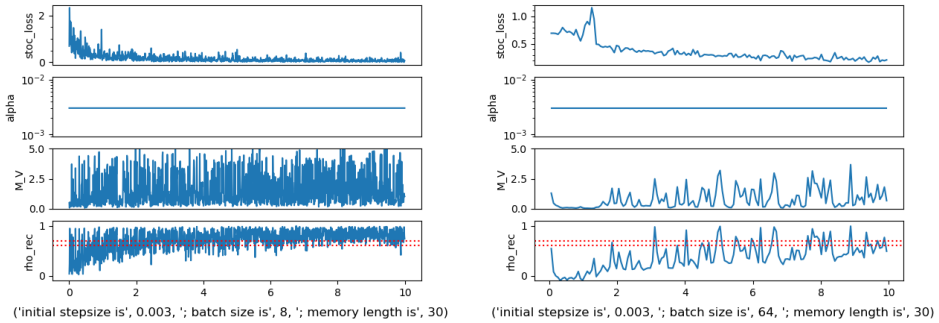

Figure 2: Logistic regression with the gisette data. Each of left and right panels contains a typical run with a constant step length of 0.003; each panel contains from top to bottom: $F_{S_k}(w_k)$, $\hat{M}_{V,k}$, $\mathbb{V}_k$, $\hat{\rho}_k(\alpha)$; the left panel has a minibatch size of 8 while the right panel has a minibatch size of 64.

As we observe from Figure 2, throughout the training progress, the values of $\hat{\rho}_k$ remained largely comparable when batch size increased from 8 to 64; however, $\hat{M}_V$ values reduced, indicating that the noise in the gradient is reduced and larger steps can be favored.

We continue to perform similar experiments on Deep Neural Networks. We train a 3-layer feed-forward neural network on the Fashion-MNIST data, using a constant step length of 0.5. Similar to before, we vary the batch size from 16 to 256 in the left and right panels of 3, respectively. Each panel again contains $F_{S_k}(w_k)$, step length, $\hat{M}_{V,k}$ and $\hat{\rho}_k$, from top to bottom.

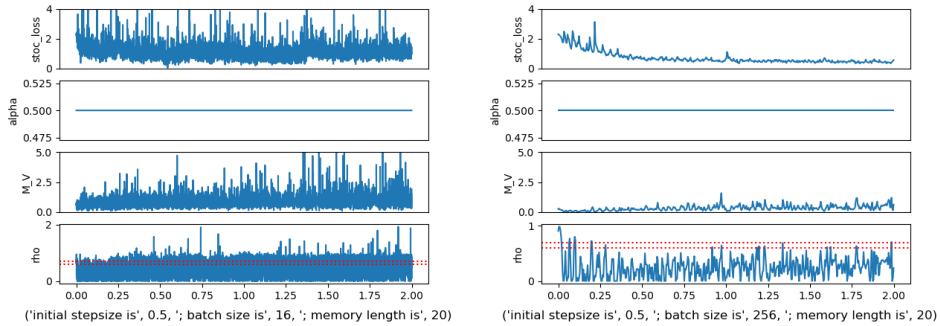

Figure 3: DNN with the Fashion-MNIST data. Each of left and right panels contains a typical run with a constant step length of 0.5; each panel contains from top to bottom: $F_{S_k}(w_k)$, $\hat{M}_{V,k}$, $\mathbb{V}_k$, $\hat{\rho}_k(\alpha)$; the left panel has a minibatch size of 16 while the right panel has minibatch size of 256.

Similar to Figure 2, for deep neural networks results presented in Figure 3, throughout the training progress, when batch size increased from 16 to 256, $\hat{M}_V$ values reduced, as expected.

3.2 Step size's influence on $\hat{\rho}_k$

As discussed in the section around (9), we expect $\hat{\rho}_k$ to be close to 1 when step lengths are picked to be small (as analytical functions are locally approximated by linear ones) and may oscillate wildly when step lengths are large relatively—we demonstrate this effect in this subsection.

In the logistic regression task, we fixed a batch size of 8 and varied the step length from 0.3 to $3e-4$ and examined the resulting values of $\hat{\rho}_k(0.3)$ and $\hat{\rho}_k(3e-4)$ in Figure 4. We observed that the values of $\hat{\rho}_k(0.3)$ were widely dispersed across the interval [0,1], with many values being close to 0, indicating potentially too large of step sizes. In contrast, the values of $\hat{\rho}_k(3e-4)$ were clustered around 1, indicating more progress maybe available for larger step sizes.

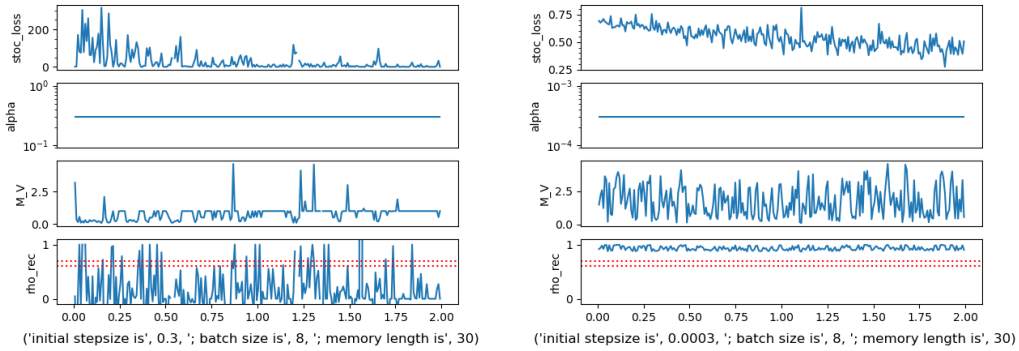

Figure 4: Logistic Regression with Different Step Lengths. Left column: Steplength 3e-1; Right column: Steplength 3e-4.

We conducted similar experiments on DNN for the Fashion-MNIST dataset and varied the step lengths 0.5 to $5e-4$ and examined the resulting values of $\hat{\rho}_k(0.5)$ and $\hat{\rho}_k(5e-4)$ in Figure 5. We observed that, similar to logistic regression, $\hat{\rho}_k(0.5)$ exhibited wild oscillations with values approaching 0, while $\hat{\rho}_k(5e-4)$ clustered around 1. For this application, a step length of 0.5 improved the stochastic objective but caused erratic oscillations, while 5e-4 resulted in slow training progress or "stalling". SRT would decrease the step length in the former case to prevent oscillations and increase it in the latter case to promote progress.

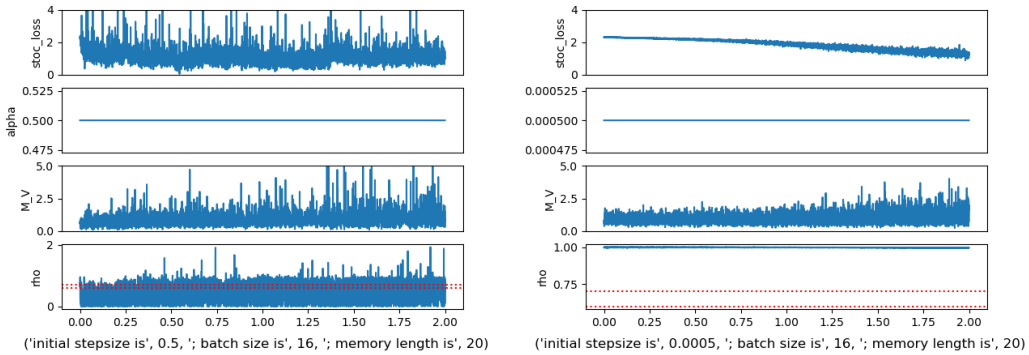

Figure 5: Deep Neural Network with Different Step Lengths. Left column: Step length 5e-1 (erratic oscillation in objective); Right column: Step length 5e-4 (training stalling).

Given that the behaviors of $\hat{\rho}_k$ and $\hat{M}_{V,k}$ generally align with our expectations, we will now proceed to present the convergence results for this algorithm in the next section, followed by the application of the algorithm for training tasks in the subsequent section.

4 Convergence Result for the SRT Algorithm

We present the main convergence result for the algorithm below and defer its proof to the appendix.

Convergence Theorem *Assume that the stochastic objective function $F_{S_k}(w)$ and the true objective function $F(w)$ are μ, L smooth; that $\mathbb{E}_{\xi_k}[\nabla F_{S_k}(w_k)] = \nabla F(w_k)$, and $\mathbb{E}_{\xi_k}[\|\nabla F_{S_k}(w_k)\|^2] - \|\nabla F_k\|^2 \leq M_V \|\nabla F_k\|^2 + M$ and let c_1 and c_2 be picked such that:*

$$1 - \frac{\mu}{2L} < c_1 < c_2 < 1 - \frac{\mu}{2\tau L} \tag{13}$$

and that the estimated $\hat{M}_{V,k} = M_V$, then for all k such that $k > k_0$, where

$$k_0 = N \cdot \max\left(0, \log_\tau \frac{\mu}{\tau L^2 \alpha_0}, \log_\tau \frac{\alpha_0}{L}\right)$$

the expected optimality gap satisfies the following inequality:

$$\mathbb{E}[F(w_k) - F^*] \leq \eta + \left(1 - \frac{2(1-c_1)\mu}{\tau(M_V+1)L}\right)^{k-k_0} [F(w_{k_0}) - F^* - \eta] \tag{14}$$

where

$$\eta = \frac{\tau M}{4\mu(1-c_1)(M_V+1)}. \tag{15}$$

This theorem highlights that under μ, L smoothness assumptions of the stochastic and true objective functions, the algorithm proposed is convergent towards a stationary neighborhood surrounding the true objective of size η. The stationary neighborhood η is proportional to the non-diminishing error term denoted as M in the second movement bound of the gradient. The theorem further shows that the optimal step length is attained at iteration k_0, after which the algorithm additionally manifests a linear convergence rate that depends upon the problem's conditioning.

While the algorithms enjoy a favorable linear convergence rate to the stationary neighborhood as characterized by the optimality gap, the somewhat restrictive assumptions leave room for future research directions. For instance, we have assumed for simplicity that the estimated $\hat{M}_V$ reflects the true value. As a potential future direction, one may assume the distribution of $\|\nabla F_{S_k}\|$ and derive high probability bounds for the estimated values. Another limitation of the analysis is that the choices of c_1, c_2 require information about the conditioning of the problem. While many practical methods exist for estimating this information, we have found in practice that the algorithm is robust with respect to different choices of c_1, c_2, and that this restriction is purely technical to ensure that the step length eventually settles to a fixed value.

5 Numerical Experiments with SRT

To show SRT algorithm's capability of automatic step length tuning, we tested with logistic regression and deep neural network training tasks. All codes were written in Python; experiments on Deep Neural Networks were implemented in Pytorch with BackPACK[8]. Logistic regression experiments are done on a MacBook Pro with Intel i7 processor with 32 GBs of DDR4 RAM. Deep Neural Network experiments were done on a PC with Nvidia GeForce GPU with 11GB of dedicated VRAM.

For the first set of experiments reported in Figure 6, we ran logistic regression on the gisette dataset for 10 epochs with a batch size of 8, and initialized the step length at 0.1 and $1e-4$, in the left and right panel respectively. We recorded the stochastic objective value in the top panels, the adjusted step lengths α in the middle panels, and the computed $\hat{\rho}$ in the bottom panels.

As observed in Figure 6, the SRT algorithm identifies step lengths as too large or small and adjusts them accordingly. In both cases, the step lengths settled slightly above 1e-3, which match our best hand-tuned step length.

For DNN training tasks, the behavior observed in Figure 7 is comparable to what was observed before, where we applied the SRT algorithm to train a 3-layer feed-forward neural network on the Fashion-Mnist dataset for 20 epochs. The SRT algorithm reduces the step length when it is too large

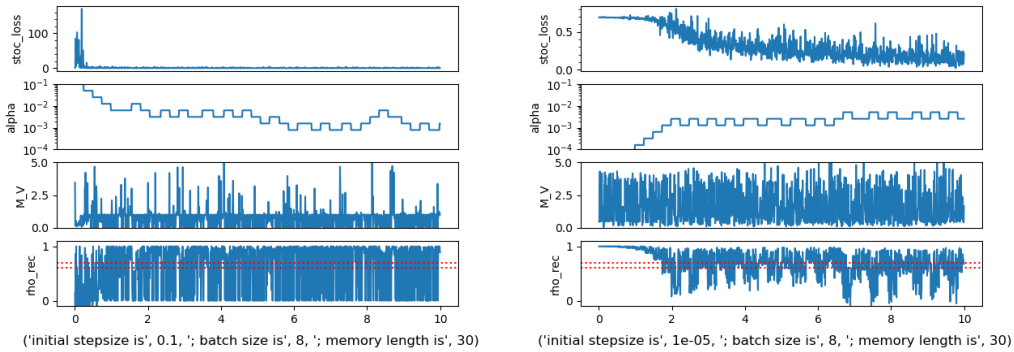

Figure 6: SRT on Logistic Regression with Different Initial Step Lengths. Left column:Initial Steplength 1e-1; Right column: Initial Steplength 1e-5; Each of the panels contains from top to bottom: $F_{S_k}(w_k)$, $\hat{M}_{V,k}$, $\mathbb{V}_k$, $\hat{\rho}_k(\alpha)$.

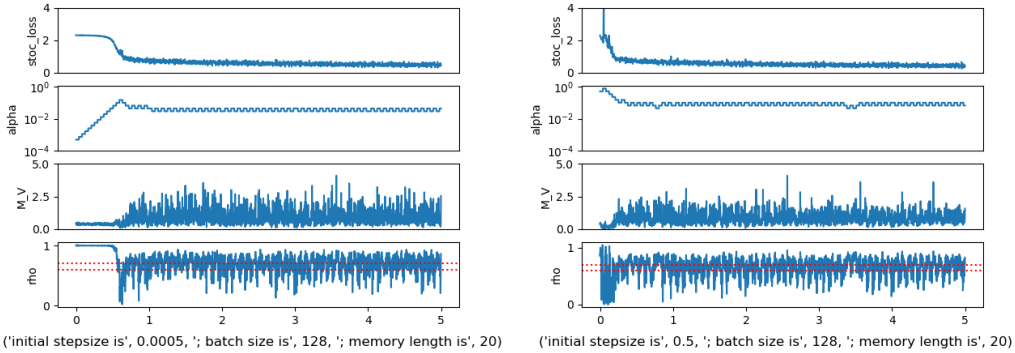

Figure 7: SRT on the training of a feed-forward 3-layer Neural Network on the Fashion-Mnist data with batch size 128. Left column:Initial Steplength 0.5; Right column: Initial Steplength 0.0005. Each of the panels contains from top to bottom: $F_{S_k}(w_k)$, $\hat{M}_{V,k}$, $\mathbb{V}_k$, $\hat{\rho}_k(\alpha)$.

and increases it when it is too small. And in both cases, the final step length settled to around 0.06, again comparable to our best hand-tuned step length.

In both Logistic regression and DNN training tasks, with initially too short or too long of step length choices, SRT algorithm was successful in identifying and adjusting to step lengths which eventually settle into similar values that are comparable to the best hand-tuned values.

6 Conclusions

This paper presents a novel step length selection algorithm, namely SRT, which stands out from traditional methods by obviating the need for manual tuning efforts and enabling automatic step size adjustments. The proposed algorithm is shown to be convergent under mild assumptions, and the numerical results demonstrate its competitive performance across different training tasks. In particular, SRT produces effective step lengths that are comparable to those obtained from manual tunning.

As the proposed estimations of $\hat{\rho}_k$ and $\hat{M}_{V,k}$ offer novel insights for step length tuning and are applicable to various problems and tasks, future work may involve conducting numerical experiments with acceleration directions instead of the steepest descent direction. Additionally, relaxing the assumptions necessary for convergence analysis by incorporating the underlying statistical nature could be explored. We leave these potential avenues for future research.

Acknowledgments and Disclosure of Funding

Sun was supported by NSF grant DMS-1620022.

The authors would like to acknowledge Prof.Jorge Nocedal for his perceptive insights and critical evaluations of this work, which greatly enriched the theory and interpretation presented in this paper.

The authors would also like to thank Prof.Richard Byrd and Prof.Figen Öztoprak for their feedback.

References

[1] Reza Babanezhad, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečnỳ, and Scott Sallinen. Stop wasting my gradients: Practical svrg. Advances in Neural Information Processing Systems (NIPS), 2015.

[2] Raef Bassily, Mikhail Belkin, and Siyuan Ma. On exponential convergence of sgd in non-convex over-parametrized learning. arXiv preprint arXiv:1811.02564, 2018.

[3] Stefania Bellavia, Gianmarco Gurioli, Benedetta Morini, and Philippe Toint. The impact of noise on evaluation complexity: The deterministic trust-region case. arXiv preprint arXiv:2104.02519, 2021.

[4] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[5] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. Siam Review, 60(2):223–311, 2018.

[6] Coralia Cartis and Katya Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. Mathematical Programming, pages 1–39, 2015.

[7] Frank E Curtis, Katya Scheinberg, and R. Shi. A stochastic trust region algorithm based on careful step normalization. INFORMS Journal on Optimization, 1(3):200–220, 2019.

[8] Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. In International Conference on Learning Representations, 2020.

[9] Aaron Defazio, Francis Bach, and Simon 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.

[10] Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation. arXiv preprint arXiv:2301.07733, 2023.

[11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121–2159, 2011.

[12] Sébastien Gadat, Fabien Panloup, and Sofiane Saadane. Stochastic heavy ball. 2018.

[13] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the NIPS 2003 feature selection challenge. In Advances in neural information processing systems, pages 545–552, 2004.

[14] Chonghai Hu, Weike Pan, and James Kwok. Accelerated gradient methods for stochastic optimization and online learning. Advances in Neural Information Processing Systems, 22, 2009.

[15] Billy Jin, Katya Scheinberg, and Miaolan Xie. High probability complexity bounds for line search based on stochastic oracles. Advances in Neural Information Processing Systems, 34, 2021.

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[17] Jorge Nocedal and Stephen Wright. Numerical Optimization. Springer New York, 2 edition, 1999.

[18] Courtney Paquette and Katya Scheinberg. A stochastic line search method with convergence rate analysis. arXiv preprint arXiv:1807.07994, 2018.

[19] Vivak Patel. The impact of local geometry and batch size on stochastic gradient descent for nonconvex problems. arXiv preprint arXiv:1709.04718, 2017.

[20] H. Robbins and S. Monro. A stochastic approximation method. The Annals of Mathematical Statistics, pages 400–407, 1951.

[21] Hao-Jun M Shi, Yuchen Xie, Richard Byrd, and Jorge Nocedal. A noise-tolerant quasi-newton algorithm for unconstrained optimization. SIAM Journal on Optimization, 32(1):29–55, 2022.

[22] Shigeng Sun and Jorge Nocedal. A trust region method for the optimization of noisy functions. arXiv preprint arXiv:2201.00973, 2022.

Appendix

In this section we present the proof for the main global convergence theorem:

Convergence Theorem *Assume that the stochastic objective function $F_{S_k}(w)$ and the true objective function $F(w)$ are μ, L smooth; that $\mathbb{E}_{\xi_k}[\nabla F_{S_k}(w_k)] = \nabla F(w_k)$, and $\mathbb{E}_{\xi_k}[\|\nabla F_{S_k}(w_k)\|^2] - \|\nabla F_k\|^2 \leq M_V \|\nabla F_k\|^2 + M$ and let c_1 and c_2 be picked such that:*

$$1 - \frac{\mu}{2L} < c_1 < c_2 < 1 - \frac{\mu}{2\tau L} \tag{16}$$

and that the estimated $\hat{M}_{V,k} = M_V$, then for all k such that $k > k_0$, where

$$k_0 = N * \max\left(0, \log_\tau \frac{\mu}{\tau L^2 \alpha_0}, \log_\tau \frac{\alpha_0}{L}\right)$$

the expected optimality gap satisfies the following inequality:

$$\mathbb{E}[F(w_k) - F^*] \leq \eta + \left(1 - \frac{2(1-c_1)\mu}{\tau(M_V+1)L}\right)^{k-k_0} [F(w_{k_0}) - F^* - \eta] \tag{17}$$

where

$$\eta = \frac{\tau M}{4\mu(1-c_1)(M_V+1)}. \tag{18}$$

Proof: For any particular iteration k, by μ, L smoothness assumption of F_{S_k},

$$\alpha \nabla F_{S_k}^T d_k + \frac{\alpha^2 \mu}{2} \|d_k\|_2^2 \leq F_{S_k}(w_k + \alpha d_k) - F_{S_k}(w_k) \leq \alpha \nabla F_{S_k}^T d_k + \frac{\alpha^2 L}{2} \|d_k\|_2^2 \tag{19}$$

using the fact that $d_k = \nabla F_{S_k}$,

$$-\alpha \|\nabla F_{S_k}\|^2 + \frac{\alpha^2 \mu}{2} \|\nabla F_{S_k}\|_2^2 \leq F_{S_k}(w_k + \alpha d_k) - F_{S_k}(w_k) \leq -\alpha \|\nabla F_{S_k}\|^2 + \frac{\alpha^2 L}{2} \|\nabla F_{S_k}\|_2^2 \tag{20}$$

dividing this inequality by $-\alpha \|\nabla F_{S_k}\|^2$ and combining terms:

$$1 - \frac{1}{2}\mu\alpha \geq \frac{F_{S_k}(w_k + \alpha d_k) - F_{S_k}(w_k)}{-\alpha \|g_k\|^2} \geq 1 - \frac{1}{2} L\alpha \tag{21}$$

using definition of $\hat{\rho}_k(\alpha)$ in (9) this implies:

$$\hat{\rho}_k(\alpha) \in \left[1 - \frac{L\alpha}{2}, 1 - \frac{\mu\alpha}{2}\right] \tag{22}$$

If $\alpha > 1/L$ at iteration k this implies that

$$\hat{\rho}_k(\alpha) \leq 1 - \frac{\mu\alpha}{2} < 1 - \frac{\mu}{2L} < c_1 \tag{23}$$

Since the same argument for k will apply for $k+1, ..., k+N-1$, α will be decreased based on the algorithm within N iterations.

Conversely, if $\alpha < \frac{1}{\tau L} \frac{\mu}{L}$ at iteration k we have

$$\hat{\rho}_k(\alpha) \geq 1 - \frac{L\alpha}{2} > 1 - \frac{\mu}{2\tau L} > c_2 \tag{24}$$

Since the same argument for k will apply for $k+1, ..., k+N-1$, α will be increased based on the algorithm within N iterations.

Thus after finitely many iterations,

$$\alpha_k \in \left(\frac{1}{\tau L}\frac{\mu}{L}, \frac{1}{L}\right) \tag{25}$$

Also note: $\frac{\mu}{L} > 2(1-c_1)$ by the choice of c_1,

$$\alpha_k \in \left(\frac{1}{\tau L}\frac{\mu}{L}, \frac{1}{L}\right) \subset \left(\frac{2(1-c_1)}{\tau L}, \frac{1}{L}\right). \tag{26}$$

Since

$$\frac{1}{L} \Big/ \left(\frac{1}{\tau L}\ \frac{\mu}{L}\right) > \tau, \tag{27}$$

α_k must eventually settle to a value in this interval. Furthermore, the number of iterations required to produce a step in the above interval is

$$k_0 = N * \max\left(0, \log_\tau \frac{\mu}{\tau L^2 \alpha_0}, \log_\tau \frac{\alpha_0}{L}\right) \tag{28}$$

Now consider the case when $k > k_0$ i.e. a final step size $\bar{\alpha}$ is achieved such that

$$\bar{\alpha} \in \left(\frac{2(1-c_1)}{\tau L}, \frac{1}{L}\right) \tag{29}$$

From this point on, we assume that the optimal step size $\bar{\alpha}$ is achieve, by the assumption that $\hat{M}_V = M_V$, the steps taken will admit:

$$w_{k+1} = w_k - \frac{\bar{\alpha}}{M_V + 1} \nabla F_{S_k}(w_k) \tag{30}$$

By the $L-$Lipschitz continuity of F, we have

$$F\left(w_{k+1}\right) - F\left(w_k\right) \leq \nabla F\left(w_k\right)^T \left(w_{k+1} - w_k\right) + \frac{1}{2} L \left\|w_{k+1} - w_k\right\|_2^2 \tag{31}$$

$$\leq -\frac{\bar{\alpha}}{M_V + 1} \nabla F\left(w_k\right)^T \nabla F_{S_k}\left(w_k\right) + \frac{1}{2}\left(\frac{\bar{\alpha}}{M_V + 1}\right)^2 L \left\|\nabla F_{S_k}\left(w_k\right)\right\|_2^2 \tag{32}$$

Taking expectation with respect to ξ_k, we obtain

$$F\left(w_{k+1}\right) - F\left(w_k\right) \leq -\frac{\bar{\alpha}}{M_V + 1} \|\nabla F\left(w_k\right)\|^2 + \frac{1}{2}\left(\frac{\bar{\alpha}}{M_V + 1}\right)^2 L \mathbb{E}_{\xi_k}[\|\nabla F_{S_k}\left(w_k\right)\|_2^2] \tag{33}$$

By the assumption that $\mathbb{E}_{\xi_k}[\|\nabla F_{S_k}\left(w_k\right)\|^2] - \|\nabla F(w_k)\|^2 \leq M_V \|\nabla F(w_k)\|^2 + M$, we have

$$\mathbb{E}_{\xi_k}[\|\nabla F_{S_k}\left(w_k\right)\|^2] \leq (M_V + 1)\|\nabla F(w_k)\|^2 + M \tag{34}$$

Plug this into the previous equation and obtain:

$$\begin{aligned}
F(w_{k+1}) - F(w_k) &\leq -\frac{\bar{\alpha}}{M_V + 1} \left\|\nabla F(w_k)\right\|^2 + \frac{1}{2}\left(\frac{\bar{\alpha}}{M_V + 1}\right)^2 L \left[\left(M_V + 1\right) \left\|\nabla F(w_k)\right\|^2 + M\right] \\
&= -\frac{\bar{\alpha}}{M_V + 1}\left[1 - \frac{\bar{\alpha} L}{2}\right] \left\|\nabla F(w_k)\right\|^2 + \frac{\bar{\alpha}^2 L}{2(M_V + 1)^2} M \\
&= -\frac{\bar{\alpha}}{2(M_V + 1)} \left\|\nabla F(w_k)\right\|^2 + \frac{\bar{\alpha}^2 L}{2(M_V + 1)^2} M \\
&\leq -\frac{\bar{\alpha}\mu}{M_V + 1}[F(w_k) - F^*] + \frac{\bar{\alpha}^2 L}{2(M_V + 1)^2} M
\end{aligned} \tag{35}$$

rearranging terms, subtracting F^* from both sides and obtain:

$$\begin{aligned}
F(w_{k+1}) - F^* &\leq \left(1 - \frac{\bar{\alpha}\mu}{M_V + 1}\right)[F(w_k) - F^*] + \frac{\bar{\alpha}^2 L}{2(M_V + 1)^2} M \\
&\leq \left(1 - \frac{2(1-c_1)\mu}{\tau(M_V + 1)L}\right)[F(w_k) - F^*] + \frac{M}{2L(M_V + 1)^2}
\end{aligned} \tag{36}$$

We subtract $\frac{\tau M}{4\mu(1-c_1)(M_V+1)}$ from both sides and obtain

$$F(w_{k+1}) - F^* - \frac{\tau M}{4\mu(1-c_1)(M_V+1)} \leq \left(1 - \frac{2(1-c_1)\mu}{\tau(M_V + 1)L}\right)\left[F(w_k) - F^* - \frac{\tau M}{4\mu(1-c_1)(M_V+1)}\right] \tag{37}$$

Recursively apply this argument from k_0 to k to attain the argument.