2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

A Stochastic Quasi-Newton Method for Non-convex Optimization with Non-uniform Smoothness

Zhenyu Sun and Ermin Wei

***Abstract*— Classical convergence analyses for optimization algorithms rely on the widely-adopted uniform smoothness assumption. However, recent experimental studies have demonstrated that many machine learning problems exhibit non-uniform smoothness, meaning the smoothness factor is a function of the model parameter instead of a universal constant. In particular, it has been observed that the smoothness grows with respect to the gradient norm along the training trajectory. Motivated by this phenomenon, the recently introduced (L_0, L_1)-smoothness is a more general notion, compared to traditional L-smoothness, that captures such positive relationship between smoothness and gradient norm. Under this type of non-uniform smoothness, existing literature has designed stochastic first-order algorithms by utilizing gradient clipping techniques to obtain the optimal $\mathcal{O}(\epsilon^{-3})$ sample complexity for finding an ϵ-approximate first-order stationary solution. Nevertheless, the studies of quasi-Newton methods are still lacking. Considering higher accuracy and more robustness for quasi-Newton methods, in this paper we propose a fast stochastic quasi-Newton method when there exists non-uniformity in smoothness. Leveraging gradient clipping and variance reduction, our algorithm can achieve the best-known $\mathcal{O}(\epsilon^{-3})$ sample complexity and enjoys convergence speedup with simple hyperparameter tuning. Our numerical experiments show that our proposed algorithm outperforms the state-of-the-art approaches.**

I. Introduction

In this paper, we consider the following stochastic optimization problem:

$$\min_{x} \quad F(x) := \mathbb{E}_{\xi\sim P}[l(x;\xi)] \tag{1}$$

where $x \in \mathbb{R}^d$ is the model parameter; ξ is the random variable drawn from some unknown distribution P; $l(\cdot;\cdot)$ is the loss function. Note that the expectation is taken over the randomness of data distribution. A well-known example of problem (1) is *empirical risk minimization* (ERM), where the distribution P is taken as the point mass at each sampled point ξ_i, i.e.,

$$\min_{x} \quad \frac{1}{n}\sum_{i=1}^{n} l(x;\xi_i) \tag{2}$$

with n being the sample size.

In modern machine learning and deep learning contexts, both (1) and (2) may be non-convex due to the structures of data distribution and the embedding of deep neural networks in the loss function, which make them difficult, if not impossible, to solve analytically. In order to solve the aforementioned problem, the idea of employing stochastic approximation (SA) is proposed [31], where stochastic gradient descent (SGD) serves as a classical SA method. Particularly, SGD mimics the gradient descent method by replacing the full gradient with stochastic gradient samples drawn from the dataset. Since only first-order information (i.e., gradients) is utilized, SGD enjoys good simplicity. Then, stochastic first-order methods have been extensively studied and widely used in the deep learning context due to their easy implementation. Techniques of variance reduction have been adopted to achieve low sample complexity [13], [7], [8], [15].

However, these first-order methods may suffer from poor performances in the sense that gradient information fails to capture the curvature properties of objective functions. In contrast, Newton's methods, which leverage the second-order Hessian information, can possibly achieve better accuracy [33], [1]. Nevertheless, computing Hessian matrix and its inverse can be computationally expensive, thus motivates the emergence of quasi-Newton methods. In particular, stochastic quasi-Newton (SQN) methods only use first-order information to approximate the Hessian or its inverse such that it can be further integrated with stochastic gradient-based approaches to achieve better performances, compared to first-order methods [5], [36]. The appealing advantages of SQN methods, e.g., efficiency, robustness and accuracy [6], draw great attention from researchers. Recent studies have shown that by leveraging variance-reduced tools proposed in [8], $\mathcal{O}(\epsilon^{-3})$ sample complexity can be achieved for SQN [39], which is order-same as the lower bound by first-order methods [3].

Note that for both pure gradient-based methods and SQN methods, existing works all assume the uniform smoothness condition on the objective function for convergence analysis, i.e. $\nabla F(x)$ is L-Lipschitz continuous with universal constant L for any model parameter x. However, recent experimental evidence has revealed that the Lipschitz constant of the objective smoothness grows in the gradient norm along the training trajectory [38]. Figure 1 illustrates this fact. This non-uniform smoothness indicates that many important objective functions fail to exhibit a universal upper bound on the smoothness constants and hence many existing convergence analyses have limited applicability in practical settings. Inspired by this phenomenon, [38] introduces a more general notion of smoothness named (L_0, L_1)-*smoothness*, where the smoothness increases linearly in the gradient norm, i.e., $\|\nabla^2 F(x)\| \le L_0 + L_1\|\nabla F(x)\|$. Later in [20], it has been shown analytically that some reinforcement learning

This work was supported by NSF ECCS-2030251, CMMI-2024774 and ECCS-2216970.

Zhenyu Sun is with Electrical & Computer Engineering, Northwestern University, Evanston, IL 60208, USA. zhenyusun2026@u.northwestern.edu

Ermin Wei is with Electrical & Computer Engineering and Industrial Engineering & Management Science, Northwestern University, Evanston, IL 60208, USA. ermin.wei@northwestern.edu

(RL) problems are characterized by this particular type of smoothness.

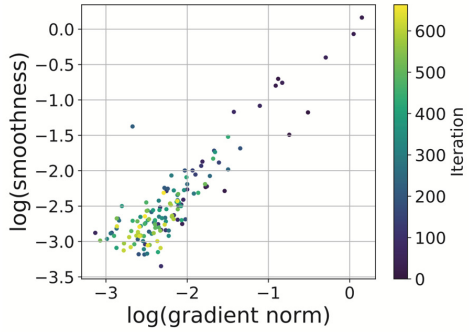

Fig. 1. Smoothness increases with gradient norm along the training trajectory (figure taken from [38]

As a result of non-uniform smoothness, gradients can grow rapidly, gradient clipping techniques have been applied to mitigate this effect. Under the stochastic setting, [38] shows that clipped SGD can find an ϵ-approximate first-order stationary solution with $\mathcal{O}(\epsilon^{-4})$ samples. Further combining the idea of variance reduction, [30] proposes (L_0, L_1)-Spider that achieves the optimal $\mathcal{O}(\epsilon^{-3})$ sample complexity. However, SQN methods are still lacking, especially when non-uniform smoothness exists. Therefore, in order to bridge this gap, we aim to design a SQN method that captures objective's non-uniform smoothness, while obtaining low sample complexity.

A. Related work

a) SGD-based methods in non-convex optimization: Stochastic first-order algorithms are widely used and well-investigated in modern machine learning tasks. For non-convex optimization, the classical SGD algorithm has been proven to achieve an overall $\mathcal{O}(\epsilon^{-4})$ samples to find an ϵ-approximate first-order stationary point [10], [11]. Later, several SGD variants equipped with variance reduction techniques emerge, including SAGA [7], SVRG [13], SARAH [24] etc. Due to the effectiveness of variance reduction, it is shown that SVRG improves the sample complexity to $\tilde{\mathcal{O}}(\epsilon^{-10/3})$ for non-convex optimization [2], [29]. Recently, [8] proposed Spider, a less costly approach that tracks the true gradients in an iterative way. Spider improves the sample complexity to $\mathcal{O}(\epsilon^{-3})$, which matches the lower bound complexity shown in [3]. This means Spider is an order-optimal algorithm to find stationary points of non-convex smooth functions. A nested variance-reduced algorithm in [42] achieves similar complexity bound. Other works with optimal complexity can be found therein [27], [16], [17].

b) SQN methods in non-convex optimization: Newton's methods usually have rapid convergence speed due to the use of Hessian information [4], [23], which have recently been applied to non-convex optimization problems [14], [41]. However, prohibitively expensive computation cost in calculating Hessian matrix and its inverse hinders the application of Newton's methods to large-scale machine learning problems. Quasi-Newton methods then serve as effective candidates to address this computation challenge. SQN methods are extensively studied for both strongly convex and convex optimization problems [22], [32]. Variance reduction was also embedded with L-BFGS method recently [23], [19]. In terms of non-convex optimization, SdLBFGS was proposed in [36] where $\mathcal{O}(\epsilon^{-4})$ complexity is achieved. Inspired by the variance reduction idea as in Spider, [39] improved the sample complexity bound to $\mathcal{O}(\epsilon^{-3})$.

c) Gradient clipping and non-uniform smoothness: Gradient clipping has been widely used in training deep learning models to mitigate issues related to very large gradients [21], [9], [26]. The work [38] lays out a theoretical basis to better understand the superior performances of clipping-based algorithms. Due to the positive relation between smoothness and gradient norm as shown in Figure 1, [38] introduced the notion of (L_0, L_1)-smoothness, which extends the conventional L-smoothness. [20] then introduced a more general smoothness notion and proved that any RL problem with a finite Markov decision process (MDP) satisfies (L_0, L_1)-smoothness. Under (L_0, L_1)-smoothness, [28] studies clipping for incremental gradient-based methods. Convergence analysis on the stochastic normalized gradient descent method is provided by [40] for (L_0, L_1)-smooth non-convex optimization. Very recently, [30] combines gradient clipping with variance reduction techniques to show the optimal $\mathcal{O}(\epsilon^{-3})$ sample complexity under (L_0, L_1)-smoothness. Apart from optimization literature, this particular type of smoothness has been studied for variational inference problems in [34].

B. Our contributions

We summarize our main contributions as follows: (1) We propose a general stochastic quasi-Newton framework for non-convex optimization with (L_0, L_1)-smoothness, which works with any Hessian inverse approximation that is positive definite. Convergence analysis and sample complexity are given, where the complexity bound is $\mathcal{O}(\epsilon^{-3}\lambda_M^2/\lambda_m^2)$ with λ_m and λ_M being the smallest and largest eigenvalues of the Hessian inverse approximation. (2) We then design an adaptive L-BFGS-based algorithm that guarantees the positive definiteness of the generated Hessian inverse approximation. Moreover, by tuning the design parameters, we can control these eigenvalues, which further allows us to control the sample complexity. Table I provides a clear comparison of our algorithm and other state-of-the-art methods, where Clip represents "clipping"; VR represents "variance reduction". To the best of our knowledge, our approach is the first stochastic quasi-Newton method that achieves the best-known $\mathcal{O}(\epsilon^{-3})$ sample complexity to find an ϵ-approximate first-order stationary solution when the objective function is not uniformly smooth.

II. Preliminaries

In this section, we describe the goal of our algorithm in terms of an optimality condition and also present a condition

TABLE I

SAMPLE COMPLEXITY AND SMOOTHNESS FOR STOCHASTIC NON-CONVEX OPTIMIZATION ALGORITHMS

Reference	Algorithm type	Technique	Complexity	Smoothness
SGD [10]	first-order	✗	$\mathcal{O}(\epsilon^{-4})$	L-smooth
SVRG [2]	first-order	VR	$\tilde{\mathcal{O}}(\epsilon^{-10/3})$	L-smooth
ClippedSGD [38]	first-order	Clip	$\mathcal{O}(\epsilon^{-4})$	(L_0, L_1)-smooth
Spider [8]	first-order	VR	$\mathcal{O}(\epsilon^{-3})$	L-smooth
(L_0, L_1)-Spider [30]	first-order	Clip; VR	$\mathcal{O}(\epsilon^{-3})$	(L_0, L_1)-smooth
SdLBFGS [36]	quasi-Newton	✗	$\mathcal{O}(\epsilon^{-4})$	L-smooth
SSQNMED [39]	quasi-Newton	VR	$\mathcal{O}(\epsilon^{-3})$	L-smooth
ClippedSQN (ours)	quasi-Newton	Clip; VR	$\mathcal{O}(\epsilon^{-3})$	(L_0, L_1)-smooth

of non-uniform smoothness, under which our proposed method has performance guarantees.

A. Optimality condition

Noting that finding the global minimum for general stochastic non-convex optimization problems is NP-hard [12], in this work we instead focus on finding an *ϵ-approximate first-order stationary solution*, which is a standard performance measure for non-convex optimization algorithms. Formally, this optimality condition is defined as follows:

Definition 2.1: Given some algorithm $\mathcal{A}$, let $\tilde{x}$ be an output of algorithm $\mathcal{A}$. Then $\tilde{x}$ is said to be an *ϵ-approximate first-order stationary solution* of (1) if $\mathbb{E}\|\nabla F(\tilde{x})\| \leq \epsilon$ for any $\epsilon > 0$.

We note that the expectation in $\mathbb{E}\|\nabla F(\tilde{x})\|$ is taken over the randomness of $\tilde{x}$ because $\mathcal{A}$ is a randomized algorithm.

B. (L_0, L_1)-smoothness

We first provide a more general notion of smoothness, i.e., (L_0, L_1)-smoothness introduced in [38], [30], which captures the non-uniform property of the smoothness factor. It is formally described by the following definition:

Definition 2.2: [30] The differentiable function $F : \mathbb{R}^d \to \mathbb{R}$ is (L_0, L_1)-smooth if for any $x, y \in \mathbb{R}^d$,

$$\|\nabla F(x) - \nabla F(y)\| \leq (L_0 + L_1\|\nabla F(x)\|)\|x - y\| \quad (3)$$

where $L_0 > 0$ and $L_1 \geq 0$ are two constants.

Clearly, (L_0, L_1)-smoothness notion generalizes the traditional Lipschitz-gradient condition, since any function F with Lipschitz gradient satisfying $\|\nabla F(x) - \nabla F(y)\| \leq L\|x - y\|$ for any x and y satisfies (3) with $L_0 = L$ and $L_1 = 0$.

C. Examples with (L_0, L_1)-smooth properties

For the rest of this paper, we focus on the objective functions with (L_0, L_1)-smoothness shown in Definition 2.2. This is motivated by the observation that many machine learning problems have (L_0, L_1)-smoothness. We next illustrate this by two examples.

Proposition 2.3: Consider $F(x) = y\log(\hat{y})$, where $\hat{y} = \sigma(u^T x)$ with $\sigma(\cdot)$ being the sigmoid function and y, u are constant scalars or vectors with suitable dimensions. Then, $F(x)$ is $(L_0, \|u\|)$-smooth for any $L_0 > 0$.

Function F in the above proposition is commonly used in classification tasks with cross-entropy loss [37]. More generally, if x is the parameter of the last layer of the designed neural network and u is the output of the previous layer, we observe from Proposition 2.3 that the Hessian and the gradient are positively related, which also demonstrates the validity of (L_0, L_1)-smoothness in deep learning context. All the proofs are omitted due to space limitation. Please refer to [35] for proof details.

In addition, as shown by [20], in the reinforcement learning (RL) problem with finite Markov Decision Processes (MDPs), (L_0, L_1)-smoothness is also satisfied by the value function:

Proposition 2.4: [20] For any finite MDP, let $V^{\pi_\theta}(s_0)$ be the value function, where π_θ and s_0 are the policy parametrized by θ and the initial state, respectively. Then, $V^{\pi_\theta}(s_0)$ is (L_0, L_1)-smooth for any $L_0 > 0$ and some $L_1 > 0$ determined by problem related constants.

III. A CLIPPED STOCHASTIC QUASI-NEWTON METHOD

In this section we propose a general stochastic quasi-Newton (SQN) framework with clipping techniques that yields fast convergence and low sample complexity for the particular type of non-uniformly smooth functions defined in Section II-B. We first present some assumptions on the objective property and sampling process.

Assumption 3.1: Suppose the loss function $l(x; \cdot)$ is differentiable in x for any x. Then F is averaged (L_0, L_1)-smooth if for any x, y, there exist some constant $L_0 > 0$ and $L_1 \geq 0$ such that

$$\mathbb{E}\|\nabla l(x; \xi) - \nabla l(y; \xi)\|^2 \leq (L_0 + L_1\|\nabla F(x)\|)^2\|x - y\|^2.$$

Assumption 3.2: For any x, we have

$$\mathbb{E}[\nabla l(x; \xi)] = \nabla F(x)$$
$$\mathbb{E}\|\nabla l(x; \xi) - \nabla F(x)\|^2 \leq \sigma^2 < \infty.$$

Our proposed method relies on building an approximation of Hessian inverse, H_k, at each iteration, which satisfies the following assumptions.

Assumption 3.3: There exist some strictly positive constants λ_m and λ_M such that for any $k = 0, \ldots, K$, we have

$$\lambda_m I \preceq H_k \preceq \lambda_M I.$$

Algorithm 1 Clipped Stochastic Quasi-Newton Method

Input: initial point x_0, positive definite H_0, batch size $|\mathcal{S}_1|$ and $|\mathcal{S}_2|$, integers K and r, stepsizes $\{\eta_k\}$
for $k = 0, 1, \ldots, K-1$ **do**
  if $k \bmod r = 0$ **then**
    Draw samples $\mathcal{S}_1$ and compute $v_k = \nabla l(x_k; \mathcal{S}_1)$.
  else
    Draw samples $\mathcal{S}_2$ and compute $v_k = v_{k-1} + \nabla l(x_k; \mathcal{S}_2) - \nabla l(x_{k-1}; \mathcal{S}_2)$.
  end if
  Calculate stepsize η_k from (4).
  Generate a positive definite Hessian inverse approximation H_k at x_k and calculate $H_k v_k$ according to Algorithm 2.
  $x_{k+1} = x_k - \eta_k H_k v_k$.
end for
Output: $\tilde{x}_K$ sampled uniformly from $\{x_k\}_{k=0}^{K-1}$

Assumption 3.4: For any x_k and v_k with $k \geq 1$, H_k depends only on $\xi_{[k-1]}$, where $\xi_{[k]} := (\xi_0, \ldots, \xi_k)$ which is all the samples from iteration 0 to iteration k. Or equivalently, it means that

$$\mathbb{E}[H_k v_k | \xi_{[k-1]}, x_k] = H_k \mathbb{E}[v_k | x_k].$$

It is worth noting that Assumption 3.1 implies (L_0, L_1)-smoothness on F by $\|\nabla F(x) - \nabla F(y)\|^2 \leq \mathbb{E}\|\nabla l(x;\xi) - \nabla l(y;\xi)\|^2$.

We first assume conditions in Assumptions 3.3 and 3.4 are met and defer their justification to the next section where we describe a specific algorithm to generate H_k. We next present Algorithm 1, a general SQN framework that obtains fast convergence and $\mathcal{O}(\epsilon^{-3})$ sample complexity to achieve ϵ-approximate first-order solutions for any stochastic non-convex optimization problem with averaged (L_0, L_1)-smoothness.

Algorithm 1 is inspired by Spider [8]. In particular, Spider maintains an estimate of the true gradient $\nabla F(x_k)$ denoted by v_k and updates v_k by utilizing mini-batches $\mathcal{S}_1$ and $\mathcal{S}_2$ to reduce the variance. Then, assuming L-smooth F, the model parameter is updated by $x_{k+1} = x_k - \eta_k v_k$ with $\eta_k = \min\{1/(2L), \epsilon/(L\|v_k\|)\}$. However, Spider cannot deal with non-uniformly smooth objective functions in the sense that there is no uniform upper bound for $\|\nabla^2 F(x)\|$. If the smoothness parameter grows with respect to the gradient norm, stepsize η_k will keep decreasing and finally vanish when the norm of the gradient is extremely large, which means x_k could get stuck at a non-stationary point in our setting.

In order to deal with the stepsize-vanishing problem caused by non-uniform smoothness, [30] proposes (L_0, L_1)-Spider for (L_0, L_1)-smooth functions. In [30], the authors modify the stepsize as $\eta_k = \min\{1/(2L_0), \epsilon/(L_0\|v_k\|), \epsilon/(L_1\|v_k\|^2)\}$ and retain the update $x_{k+1} = x_k - \eta_k v_k$. However, both Spider and (L_0, L_1)-Spider are first-order methods, which compared to quasi-Newton methods are usually slower and less robust [36]. Adopting the potential speed advantage of quasi-Newton methods to non-uniform smooth objective functions is main contribution of proposing Algorithm 1.

Note that for now we use $H_k v_k$ rather than v_k to update our model, which introduces additional challenges for stepsize design and convergence analysis. Next theorem addresses these challenges and characterizes an upper bound on the sample complexity of finding an ϵ-approximate stationary solution for Algorithm 1. Denoting $\Delta_0 := \mathbb{E}[F(x_0) - F^*]$ with $F^* := \min_x F(x) > -\infty$, we have:

Theorem 3.5: Suppose Assumptions 3.1-3.4 hold. For any $0 < \beta \leq \frac{\lambda_m}{1+\lambda_M^2}$, $0 < c \leq \frac{4\lambda_m - 2\beta(1+\lambda_M^2)}{L_1 \lambda_M^2 \beta(3+\beta^2)}$, defining $h_1^\beta(c) = \lambda_m - \frac{\beta\lambda_M^2}{4}(2 + 3L_1 c)$ and setting $|\mathcal{S}_1| = \frac{2\sigma^2}{\epsilon^2}$, $|\mathcal{S}_2| = \frac{4h_1^\beta(c)^2}{\epsilon}$, $r = \frac{1}{\epsilon}$ with stepsizes selected by

$$\eta_k = \min\left\{ \frac{h_1^\beta(c)}{2L_0\lambda_M^2}, \frac{h_1^\beta(c)\epsilon}{L_0\lambda_M^2\|v_k\|}, \frac{h_1^\beta(c)\epsilon}{L_1\lambda_M^2\|v_k\|^2} \right\}, \quad (4)$$

then we can achieve

$$\mathbb{E}\|\nabla F(\tilde{x}_K)\| = \mathcal{O}((1+\beta^{-2})\epsilon)$$

for $\tilde{x}_K$ being the output of Algorithm 1 by running $K = \lceil \frac{2L_0\lambda_M^2\Delta_0}{h_1^\beta(c)^2\epsilon^2} \rceil$ iterations. Moreover, the total sample complexity is with the order of $\mathcal{O}(\epsilon^{-3}\lambda_M^2/\lambda_m^2)$.

Remark 3.6: In Theorem 3.5, when $0 < \beta \leq \frac{\lambda_m}{1+\lambda_M^2}$, the upper bound for c is positive. Moreover, for c chosen as stated, one can easily check that $0 < \beta^2\big(\frac{1}{2\beta} + \frac{L_1\beta\lambda_M^2 c}{4}\big) \leq h_1^\beta(c)$, indicating the proper choice for η_k.

IV. Generating H_k with Controllable (λ_m, λ_M)

The convergence result in Theorem 3.5 is established based on Assumptions 3.3 and 3.4, both of which we yet need to justify. In this section, we propose one approach to generating H_k that satisfies these two assumptions. Specifically, we adopt the idea of L-BFGS method proposed by [18] to our setting, which simultaneously preserves the positive definiteness of H_k and reduces the computational cost. Moreover, certain design parameters in our algorithm can be adjusted based on problem features, which makes our algorithm adaptive. Due to these reasons, we call it Stochastic Adaptive L-BFGS method.

We define the following quantities for notational simplicity: $\bar{g}_k := \nabla l(x_k; \mathcal{S}_{k-1}) = \frac{1}{|\mathcal{S}_{k-1}|}\sum_{i\in\mathcal{S}_{k-1}} \nabla l(x_k; \xi_{i,k-1})$ and $g_k := \nabla l(x_k; \mathcal{S}_k) = \frac{1}{|\mathcal{S}_k|}\sum_{i\in\mathcal{S}_k} \nabla l(x_k; \xi_{i,k})$, where $\mathcal{S}_k$ is the batch of samples drawn at iteration k and $\xi_{i,k}$ is the i-th sample in $\mathcal{S}_k$. And let $y_{k-1} := \bar{g}_k - g_{k-1}$, $s_{k-1} := x_k - x_{k-1}$.

At iteration k, we maintain a memory with size p to store two sequences $\{s_j\}$ and $\{y_j\}$ for $j = k-p, \ldots, k-1$. Given the initial $H_{k,0}$, keep updating $H_{k,i}$ with $i = p+1+j-k$ by utilizing $\{s_j\}$ and $\{y_j\}$ in the memory for p times. Finally, output $H_{k,p}$ as the approximation of the Hessian inverse at x_k.

Specifically, the initial $H_{k,0}$ is given by

$$H_{k,0} = c_k^{-1} I, \text{with } c_k = \max\left\{ \delta, w_{k-1}\frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}} \right\} \quad (5)$$

where $\delta > 0$ and $w_{k-1} > 0$ are design parameters.

Algorithm 2 Stochastic Adaptive L-BFGS Method

Input: memory size p, positive scalars δ, q, κ, model parameters x_k and x_{k-1}, mini-batch samples $\mathcal{S}_{k-1}$, sequences s_j, $\bar{y}_j$, ρ_j with $j = k-p, \ldots, k-2$, $u_0 = v_k$.

Compute $g_{k-1} = \nabla l(x_{k-1}; \mathcal{S}_{k-1})$ and $\bar{g}_k = \nabla l(x_k; \mathcal{S}_{k-1})$.
Set $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = \bar{g}_k - g_{k-1}$.
Compute Γ_{k-1} by (9) and c_k by (5)
Calculate $\bar{y}_{k-1}$ through (6) and $\rho_{k-1} = (s_{k-1}^T \bar{y}_{k-1})^{-1}$.
for $i = 0, \ldots, \min\{p, k-1\} - 1$ **do**
  Compute $\nu_i = \rho_{k-i-1} u_i^T s_{k-i-1}$.
  Compute $u_{i+1} = u_i - \nu_i \bar{y}_{k-i-1}$.
end for
Calculate $\alpha_0 = c_k^{-1} u_p$.
for $i = 0, \ldots, \min\{p, k-1\} - 1$ **do**
  Compute $\zeta_i = \rho_{k-p+i} \alpha_i^T \bar{y}_{k-p+i}$.
  Compute $\alpha_{i+1} = \alpha_i + (\nu_{p-i-1} - \zeta_i) s_{k-p+i}$.
end for
Output: $\alpha_p = H_k v_k$.

Then we define

$$\bar{y}_{k-1} = w_{k-1}\big(\theta_{k-1} y_{k-1} + (1-\theta_{k-1}) H_{k,0}^{-1} s_{k-1}\big) \quad (6)$$

where

$$\theta_{k-1} = \begin{cases} \frac{(1-q_{k-1})\bar{\mu}_{k-1}}{\bar{\mu}_{k-1} - s_{k-1}^T y_{k-1}} & , \quad s_{k-1}^T y_{k-1} < q_{k-1}\bar{\mu}_{k-1} \\ 1 & , \quad \text{otherwise} \end{cases}$$

with $\bar{\mu}_{k-1} := s_{k-1}^T H_{k,0}^{-1} s_{k-1}$ and some designed $0 < q_{k-1} < 1$. Consider the update of $H_{k,i}$ as follows: for any $i = 1, \ldots, p$, $j = k - (p - i + 1)$,

$$H_{k,i} = (I - \rho_j s_j \bar{y}_j^T) H_{k,i-1} (I - \rho_j \bar{y}_j s_j^T) + \rho_j s_j s_j^T, \quad (7)$$

where $\rho_j = (s_j^T \bar{y}_j)^{-1}$. Finally, set $H_k = H_{k,p}$ as the Hessian inverse approximation stated in Algorithm 1. Note that the designed H_k by the above process satisfies Assumption 3.4.

Justified by the following lemma, ρ_j's are guaranteed to be strictly positive for all j, hence $H_{k,i}$'s are well defined. Further, we have positive definite $H_{k,i}$ for any k and i.

Lemma 4.1: Considering the update (7) with $H_{k,0}$ defined by (5), then $s_k^T \bar{y}_k \geq w_k q_k s_k^T H_{k+1,0} s_k > 0, \forall k \geq 0$ and $H_{k,i} \succ 0, \forall i = 1, \ldots, p$.

Algorithm 2 summarizes a practical way to obtain $H_k v_k$ rather than H_k, which essentially achieves lower computational complexity. In particular, note that two loops in Algorithm 2 involve p scalar-vector multiplications and p vector inner products, hence indicating the total number of multiplications in Algorithm 2 is with the order of $\mathcal{O}(pd)$. As suggested in [25], p can be chosen to be much smaller than d in high-dimensional cases. Thus, our Adaptive L-BFGS method enjoys computational efficiency in modern neural-network based optimization.

Algorithm 2, a stochastic adaptive L-BFGS method, boasts the following two desirable features: first, by borrowing the idea of the L-BFGS method, the computational cost is now reduced to $\mathcal{O}(pd)$; second, adaptivity in w_{k-1} and q_{k-1} will enable us to make the control of λ_m, λ_M much easier.

Before we formally characterize λ_m and λ_M, the following assumption is needed.

Assumption 4.2: Given any sample ξ, the following condition is satisfied: for any $x, y \in \mathbb{R}^d$

$$\|\nabla l(x;\xi) - \nabla l(y;\xi)\| \leq (\gamma_0 + \gamma_1 \|\nabla l(x;\xi)\|)\|x - y\|$$

where $\gamma_0 > 0$ and $\gamma_1 \geq 0$ are two constants.

Given Assumptions 4.2 and 3.2, we can easily deduce Assumption 3.1 by noting that

$$\begin{aligned} &\mathbb{E}\|\nabla l(x;\xi) - \nabla l(y;\xi)\|^2 \\ &\leq \mathbb{E}\big(\gamma_0 + \gamma_1 \|\nabla l(x;\xi)\|\big)^2 \|x - y\|^2 \\ &\leq (2\gamma_0^2 + 2\gamma_1^2 \mathbb{E}\|\nabla l(x;\xi)\|^2)\|x - y\|^2 \\ &\leq \big(2(\gamma_0^2 + \gamma_1^2\sigma^2) + 2\gamma_1^2 \|\nabla F(x)\|^2\big)\|x - y\|^2 \\ &\leq \big(\sqrt{2(\gamma_0^2 + \gamma_1^2\sigma^2)} + \gamma_1\sqrt{2}\|\nabla F(x)\|\big)^2 \|x - y\|^2 \end{aligned} \quad (8)$$

with $L_0 = \sqrt{2(\gamma_0^2 + \gamma_1^2\sigma^2)}, L_1 = \gamma_1\sqrt{2}$ and we make use of $\mathbb{E}\|\nabla l(x;\xi)\|^2 = \mathbb{E}\|\nabla l(x;\xi) - \nabla F(x)\|^2 + \|\nabla F(x)\|^2 \leq \|\nabla F(x)\|^2 + \sigma^2$.

Remark 4.3: In [36], [39], a special case of this assumption is adopted to bound λ_m and λ_M, where the authors set γ_1 in Assumption 4.2 to be zero. However, due to the non-uniform smoothness condition in our setting (see Assumption 3.1), γ_1 is in general nonzero as shown by (8).

Define

$$\Gamma_{k-1} = \psi_0 + \frac{\gamma_1^2}{|\mathcal{S}_{k-1}|} \sum_{l \in \mathcal{S}_{k-1}} \|\nabla l(x_{k-1}; \xi_{l,k-1})\| \quad (9)$$

with $\psi_0 = \gamma_0(1 + e^{\gamma_1/L_0}/L_0), L_0 = \sqrt{2(\gamma_0^2 + \gamma_1^2\sigma^2)}$ as in (8). Then we are ready to state the results that specify the quantities of λ_m and λ_M in Assumption 3.3.

Theorem 4.4: Suppose Assumption 4.2 hold. For any $k \geq 0$, setting $q_k = q\Gamma_k^4$ such that $q_k \in (0,1)$ and $w_k = \kappa^2/\Gamma_k^2$ for some $q, \kappa > 0$, we have

$$\begin{aligned} \lambda_m &= \left(\delta + \frac{\kappa^2 p}{q\gamma_0^4}\left(\frac{1}{\delta} + \frac{\delta\gamma_0 + \kappa^2}{\gamma_0^3} + \frac{p + 2q\gamma_0^4}{2p\gamma_0}\right)\right)^{-1} \\ \lambda_M &= \frac{1}{\delta\gamma_0^2\kappa^2 q}\frac{a^p - 1}{a - 1} \end{aligned}$$

where $a = 1 + \frac{2}{\delta\gamma_0\kappa^2 q} + \big(\frac{1}{\delta\gamma_0\kappa^2 q}\big)^2$.

Then the following corollary provides some guidance on how to adjust λ_m, λ_M as we expect.

Corollary 4.5: Suppose all conditions in Theorem 4.4 hold. Then $\lambda_m = \Omega\big(\frac{\delta q}{\kappa^2} + \frac{q}{\kappa^2\delta}\big)$ and $\lambda_M = \mathcal{O}\left(\frac{1}{(\delta\kappa^2 q)^{2p+1}}\right)$.

Corollary 4.5 essentially indicates that we could simply tune hyperparameters δ and q to control λ_m and λ_M simultaneously. For instance, the ratio λ_M/λ_m gets bigger if we decrease δ or q and vice versa. Combining this observation together with Theorem 3.5, we conclude that when β is fixed, increasing δ or q could decrease λ_M/λ_m and hence correspondingly reduce the sample complexity. This in turn achieves convergence speedup.

If $\gamma_1 = 0$, Assumption 4.2 reduces to the same one used in [36] to obtain bounds on λ_m and λ_M. Even in this case, [36] falls short in controlling λ_m and λ_M by tuning hyperparameters. Therefore, our method is more general and can be adaptive to problem features.

V. Experiments

In this section, we numerically evaluate the performances of our Clipped Stochastic Quasi-Newton method (i.e., Algorithm 1 and Algorithm 2). We consider the practical *empirical risk minimization* (ERM) version of problem (1), i.e., (2). Both first-order and quasi-Newton methods are compared with ours, including mini-batch SGD, Spider [8], (L_0, L_1)-Spider [30], SdLBFGS [36].

We first generate a synthetic dataset as follows: the i-th data sample is denoted by (a_i, b_i), where $a_i \in \mathbb{R}^d$ is the feature and $b_i \in \mathbb{R}$ is a scalar. Moreover, the feature vector a_i is a sparse vector with 10% nonzero elements and is drawn uniformly from $[0, 1]^d$. Each b_i is set as $b_i = \text{sign}(u_i^T a_i)$ for some $u_i \in \mathbb{R}^d$ drawn uniformly from $[-1, 1]^d$. We set $d = 100$.

For mini-batch SGD and SdLBFGS, we choose a fixed batch size of 500. For Spider, (L_0, L_1)-Spider and our method, we set $|\mathcal{S}_1| = 2000$ and $|\mathcal{S}_2| = 100$. For two quasi-Newton methods, we select the memory size as $p = 5$ as suggested in [25]. It is worth noting that given number of iteration K, our method exactly draw the same number of samples as those in Spider and (L_0, L_1)-Spider, when the batch sizes $|\mathcal{S}_1|$ and $\mathcal{S}_2$ are fixed, since Algorithm 2 involves no additional sampling.

A. Non-convex robust linear regression

First, we compare the above-mentioned algorithms under the following non-convex robust linear regression problem [39]:

$$F(x) = \frac{1}{n}\sum_{i=1}^{n} l(b_i - a_i^T x),$$

where the non-convex loss function is defined by $l(x) = \log\left(\frac{x^2}{2} + 1\right)$. The initial value x_0 is drawn from a multi-dimensional normal distribution.

Figure 2 shows the numerical performances of five algorithms given the robust linear regression problem. On one hand, we can observe that two quasi-Newton methods (SdLBFGS and ours) enjoy faster convergence speed, compared to first-order methods. On the other hand, our method finally achieves a smaller training error, although at the beginning converges slower compared to SdLBFGS.

B. Non-convex logistic regression

We then test these algorithms by considering the following non-convex logistic regression problem [6]. In particular, the objective takes the following form:

$$F(x) = -\frac{1}{n}\sum_{i=1}^{n} b_i \log(\sigma(a_i^T x)) + (1 - b_i)\log(\sigma(-a_i^T x))$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. The starting point x_0 is drawn from $[-1, 1]^d$ uniformly. The numerical comparison among algorithms is illustrated in Figure 3. Clearly, our method outperforms other algorithms for achieving better speed and accuracy.

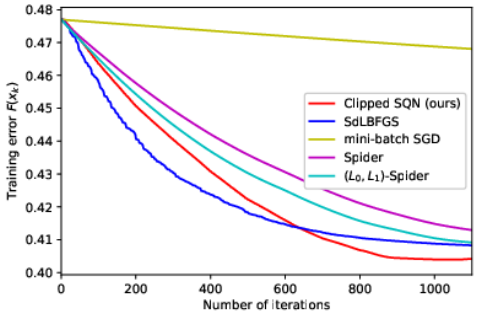

Fig. 2. Training errors for algorithms solving non-convex robust linear regression problem

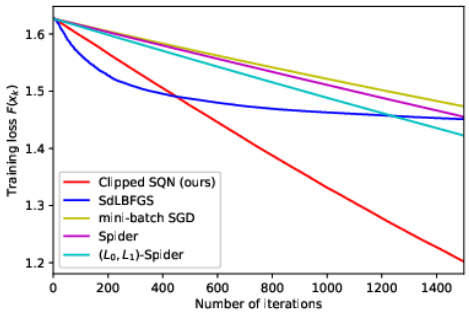

Fig. 3. Training errors for algorithms solving non-convex logistic regression problem

VI. Conclusion

In this paper, we study the stochastic quasi-Newton method for non-convex optimization problems. Instead of limiting objective functions to be L-smooth, we focus on a more general non-uniform smoothness notion called (L_0, L_1)-smoothness. Then, we propose a stochastic quasi-Newton method by leveraging clipping techniques such that $\mathcal{O}(\epsilon^{-3})$ sample complexity can be achieved. Furthermore, we propose an adaptive L-BFGS method such that the eigenvalues of the Hessian inverse approximation can be controlled, hence allowing us to control the convergence speed. Numerical studies then are presented, which demonstrate a better performance of our method, compared to existing algorithms.

References

[1] Z. Allen-Zhu. Natasha 2: Faster non-convex optimization than sgd. *Advances in neural information processing systems*, 31, 2018.

[2] Z. Allen-Zhu and E. Hazan. Variance reduction for faster non-convex optimization. In *International conference on machine learning*, pages 699–707. PMLR, 2016.

[3] Y. Arjevani, Y. Carmon, J. C. Duchi, D. J. Foster, N. Srebro, and B. Woodworth. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, 199(1-2):165–214, 2023.

[4] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[6] H. Chen, H.-C. Wu, S.-C. Chan, and W.-H. Lam. A stochastic quasi-newton method for large-scale nonconvex optimization with applications. *IEEE transactions on neural networks and learning systems*, 31(11):4776–4790, 2019.

[7] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

[8] C. Fang, C. J. Li, Z. Lin, and T. Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in neural information processing systems*, 31, 2018.

[9] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR, 2017.

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

[11] S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

[12] C. J. Hillar and L.-H. Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

[13] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

[14] J. M. Kohler and A. Lucchi. Sub-sampled cubic regularization for non-convex optimization. In *International Conference on Machine Learning*, pages 1895–1904. PMLR, 2017.

[15] L. Lei, C. Ju, J. Chen, and M. I. Jordan. Non-convex finite-sum optimization via scsg methods. *Advances in Neural Information Processing Systems*, 30, 2017.

[16] Z. Li, H. Bao, X. Zhang, and P. Richtárik. Page: A simple and optimal probabilistic gradient estimator for nonconvex optimization. In *International conference on machine learning*, pages 6286–6295. PMLR, 2021.

[17] Z. Li, S. Hanzely, and P. Richtárik. Zerosarah: Efficient nonconvex finite-sum optimization with zero full gradient computation. *arXiv preprint arXiv:2103.01447*, 2021.

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

[19] A. Lucchi, B. McWilliams, and T. Hofmann. A variance reduced stochastic newton method. *arXiv preprint arXiv:1503.08316*, 2015.

[20] J. Mei, Y. Gao, B. Dai, C. Szepesvari, and D. Schuurmans. Leveraging non-uniformity in first-order non-convex optimization. In *International Conference on Machine Learning*, pages 7555–7564. PMLR, 2021.

[21] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing lstm language models. arxiv 2017. *arXiv preprint arXiv:1708.02182*, 2017.

[22] A. Mokhtari and A. Ribeiro. Res: Regularized stochastic bfgs algorithm. *IEEE Transactions on Signal Processing*, 62(23):6089–6104, 2014.

[23] P. Moritz, R. Nishihara, and M. Jordan. A linearly-convergent stochastic l-bfgs algorithm. In *Artificial Intelligence and Statistics*, pages 249–258. PMLR, 2016.

[24] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International conference on machine learning*, pages 2613–2621. PMLR, 2017.

[25] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 1999.

[26] M. E. Peters, M. Neumann, R. L. Logan IV, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith. Knowledge enhanced contextual word representations. *arXiv preprint arXiv:1909.04164*, 2019.

[27] N. H. Pham, L. M. Nguyen, D. T. Phan, and Q. Tran-Dinh. Proxsarah: An efficient algorithmic framework for stochastic composite nonconvex optimization. *The Journal of Machine Learning Research*, 21(1):4455–4502, 2020.

[28] J. Qian, Y. Wu, B. Zhuang, S. Wang, and J. Xiao. Understanding gradient clipping in incremental gradient methods. In *International Conference on Artificial Intelligence and Statistics*, pages 1504–1512. PMLR, 2021.

[29] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323. PMLR, 2016.

[30] A. Reisizadeh, H. Li, S. Das, and A. Jadbabaie. Variance-reduced clipping for non-convex optimization. *arXiv preprint arXiv:2303.00883*, 2023.

[31] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[32] N. N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-newton method for online convex optimization. In *Artificial intelligence and statistics*, pages 436–443. PMLR, 2007.

[33] J. Sohl-Dickstein, B. Poole, and S. Ganguli. Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods. In *International Conference on Machine Learning*, pages 604–612. PMLR, 2014.

[34] L. Sun, A. Karagulyan, and P. Richtarik. Convergence of stein variational gradient descent under a weaker smoothness condition. In *International Conference on Artificial Intelligence and Statistics*, pages 3693–3717. PMLR, 2023.

[35] Z. Sun and E. Wei. A stochastic quasi-newton method for non-convex optimization with non-uniform smoothness. *arXiv preprint arXiv:2403.15244*, 2024.

[36] X. Wang, S. Ma, D. Goldfarb, and W. Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

[37] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9:611–629, 2018.

[38] J. Zhang, T. He, S. Sra, and A. Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*, 2019.

[39] Q. Zhang, F. Huang, C. Deng, and H. Huang. Faster stochastic quasi-newton methods. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):4388–4397, 2021.

[40] S.-Y. Zhao, Y.-P. Xie, and W.-J. Li. On the convergence and improvement of stochastic normalized gradient descent. *Science China Information Sciences*, 64:1–13, 2021.

[41] D. Zhou, P. Xu, and Q. Gu. Stochastic variance-reduced cubic regularized newton methods. In *International Conference on Machine Learning*, pages 5990–5999. PMLR, 2018.

[42] D. Zhou, P. Xu, and Q. Gu. Stochastic nested variance reduction for nonconvex optimization. *The Journal of Machine Learning Research*, 21(1):4130–4192, 2020.