Article

Accelerating Symmetric Rank 1 Quasi-Newton Method with Nesterov's Gradient for Training Neural Networks

S. Indrapriyadarsini [1], Shahrzad Mahboubi [2], Hiroshi Ninomiya [2], Takeshi Kamio [3] and Hideki Asai [4]

[1] Graduate School of Science and Technology, Shizuoka University; s.indrapriyadarsini.17@shizuoka.ac.jp
[2] Graduate School of Electrical and Information Engineering, Shonan Institute of Technology; 20T2502@sit.shonan-it.ac.jp ; ninomiya@info.shonan-it.ac.jp
[3] Graduate School of Information Sciences, Hiroshima City University; kamio@hiroshima-cu.ac.jp
[4] Research Institute of Electronics, Shizuoka University; asai.hideki@shizuoka.ac.jp
† Correspondence: s.indrapriyadarsini.17@shizuoka.ac.jp

Abstract: Neural networks are popularly used in several applications. Optimization algorithms play an important in training these neural networks. Several optimizers such as SGD, Adam etc., and acceleration techniques such as the Nesterov's acceleration have been proposed in the last few years. However, much of these works revolve around first order-methods. Incorporating second order curvature information in gradient based methods have shown to improve convergence drastically despite its computational intensity. Recently, the Nesterov's accleration applied to BFGS quasi-Newton method was shown to have improved performance. This paper investigates accelerating the Symmetric Rank-1 (SR1) quasi-Newton method with the Nesterov's gradient for training neural networks and briefly discuss its convergence. The performance of the proposed method is evaluated on a function approximation and image classification problem. From the results, it can confirmed that the SR1 method can be effectively accelerated using the Nesterov's acceleration.

Keywords: Neural networks; quasi-Newton; symmetric rank-1; Nesterov's accelerated gradient; limited memory

1. Introduction

Neural networks have shown to have great potential in several applications. A majority of recent applications employ large neural network models trained using massive amounts of data, thereby imposing high computational load and storage memory. Hence, there is a great demand for large scale algorithms that can train neural networks effectively and efficiently. Neural network training posses several challenges such as ill-conditioning, hyperparameter tuning, exploding and vanishing gradients, saddle points, etc. Optimization forms the core of machine learning, deep learning and neural networks. Gradient based algorithms have been widely used in optimization and can be broadly categorized as (1) first order methods (eg. SGD, Adam) and (2) higher order methods (eg. Newton method, quasi-Newton method), each with its own pros and cons. Much progress has been made in the last 20 years in designing and implementing robust and efficient methods and yet there are many classes of applications where current state of the art optimizers fails.

Thus, the training algorithm employed plays an important role. Much progress has been made in the last 20 years in designing and implementing robust and efficient methods and yet there are many classes of applications where current state of the art optimizers fails. Gradient based algorithms have been widely used in optimization and can be categorized as (1) first order methods (eg. SGD, Adam) (2) higher order methods (eg. Newton method, quasi-Newton method) and (3) heuristic derivative-free methods (eg. coordinate descent, SPSA), each with its own pros and cons.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

First order methods are most commonly used due to their simplicity and low computational complexity. The simple gradient descent method is the simplest first order method.

The optimization algorithm used in training the neural networks plays an important role. Gradient based algorithms are popularly used in training these neural networks and cna be broadly categorized into first and higher order methods. . Second order methods have shown to have better convergence than first order methods in several highly non-linear problems. However, the computational cost incurred has been a major drawback and thus quasi-Newton methods have been popularly used. Among the quasi-Newton methods, the BFGS method is widely used in training neural networks. Recently [1][2] proposed accelerating the BFGS method using the Nesterov's accelerated gradient and momentum terms. In this study, we explore if the Nesterov's acceleration can be applied to other quasi-Newton methods as well. Thus, this paper proposes a Nesterov's accelerated LSR1 (L-SR1-N) and momentum accelerated LSR1 (L-MoSR1) methods for training neural networks.

Gradient based methods are popularly used in training NNs and can be broadly classified as first and second order methods. Despite the high computational cost, second order methods such as the BFGS quasi-Newton method have shown to have faster convergence compared to first order methods. Incorporating second order curvature information in stochastic settings is a challenging task and has been an active area of research. This paper proposes a stochastic (online) momentum accelerated quasi-Newton method in both its full and limited memory forms for solving large scale non-convex optimization problems in neural networks. Since the stochastic or online methods operate on small subsamples of the data and its gradients, they significantly reduce the computational and memory requirements.

1.1. Related Works

Gradient based algorithms are popularly used in training neural network models. These algorithms can be broadly classified into first order and second order methods [1]. Several works have been devoted to stochastic first-order methods such as stochastic gradient descent (SGD) [2,3] and its variance-reduced forms [4**? ?**], AdaGrad [5], RMSprop [6] and Adam [7]. First order methods are popular due to its simplicity and relatively lower complexity. However, incorporating the second order curvature information have shown to improve convergence. But one of the major drawbacks in second order methods is its need for high computational and memory resources. Thus several approximations have been proposed under Newton[8,9] and quasi-Newton[10] methods in order to make use of the second order information while keeping the computational load minimal.

2. Background

$$\min_{\mathbf{w}\in\mathbb{R}^d} E(\mathbf{w}) = \frac{1}{b}\sum_{p\in X} E_p(\mathbf{w}), \tag{1}$$

Training in neural networks is an iterative process in which the parameters are updated in order to minimize an objective function. Given subset of the training dataset $X \subseteq T_r$ with samples $(x_p, d_p)_{p\in X}$ drawn at random from the training set T_r and error function $E_p(\mathbf{w}; x_p, d_p)$ parameterized by a vector $\mathbf{w} \in \mathbb{R}^d$, the objective function is defined as in (1) where $b = |X|$, is the batch size. In full batch, $X = T_r$ and $b = n$ where $n = |T_r|$. In gradient based methods, the objective function $E(\mathbf{w})$ under consideration is minimized by the iterative formula (2) where k is the iteration count and $\mathbf{v}_{k+1}$ is the update vector, which is defined for each gradient algorithm.

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}. \tag{2}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

In the following sections, we briefly discuss the common first and second order gradient based methods.

2.1. Gradient Descent (GD)

The gradient descent (GD) method is one of the earliest and simplest gradient based algorithms. Its update vector v_k is given as

$$v_{k+1} = -\alpha_k \nabla E(w_k) \tag{3}$$

The learning rate α_k determines the step size along the direction of the gradient $\nabla E(w_k)$. The step size α_k is usually set to

$$\alpha_k = \frac{\tau}{\tau + t}\alpha_0 \tag{4}$$

2.2. Nesterov's Accelerated Gradient Descent (NAG)

Nesterov's Accelerated Gradient (NAG) method is a simple modification of GD in which the gradient is computed at $\mathbf{w}_k + \mu\mathbf{v}_k$ instead of $\mathbf{w}_k$ [6]. Thus, the update vector is given by:

$$\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \alpha_k \nabla E(\mathbf{w}_k + \mu\mathbf{v}_k). \tag{5}$$

where $\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$ is the gradient at $\mathbf{w}_k + \mu\mathbf{v}_k$ and is referred to as Nesterov's accelerated gradient vector.

2.3. Adam

Adam is one of the most popular and effective first order methods that uses exponentially decaying average of past squared gradients and past gradients [9].

$$\mathbf{v}_{k+1} = -\alpha \frac{\hat{\mathbf{m}}_\mathbf{k}}{(\sqrt{\hat{\boldsymbol{\theta}}_k} + \epsilon)}, \tag{6}$$

where

$$\hat{\mathbf{m}}_k = \frac{\mathbf{m}_k}{(1-\beta_1^k)}, \qquad \hat{\boldsymbol{\theta}}_k = \frac{\boldsymbol{\theta}_k}{(1-\beta_2^k)}. \tag{7}$$

$\mathbf{m}_k$ and $\boldsymbol{\theta}_k$ given by:

$$\mathbf{m}_k = \beta_1 \mathbf{m}_{k-1} + (1-\beta_1)\nabla E(\mathbf{w}_k), \tag{8}$$

$$\boldsymbol{\theta}_k = \beta_2 \boldsymbol{\theta}_{k-1} + (1-\beta_2)(\nabla E(\mathbf{w}_k))^2. \tag{9}$$

The hyper-parameters $0 \leq \beta_1, \beta_2 < 1$ control the exponential decay rates of these running averages. The running average themselves are estimates of the first (the mean) moment and the second raw (the uncentered variance) moment of the gradient.

Algorithm 1 GD Method

Require: ε and k_{max}
Ensure: $\mathbf{w}_k \in \mathbb{R}^d$.
1: $k \leftarrow 1$
2: **while** $||E(\mathbf{w}_k)|| > \varepsilon$ and $k < k_{max}$ **do**
3: Calculate $\nabla E(\mathbf{w}_k)$
4: $\mathbf{v}_{k+1} \leftarrow -\alpha_k \nabla E(\mathbf{w}_k)$
5: $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + \mathbf{v}_{k+1}$
6: $k \leftarrow k+1$
7: **end while**

Algorithm 2 NAG Method

Require: $0 < \mu < 1$, ε and k_{max}
Ensure: $\mathbf{w}_k \in \mathbb{R}^d$ and $\mathbf{v}_k = 0$.
1: $k \leftarrow 1$
2: **while** $||E(\mathbf{w}_k)|| > \varepsilon$ and $k < k_{max}$ **do**
3: Calculate $\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$
4: $\mathbf{v}_{k+1} \leftarrow \mu\mathbf{v}_k + \alpha_k \nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$
5: $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + \mathbf{v}_{k+1}$
6: $k \leftarrow k+1$
7: **end while**

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

Algorithm 3 BFGS Method

Require: ε and k_{max}
Ensure: $\mathbf{w}_k \in \mathbb{R}^d$ and $\mathbf{H}_k = \mathbf{I}$.
$k \leftarrow 1$
Calculate $\nabla E(\mathbf{w}_k)$
while $||E(\mathbf{w}_k)|| > \varepsilon$ and $k < k_{max}$ **do**
  $\mathbf{g}_k \leftarrow -\mathbf{H}_k^{\text{BFGS}} \nabla E(\mathbf{w}_k)$
  Determine α_k by line search
  $\mathbf{v}_{k+1} \leftarrow \alpha_k \mathbf{g}_k$
  $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + \mathbf{v}_{k+1}$
  Calculate $\nabla E(\mathbf{w}_{k+1})$
  Update $\mathbf{H}_{k+1}^{\text{BFGS}}$ using (11)
  $k \leftarrow k + 1$
end while

Algorithm 4 NAQ Method

Require: $0 < \mu < 1$, ε and k_{max}
Ensure: $\mathbf{w}_k \in \mathbb{R}^d$, $\mathbf{H}_k = \mathbf{I}$ and $\mathbf{v}_k = 0$.
$k \leftarrow 1$
while $||E(\mathbf{w}_k)|| > \varepsilon$ and $k < k_{max}$ **do**
  Calculate $\nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$
  $\hat{\mathbf{g}}_k \leftarrow -\mathbf{H}_k^{\text{NAQ}} \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$
  Determine α_k by line search
  $\mathbf{v}_{k+1} \leftarrow \mu \mathbf{v}_k + \alpha_k \hat{\mathbf{g}}_k$
  $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + \mathbf{v}_{k+1}$
  Calculate $\nabla E(\mathbf{w}_{k+1})$
  Update $\mathbf{H}_k^{\text{NAQ}}$ using (14)
  $k \leftarrow k + 1$
end while

2.4. BFGS quasi-Newton Method

Quasi-Newton methods utilize the gradient of the objective function to achieve superlinear or quadratic convergence. The Broyden-Fletcher-Goldfarb-Shanon (BFGS) algorithm is one of the most popular quasi-Newton methods for unconstrained optimization. The update vector of the quasi-Newton method is given as

$$\mathbf{v}_{k+1} = \alpha_k \mathbf{g}_k, \tag{10}$$

where $\mathbf{g}_k = -\mathbf{H}_k^{\text{BFGS}} \nabla E(\mathbf{w}_k)$ is the search direction. The hessian matrix $\mathbf{H}_k^{\text{BFGS}}$ is symmetric positive definite and is iteratively approximated by the following BFGS formula [11].

$$\mathbf{H}_{k+1}^{\text{BFGS}} = (\mathbf{I} - \mathbf{p}_k \mathbf{q}_k^{\mathrm{T}} / \mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k) \mathbf{H}_k^{\text{BFGS}} (\mathbf{I} - \mathbf{q}_k \mathbf{p}_k^{\mathrm{T}} / \mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k) + \mathbf{p}_k \mathbf{p}_k^{\mathrm{T}} / \mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k, \tag{11}$$

where $\mathbf{I}$ denotes identity matrix,

$$\mathbf{p}_k = \mathbf{w}_{k+1} - \mathbf{w}_k \text{ and } \mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k). \tag{12}$$

The BFGS quasi-Newton algorithm is shown in Algorithm 1. Limited Memory BFGS (LBFGS) is a variant of the BFGS quasi-Newton method, designed for solving large-scale optimization problems. As the scale of the neural network model increases, the $O(d^2)$ cost of storing and updating the Hessian matrix $\mathbf{H}_k^{\text{BFGS}}$ is expensive [12]. In the limited memory version, the Hessian matrix is defined by applying m BFGS updates using only the last m_L curvature pairs $\{\mathbf{p}_k, \mathbf{q}_k\}$. As a result, the computational cost is significantly reduced and the storage cost is down to $O(m_L d)$ where d is the number of parameters and m_L is the memory size.

2.5. Nesterov's Accelerated Quasi-Newton Method

The Nesterov's Accelerated Quasi-Newton (NAQ) [13] method achieves faster convergence compared to the standard BFGS method by quadratic approximation of the objective function at $\mathbf{w}_k + \mu \mathbf{v}_k$ and by incorporating the Nesterov's accelerated gradient $\nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$ in its Hessian update. The update vector of NAQ can be written as:

$$\mathbf{v}_{k+1} = \mu \mathbf{v}_k + \alpha_k \hat{\mathbf{g}}_k, \tag{13}$$

where $\hat{\mathbf{g}}_k = -\mathbf{H}_k^{\text{NAQ}} \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$ is the search direction and the Hessian update equation is given as

$$\mathbf{H}_{k+1}^{\text{NAQ}} = (\mathbf{I} - \mathbf{p}_k \mathbf{q}_k^{\mathrm{T}} / \mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k) \mathbf{H}_k^{\text{NAQ}} (\mathbf{I} - \mathbf{q}_k \mathbf{p}_k^{\mathrm{T}} / \mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k) + \mathbf{p}_k \mathbf{p}_k^{\mathrm{T}} / \mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k, \tag{14}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

where

$$\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu\mathbf{v}_k) \text{ and } \mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu\mathbf{v}_k). \tag{15}$$

(14) is derived from the secant condition $\mathbf{q}_k = (\mathbf{H}_{k+1}^{\text{NAQ}})^{-1}\mathbf{p}_k$ and the rank-2 updating formula [13]. It is proved that the Hessian matrix $\mathbf{H}_{k+1}^{\text{NAQ}}$ updated by (14) is a positive definite symmetric matrix given $\mathbf{H}_k^{\text{NAQ}}$ is initialized to identity matrix [13]. The NAQ algorithm is given in Algorithm 2. Note that the gradient is computed twice in one iteration. This increases the computational cost compared to the BFGS quasi-Newton method. However, due to acceleration by the momentum and Nesterov's gradient term, NAQ is faster in convergence compared to BFGS. It is shown in [13] that NAQ has similar convergence properties to that of BFGS.

Similar to the LBFGS method, LNAQ [14] is the limited memory variant of NAQ that uses the last m_L curvature pairs $\{\mathbf{p}_k, \mathbf{q}_k\}$. In the limited-memory form note that the curvature pairs that are used incorporate the momemtum and Nesterov's accelerated gradient term, thus accelerating LBFGS. Implementation of LNAQ algorithm can be realized by omitting steps 4 and 9 of Algorithm 2 and determining the search direction $\hat{\mathbf{g}}_k$ using the two-loop recursion [11] shown in Algorithm 3. The last m_L vectors of $\mathbf{p}_k$ and $\mathbf{q}_k$ are stored and used in the direction update.

2.6. Momentum quasi-Newton method (MoQ)

The Momentum quasi-Newton (MoQ) method [15,16] is realized by approximating Nesterov's accelerated gradient vector as a linear combination of the current (k^{th}) and previous ($(k-1)^{\text{th}}$) normal gradients, thus making it possible to calculate only one gradient per iteration. MoQ approximates the error function $E(\mathbf{w})$ by assuming that the function is approximately quadratic in the neighborhood of $\mathbf{w}_k + \mu\mathbf{v}_k$.

$$\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k) \simeq \nabla E(\mathbf{w}_k) + \mu\nabla E(\mathbf{v}_k). \tag{16}$$

Furthermore, since $\mathbf{v}_k = \mathbf{w}_k - \mathbf{w}_{k-1}$, (16) can be rewritten as

$$\begin{aligned} \nabla E(\mathbf{w}_k) + \mu\nabla E(\mathbf{v}_k) &= \nabla E(\mathbf{w}_k) + \mu\nabla E(\mathbf{w}_k - \mathbf{w}_{k-1}) \\ &= (1+\mu)\nabla E(\mathbf{w}_k) - \mu\nabla E(\mathbf{w}_{k-1}). \end{aligned} \tag{17}$$

From (16) and (17), it is confirmed that Nesterov's accelerated gradient can be approximated as an extrapolation of $\nabla E(\mathbf{w}_k)$ and $\nabla E(\mathbf{w}_{k-1})$ with a momentum coefficient μ, that is, a weighted linear combination. Therefore, MoQ can be regarded as a method accelerating QN using the momentum term. The update vector $\mathbf{v}_{k+1}$ of MoQ can be defined by approximating the $\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$ in (9) as

$$\mathbf{v}_{k+1} = \mu\mathbf{v}_k - \alpha_k\mathbf{H}_k^{\text{MoQ}}\{(1+\mu)\nabla E(\mathbf{w}_k) - \mu\nabla E(\mathbf{w}_{k-1})\}. \tag{18}$$

The matrix $\mathbf{H}_k^{\text{MoQ}}$ is updated by,

$$\mathbf{H}_{k+1}^{\text{MoQ}} = (\mathbf{I} - (\mathbf{p}_k\hat{\mathbf{q}}_k^{\text{T}}/\mathbf{p}_k^{\text{T}}\hat{\mathbf{q}}_k))\mathbf{H}_k^{\text{MoQ}}(\mathbf{I} - (\hat{\mathbf{q}}_k\mathbf{p}_k^{\text{T}}/\mathbf{p}_k^{\text{T}}\hat{\mathbf{q}}_k)) + (\mathbf{p}_k\mathbf{p}_k^{\text{T}}/\mathbf{p}_k^{\text{T}}\hat{\mathbf{q}}_k), \tag{19}$$

$$\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu\mathbf{v}_k) = \mathbf{w}_{k+1} - (1+\mu)\mathbf{w}_k + \mu\mathbf{w}_{k-1}, \tag{20}$$

$$\hat{\mathbf{q}}_k = \nabla E(\mathbf{w}_{k+1}) - (1+\mu)\nabla E(\mathbf{w}_k) + \mu\nabla E(\mathbf{w}_{k-1}). \tag{21}$$

3. Proposed Method

Second order quasi-Newton (QN) methods build an approximation of a quadratic model recursively using the curvature information along a generated trajectory. In this section, we first show that the Nesterov's acceleration when applied to QN satisfies the secant condtion and then show the derivation of the proposed Nesterov Accelerated Symmetric Rank-1 Quasi-Newton Method.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

3.1. Nesterov Accelerated Symmetric Rank-1 Quasi-Newton Method

Suppose that $E : R^n \to R$ is continuosly differentiable and that $\mathbf{d} \in \mathbf{R}^n$, then from Taylor series, the quadratic model of the objective function at an iterate $\mathbf{w}_k$ is given as

$$E(\mathbf{w}_k + \mathbf{d}) \approx m_k(\mathbf{d}) \approx E(\mathbf{w}_k) + \nabla E(\mathbf{w}_k)^T \mathbf{d} + \frac{1}{2}\, \mathbf{d}^T \nabla^2 E(\mathbf{w}_k)\mathbf{d} \tag{22}$$

In order to find the minimizer $\mathbf{d}_k$, we equate $\nabla \mathbf{m}_k(\mathbf{d}) = 0$ and thus have

$$\mathbf{d}_k = -\nabla^2 E(\mathbf{w}_k)^{-1} \nabla E(\mathbf{w}_k) = -\mathbf{B}_k^{-1} \nabla E(\mathbf{w}_k) \tag{23}$$

The new iterate $\mathbf{w}_{k+1}$ is given as,

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \mathbf{B}_k^{-1} \nabla E(\mathbf{w}_k), \tag{24}$$

and the quadratic model at the new iterate is given as

$$E(\mathbf{w}_{k+1} + \mathbf{d}) \approx m_{k+1}(\mathbf{d}) \approx E(\mathbf{w}_{k+1}) + \nabla E(\mathbf{w}_{k+1})^T \mathbf{d} + \frac{1}{2}\, \mathbf{d}^T \mathbf{B}_{k+1}\mathbf{d}. \tag{25}$$

where α_k is the step length and $\mathbf{B}_k^{-1} = \mathbf{H}_k$ and its consecutive updates $\mathbf{B}_{k+1}^{-1} = \mathbf{H}_{k+1}$ are symmetric positive definite matrices satisfying the secant condition. The Nesterov's acceleration approximates the quadratic model at $\mathbf{w}_k + \mu_k \mathbf{v}_k$ instead of the iterate at $\mathbf{w}_k$. Here $\mathbf{v}_k = \mathbf{w}_k - \mathbf{w}_{k-1}$ and μ_k is the momentum coefficient in the range $(0, 1)$. Thus we have the new iterate $\mathbf{w}_{k+1}$ given as,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu_k \mathbf{v}_k - \alpha_k \mathbf{B}_k^{-1} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k), \tag{26}$$

$$= \mathbf{w}_k + \mu_k \mathbf{v}_k + \alpha_k \mathbf{d}_k. \tag{27}$$

In order to show that the Nesterov accelerated updates also satisfy the secant condition, we require that the gradient of $\mathbf{m}_{k+1}$ should match the gradient of the objective function at the last two iterates $(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ and $\mathbf{w}_{k+1}$. In other words, we impose the following two requirements on $\mathbf{B}_{k+1}$,

$$\nabla m_{k+1}|_{\mathbf{d}=0} = \nabla E(\mathbf{w}_{k+1} + \mathbf{d})|_{\mathbf{d}=0} = \nabla E(\mathbf{w}_{k+1}) \tag{28}$$

$$\nabla m_{k+1}|_{\mathbf{d}=-\alpha_k \mathbf{d}_k} = \nabla E(\mathbf{w}_{k+1} + \mathbf{d})|_{\mathbf{d}=-\alpha_k \mathbf{d}_k} = \nabla E(\mathbf{w}_{k+1} - \alpha_k \mathbf{d}_k) = \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k) \tag{29}$$

From (25),

$$\nabla m_{k+1}(\mathbf{d}) = \nabla E(\mathbf{w}_{k+1}) + \mathbf{B}_{k+1}\mathbf{d} \tag{30}$$

Substituting $\mathbf{d} = 0$ in (30), the condition in (28) is satisfied. From (29) and substituting $\mathbf{d} = -\alpha_k \mathbf{d}_k$ in (30), we have

$$\nabla E(\mathbf{w}_k + \mu \mathbf{v}_k) = \nabla E(\mathbf{w}_{k+1}) - \alpha_k \mathbf{B}_{k+1}\mathbf{d}_k \tag{31}$$

Substituting for $\alpha_k \mathbf{d}_k$ from (27) in (31), we get

$$\nabla E(\mathbf{w}_k + \mu \mathbf{v}_k) = \nabla E(\mathbf{w}_{k+1}) - \mathbf{B}_{k+1}(\mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k)) \tag{32}$$

On rearranging the terms, we have the secant condition

$$\mathbf{y}_k = \mathbf{B}_{k+1}\mathbf{s}_k \tag{33}$$

where,

$$\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \;\text{ and }\; \mathbf{s}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k \mathbf{v}_k) = \alpha_k \mathbf{d}_k \tag{34}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

We have thus shown that the Nesterov accelerated QN update satisfies the secant condition. The update equation of $\mathbf{B}_{k+1}$ for N-SR1 can be derived similar to that of the classic SR1 update [11]. The secant condition requires that $\mathbf{B}_k$ be updated with a symmetric matrix such that $\mathbf{B}_{k+1}$ is also symmetric and satisfies the secant condition. The update of $\mathbf{B}_{k+1}$ is defined using a symmetric-rank-1 matrix formed by an arbitrary vector $\mathbf{u}\mathbf{u}^T$ is given as

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \sigma \mathbf{u}\mathbf{u}^T \tag{35}$$

where σ and $\mathbf{u}$ are chosen such that they satisfy the secant condtion in (33). Substituting (35) in (33), we get

$$\mathbf{y}_k = \mathbf{B}_k\mathbf{s}_k + (\sigma \mathbf{u}^T\mathbf{s}_k)\mathbf{u} \tag{36}$$

Since $(\sigma \mathbf{u}^T\mathbf{s}_k)$ is a scalar, we can deduce $\mathbf{u}$ a scalar multiple of $\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k$ and thus have

$$(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k) = \sigma\delta^2[\mathbf{s}_k^T(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)](\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k) \tag{37}$$

where

$$\sigma = \text{sign}[\mathbf{s}_k^T(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)] \quad \text{and} \quad \delta = \pm|[\mathbf{s}_k^T(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)]|^{1/2} \tag{38}$$

Thus the proposed Nesterov accelerated symmetric rank-1(N-SR1) update is given as

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)^T}{(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)^T\mathbf{s}_k} \tag{39}$$

By applying the Sherman-Morrison-Woodbury Formula, we can find $\mathbf{B}_{k+1}^{-1} = \mathbf{H}_{k+1}$ as

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\mathbf{s}_k - \mathbf{H}_k\mathbf{y}_k)(\mathbf{s}_k - \mathbf{H}_k\mathbf{y}_k)^T}{(\mathbf{s}_k - \mathbf{H}_k\mathbf{y}_k)^T\mathbf{y}_k} \tag{40}$$

where,

$$\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu_k\mathbf{v}_k) \text{ and } \mathbf{s}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k\mathbf{v}_k) = \alpha_k\mathbf{d}_k \tag{41}$$

Algorithm 5 Proposed Algorithm

while $||\nabla E(w_k)|| > \epsilon$ and $k < k_{\max}$ **do**
  Determine μ_k using (3)
  Compute $\nabla E(w_k + \mu_k\mathbf{v}_k)$
  Find $\mathbf{s}_k$ by CG-Steihaug subproblem solver in ()
  Compute $\eta_k = \frac{E(x_k+\mu_k\mathbf{v}_k) - E(x_k+\mu_k\mathbf{v}_k+\mathbf{s}_k)}{m_k(0) - m_k(\mathbf{s}_k)}$
  if $\eta_k \geq \rho$ **then**
    Set $\mathbf{v}_{k+1} = \mu_k\mathbf{v}_k + \mathbf{s}_k, \mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$
  else
    Set $\mathbf{v}_{k+1} = \mathbf{v}_k, \quad \mathbf{w}_{k+1} = \mathbf{w}_k$, reset μ_k
  end if
  $\Delta_{k+1} = \text{adjustTR}(\Delta_k, \rho_k)$
  Compute $\mathbf{y}_k = \nabla E(x_{k+1}) - \nabla E(x_k + \mu_k\mathbf{v}_k) + \zeta\mathbf{s}_k$
  Update $(\mathbf{S}, \mathbf{Y})$ buffer with $(\mathbf{s_k}, \mathbf{y_k})$ is (46) is satisfied
end while

4. Convergence Analysis

In this section we dicuss the implementation of the proposed Nesterov accelerated Symmetric Rank-1 (N-SR1) algorithm in its limited memory form and show its proof of convergence. As mentioned earlier, the Nesterov's acceleration approximates the quadratic model at $\mathbf{w}_k + \mu_k\mathbf{v}_k$ instead of the iterate at $\mathbf{w}_k$. For ease of representation,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

we write $\mathbf{w}_k + \mu_k \mathbf{v}_k = \hat{\mathbf{w}}_k$. The Hessian approximation in (39) can be expressed in its compact representation form [17] as

$$\mathbf{B}_k = \mathbf{B}_0 + (\mathbf{Y}_k - \mathbf{B}_0\mathbf{S}_k)(\mathbf{L}_k + \mathbf{D}_k + \mathbf{L}_k^T - \mathbf{S}_k^T\mathbf{B}_0\mathbf{S}_k)^{-1}(\mathbf{Y}_k - \mathbf{B}_0\mathbf{S}_k) \tag{42}$$

where,

$$\mathbf{B}_0 = \gamma_k \mathbf{I},$$

$$\mathbf{S}_k = [\mathbf{s}_{k-1}, \mathbf{s}_{k-2}, ..., \mathbf{s}_{k-m-1}]$$

$$\mathbf{Y}_k = [\mathbf{y}_{k-1}, \mathbf{y}_{k-2}, ..., \mathbf{y}_{k-m-1}]$$

$$(\mathbf{L}_k)_{i,j} = \begin{cases} \mathbf{s}_i^T \mathbf{y}_i & \text{if } i > j \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{D}_k = \text{diag}\,[\mathbf{S}_k^T \mathbf{Y}_k] \tag{43}$$

Let $\mathbf{\Omega}$ be the level set such that $\mathbf{\Omega} = \{\mathbf{w} \in \mathbb{R}^d : E(\mathbf{w}) \leq E(\mathbf{w_0})\}$ and $\{\mathbf{s}_k\}$ denote the sequence generated by the explicit trust-region algorithm where Δ_k be the trust-region radius of the successful update step. We choose $\gamma_k = 0$. Since the curvature information pairs $(\mathbf{s}_k, \mathbf{y}_k)$ stored in $\mathbf{S}_k$ and $\mathbf{Y}_k$ satisfy the condition in (46), the matrix $\mathbf{M}_k = (\mathbf{L}_k + \mathbf{D}_k + \mathbf{L}_k^T - \mathbf{S}_k^T\mathbf{B}_0\mathbf{S}_k)$ is invertible and positive semi-definite.

Assumption 1: The sequence of iterates $\mathbf{w}_k$ and $\hat{\mathbf{w}}_k$ remains in the closed and bounded set $\mathbf{\Omega}$ on which the objective function is twice continuously differentiable and has Lipschitz continuous gradient, i.e. there exists a constant $L > 0$ such that

$$||\nabla E(\mathbf{w}_{k+1}) - \nabla E(\hat{\mathbf{w}}_k)|| \leq L||\mathbf{w}_{k+1} - \hat{\mathbf{w}}_k|| \quad \forall\, \mathbf{w}_{k+1},\ \hat{\mathbf{w}}_k \in \mathbb{R}^d \tag{44}$$

Assumption 2: The Hessian matrix is bounded and well-defined, .i.e, there exists constants ρ and M, such that

$$\rho \leq ||\mathbf{B}_k|| \leq M \quad \forall\ k \tag{45}$$

and for each iteration

$$|\mathbf{s}_k^T(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)| \geq \rho\, ||\mathbf{s}_k||\, ||\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k|| \tag{46}$$

Assumption 3: Let $\mathbf{B}_k$ be any $n \times n$ symmetric matrix and $\mathbf{s}_k$ be an optimal solution to the trust region subproblem,

$$\min_{\mathbf{d}}\ m_k(\mathbf{d}) = E(\hat{\mathbf{w}}_k) + \mathbf{d}^T\nabla E(\hat{\mathbf{w}}_k) + \frac{1}{2}\mathbf{d}^T\mathbf{B}_k\mathbf{d}, \tag{47}$$

where $\hat{\mathbf{w}}_k + \mathbf{d}$ lies in the trust region. Then for all $k \geq 0$,

$$\left|\nabla E(\hat{\mathbf{w}}_k)^T\mathbf{s}_k + \frac{1}{2}\mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k\right| \geq \frac{1}{2}||\nabla E(\hat{\mathbf{w}}_k)||\, \min\left\{\Delta_k, \frac{||\nabla E(\hat{\mathbf{w}}_k)||}{||\mathbf{B}_k||}\right\} \tag{48}$$

This assumption ensures that the subproblem solved by trust-region results in a sufficiently optimal solution at every iteration. The proof for this assumption can be shown similar to the trust-region proof by Powell.

Lemma 1: If assumptions A1 to A3 hold, and $\{\mathbf{s}_k\}$ is the sequence of vectors solved by the trust region subproblem, and if the initial γ_k is bounded (i.e., $0 \leq \gamma_k \leq \bar{\gamma}_k$), then the Hessian update given by Algorithm 1 and (35) is bounded.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

Proof: We begin with the proof for the general case [18], where the Hessian is bounded by

$$||\mathbf{B}_k^{(j)}|| \leq \left(1+\frac{1}{\rho}\right)^j \gamma_k + \left[\left(1+\frac{1}{\rho}\right)^j - 1\right] M. \tag{49}$$

The proof for (49) is given by mathematical induction. Let m_L be the limited memory size and $(\mathbf{s}_{k,j}, \mathbf{y}_{k,j})$ be the curvature information pairs at the k^{th} iteration for $j = 1, 2, ..., m_L$. For $j = 0$, we can see that (49) holds true. Let us assume that (49) holds true for some $j > 0$. Thus for $j+1$ we have

$$\mathbf{B}_k^{(j+1)} = \mathbf{B}_k^{(j)} + \frac{\left(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}\right)\left(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}\right)^T}{\left(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}\right)^T \mathbf{s}_{k,j+1}} \tag{50}$$

$$||\mathbf{B}_k^{(j+1)}|| \leq ||\mathbf{B}_k^{(j)}|| + \left|\left| \frac{\left(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}\right)\left(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}\right)^T}{\left(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}\right)^T \mathbf{s}_{k,j+1}} \right|\right| \tag{51}$$

$$\leq ||\mathbf{B}_k^{(j)}|| + \frac{||\left(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}\right)\left(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}\right)^T||}{\rho\, ||\left(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}\right)||\ \ ||\mathbf{s}_{k,j+1}||} \tag{52}$$

$$\leq ||\mathbf{B}_k^{(j)}|| + \frac{||\left(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}\right)||}{\rho\, ||\mathbf{s}_{k,j+1}||} \tag{53}$$

$$\leq ||\mathbf{B}_k^{(j)}|| + \frac{||\mathbf{y}_{k,j+1}||}{\rho\, ||\mathbf{s}_{k,j+1}||} + \frac{||\mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}||}{\rho\, ||\mathbf{s}_{k,j+1}||} \tag{54}$$

$$\leq ||\mathbf{B}_k^{(j)}|| + \frac{||\mathbf{y}_{k,j+1}||}{\rho\, ||\mathbf{s}_{k,j+1}||} + \frac{||\mathbf{B}_k^{(j)}||}{\rho} \tag{55}$$

$$\leq \left(1+\frac{1}{\rho}\right)||\mathbf{B}_k^{(j)}|| + \frac{M}{\rho} \tag{56}$$

$$\leq \left(1+\frac{1}{\rho}\right)\left[\left(1+\frac{1}{\rho}\right)^j \gamma_k + \left[\left(1+\frac{1}{\rho}\right)^j - 1\right] M\right] + \frac{M}{\rho} \tag{57}$$

$$||\mathbf{B}_k^{(j+1)}|| \leq \left(1+\frac{1}{\rho}\right)^{j+1} \gamma_k + \left[\left(1+\frac{1}{\rho}\right)^{j+1} - 1\right] M \tag{58}$$

Since we use the limited memory scheme, $\mathbf{B}_{k+1} = \mathbf{B}_k^{(m_L)}$, where m_L is the limited memory size. Therefore, the Hessian approximation at the k^{th} iteration satisfies

$$||\mathbf{B}_{k+1}|| \leq \left(1+\frac{1}{\rho}\right)^{m_L} \gamma_k + \left[\left(1+\frac{1}{\rho}\right)^{m_L} - 1\right] M \tag{59}$$

We choose $\gamma_k = 0$ and thus the Hessian approximation at the k^{th} iteration satisfies (60) and is still bounded.

$$||\mathbf{B}_{k+1}|| \leq \left[\left(1+\frac{1}{\rho}\right)^{m_L} - 1\right] M \tag{60}$$

This completes the inductive proof.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

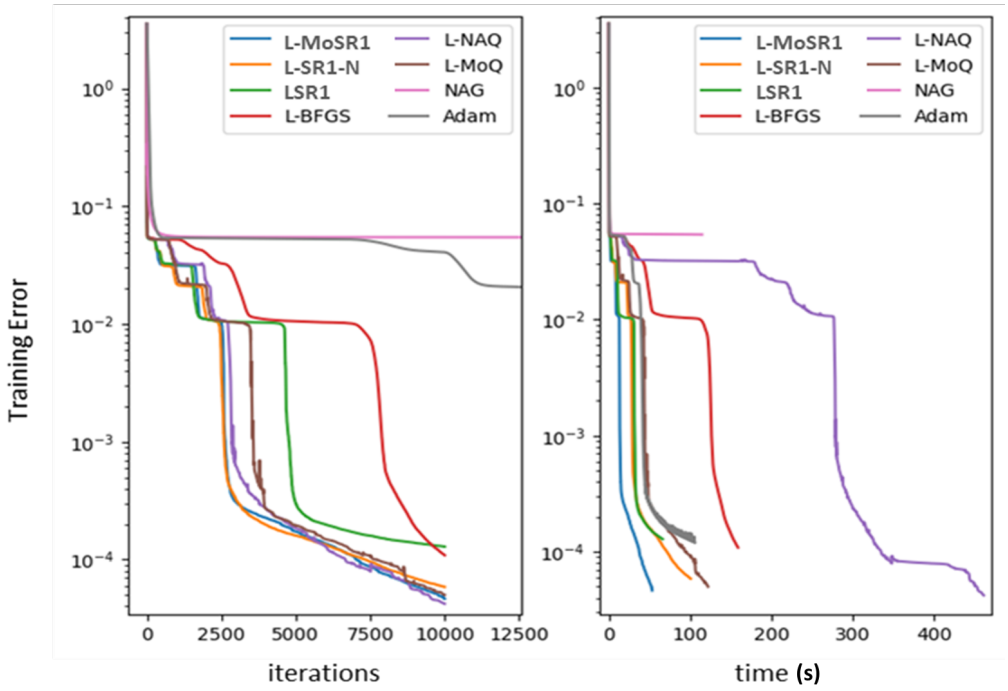

Figure 1. Average results on levy function approximation problem (full batch)

Theorem 1: If the sequence $\{\mathbf{w}_k\}$ is generated by Algorithm 1 and Assumptions (A1) to (A3) hold,

$$\lim_{k\to\infty} ||\nabla E(\mathbf{w}_k)|| = 0. \tag{61}$$

Proof : From the derivation of the proposed N-SR1 algorithm, it is shown that the Nesterov's acceleration to quasi-Newton method satisfies the secant condition. The proposed algorithm ensures the definiteness of the Hessian update as the curvature pairs used in the Hessian update satisfies (46) for all k. The sequence of updates are generated by solving using the trust region method where $\mathbf{s}_k$ is the optimal solution to the subproblem in (47). From Theorem 2.2 in [19], it can be shown that the updates made by the trust region method converges to a stationary point. Since $\mathbf{B_k}$ is shown to be bounded (Lemma 1), it follows from that theorem that as $k \to \infty$, $\mathbf{w}_k$ converges to a point such that $||\nabla E(\mathbf{w}_k)|| = 0$.

5. Simulation Results

We evaluate the performance of the proposed Nesterov accelerated symmetric rank-1 quasi-Newton (L-SR1-N) method in its limited memory form in comparison to conventional first order methods and second order methods. Note that the proposed L-SR1-N has two gradient computations per iteration. The Nesterov's gradient $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v_k})$ can be approximated [15,16] as

$$\nabla E(\mathbf{w}_k + \mu_k \mathbf{v_k}) \approx (1 + \mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1}) \tag{62}$$

We also compare the performance using the above approximation in the proposed L-SR1-N as L-MoSR1.

5.1. Results on Levy function approximation

Consider the following function approximation problem with $\{x_1, x_2, ..., x_5\}$ to be modeled by a neural network.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

$$f(x_1 \dots x_n) = \frac{\pi}{n}\Big\{ \sum_{i=1}^{n-1} [(x_i - 1)^2(1 + 10\sin^2(\pi x_{i+1}))] + 10\sin^2(\pi x_1) + (x_n - 1)^2 \Big\}, x_i \in [-4, 4], \forall i. \quad (63)$$

The performance of the proposed L-SR1-N and L-MoSR1 is evaluated on the Levy function (63) using a $5 - 50 - 1$ NN with $k_{\max} = 10000$, $\epsilon = 10^{-6}$ and $m = 10$. The number of parameters is $d = 351$. Note that we use full batch for the training in this example and the number of training samples is $n = 5000$. Fig. 1 shows the average results of 30 independent trials. The results confirm that the proposed L-SR1-N and L-MoSR1 have better performance compared to SR1.

6. Discussion

Table 1 shows the summary of the computational and storage cost.

7. Conclusion

In this paper we have introduced a stochastic momentum accelerated quasi-Newton method o(L)MoQ in its full and limited memory forms. The proposed algorithm is shown to be efficient compared to the state of the art algorithms such Adam and classical quasi-Newton methods. From the results presented above, we can conclude that the proposed o(L)MoQ methods performs better than conventional o(L)BFGS methods and on par with o(L)NAQ with a reduced computation cost as a result of only one gradient computation per iteration. From the results, we observe that the per iteration computation time of Adam and oLMoQ are comparable. In the future, the effectiveness of the proposed o(L)MoQ will be studied on larger problems. Also, a detailed study on the effect of the limited memory size m, choice of momentum parameter and learning rate scheme will be studied in future works.

Author Contributions: For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization, X.X. and Y.Y.; methodology, X.X.; software, X.X.; validation, X.X., Y.Y. and Z.Z.; formal analysis, X.X.; investigation, X.X.; resources, X.X.; data curation, X.X.; writing—original draft preparation, X.X.; writing—review and editing, X.X.; visualization, X.X.; supervision, X.X.; project administration, X.X.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.", please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Data Availability Statement: Source code will be made available on : https://github.com/indra-ipd/sr1-n

Conflicts of Interest: The authors declare no conflict of interest

Appendix A

Appendix A.1

The appendix is an optional section that can contain details and data supplemental to the main text—for example, explanations of experimental details that would disrupt the flow of the main text but nonetheless remain crucial to understanding and reproducing the research shown; figures of replicates for experiments of which representative data are shown in the main text can be added here if brief, or as Supplementary Data. Mathematical proofs of results not central to the paper can be added as an appendix.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2021 doi:10.20944/preprints202112.0097.v1

Table 1: Summary of Computational Cost and Storage.

Algorithm	Computational Cost	Storage
LBFGS	$nd + 4md + 2d + \zeta nd$	$2md$
LNAQ	$2nd + 4md + 2d + \zeta nd$	$2md$
LMoQ	$nd + 4md + 2d + \zeta nd$	$(2m + 1)d$
LSR1	$nd + nd + \zeta_{TR} md$	$2md$
L-N-SR1	$2nd + nd + \zeta_{TR} md$	$2md$
L-MoSR1	$nd + nd + \zeta_{TR} md$	$(2m + 1)d$

Table A1. This is a table caption. Tables should be placed in the main text near to the first time they are cited.

Title 1	Title 2	Title 3
Entry 1	Data	Data
Entry 2	Data	Data

Appendix B

All appendix sections must be cited in the main text. In the appendices, Figures, Tables, etc. should be labeled, starting with "A"—e.g., Figure A1, Figure A2, etc.

References

1. Haykin, S. *Neural Networks and Learning Machines.*, 3rd ed.; Pearson Prentice Hall,, 2009.
2. Bottou, L.; Cun, Y.L. Large scale online learning. Advances in neural information processing systems, 2004, pp. 217–224.
3. Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*; Springer, 2010; pp. 177–186.
4. Robbins, H.; Monro, S. A stochastic approximation method. *The annals of mathematical statistics* **1951**, pp. 400–407.
5. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* **2011**, *12*, 2121–2159.
6. Tieleman, T.; Hinton, G. Lecture 6.5-RMSProp, COURSERA: Neural networks for machine learning. *University of Toronto, Technical Report* **2012**.
7. Kingma, D.P.; Ba, J. Adam : A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**.
8. Martens, J. Deep learning via Hessian-free optimization. ICML, 2010, Vol. 27, pp. 735–742.
9. Roosta-Khorasani, F.; Mahoney, M.W. Sub-sampled Newton methods I: globally convergent algorithms. *arXiv preprint arXiv:1601.04737* **2016**.
10. Dennis, Jr, J.E.; Moré, J.J. Quasi-Newton methods, motivation and theory. *SIAM review* **1977**, *19*, 46–89.
11. Nocedal, J.; Wright, S.J. *Numerical Optimization*; Springer Series in Operations Research. Springer, second edition, 2006.
12. Schraudolph, N.N.; Yu, J.; Günter, S.; . A stochastic quasi-Newton method for online convex optimization. Artificial Intelligence and Statistics, 2007, pp. 436–443.
13. Ninomiya, H. A novel quasi-Newton-based optimization for neural network training incorporating Nesterov's accelerated gradient. *Nonlinear Theory and Its Applications, IEICE* **2017**, *8*, 289–301.
14. Mahboubi, S.; Ninomiya, H. A Novel Training Algorithm based on Limited-Memory quasi-Newton method with Nesterov's Accelerated Gradient in Neural Networks and its Application to Highly-Nonlinear Modeling of Microwave Circuit. *IARIA International Journal on Advances in Software* **2018**, *11*, 323–334.
15. Mahboubi, S.; Indrapriyadarsini, S.; Ninomiya, H.; Asai, H. Momentum Acceleration of Quasi-Newton Training for Neural Networks. Pacific Rim International Conference on Artificial Intelligence. Springer, 2019, pp. 268–281.
16. Mahboubi, S.; Indrapriyadarsini, S.; Ninomiya, H.; Asai, H.; others. Momentum acceleration of quasi-Newton based optimization technique for neural network training. *Nonlinear Theory and Its Applications, IEICE* **2021**, *12*, 554–574.
17. Byrd, R.H.; Nocedal, J.; Schnabel, R.B. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming* **1994**, *63*, 129–156.
18. Lu, X.; Byrd, R.H. A Study of the Limited Memory Sr1 Method in Practice. PhD thesis, University of Colorado at Boulder, USA, 1996.
19. Shultz, G.A.; Schnabel, R.B.; Byrd, R.H. A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. *SIAM Journal on Numerical analysis* **1985**, *22*, 47–67.