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, Hamamatsu 432-8561, Shizuoka, Japan
2 Graduate School of Electrical and Information Engineering, Shonan Institute of Technology, Fujisawa 251-8511, Kanagawa, Japan; 20T2502@sit.shonan-it.ac.jp (S.M.); ninomiya@info.shonan-it.ac.jp (H.N.)
3 Graduate School of Information Sciences, Hiroshima City University, Hiroshima 731-3194, Shizuoka, Japan; kamio@hiroshima-cu.ac.jp
4 Research Institute of Electronics, Shizuoka University, Hamamatsu 432-8561, Shizuoka, Japan; asai.hideki@shizuoka.ac.jp
* Correspondence: s.indrapriyadarsini.17@shizuoka.ac.jp

Abstract: Gradient-based methods are popularly used in training neural networks and can be broadly categorized into first and second order methods. Second order methods have shown to have better convergence compared to first order methods, especially in solving highly nonlinear problems. The BFGS quasi-Newton method is the most commonly studied second order method for neural network training. Recent methods have been shown to speed up the convergence of the BFGS method using the Nesterov's acclerated gradient and momentum terms. The SR1 quasi-Newton method, though less commonly used in training neural networks, is known to have interesting properties and provide good Hessian approximations when used with a trust-region approach. Thus, this paper aims to investigate accelerating the Symmetric Rank-1 (SR1) quasi-Newton method with the Nesterov's gradient for training neural networks, and to briefly discuss its convergence. The performance of the proposed method is evaluated on a function approximation and image classification problem.

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

check for updates

Citation: Indrapriyadarsini, S.; Mahboubi, S.; Ninomiya, H.; Kamio, T.; Asai, H. Accelerating Symmetric Rank-1 Quasi-Newton Method with Nesterov's Gradient for Training Neural Networks. *Algorithms* **2022**, *15*, 6. https://doi.org/10.3390/a15010006

Academic Editor: Stephanie Allassonniere

Received: 30 November 2021
Accepted: 20 December 2021
Published: 24 December 2021

1. Introduction

Neural networks have shown to have great potential in several applications. 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. Thus the optimization algorithm plays an important role in training neural networks. Gradient-based algorithms have been widely used in training neural networks and can be broadly categorized into first order methods (e.g., SGD, Adam) and higher order methods (e.g., 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 suitable for deep learning and neural networks. While several works focus on sophisticated update strategies for improving the performance of the optimization algorithm, several works propose acceleration techniques such as incorporating momentum, Nesterov's acceleration or Anderson's accleration. Furthermore, it has been shown that second-order methods show faster convergence compared to first order methods, even without the acceleration techniques. While most of the second-order quasi-Newton methods used in training neural networks are rank-2 update methods, rank-1 methods are not widely used since they do not perform as well as the rank-2 update methods. In this paper, we investigate if the Nesterov's acceleration can be applied to the rank-1 update methods of the quasi-Newton family to improve performance.

Related Works

First order methods are most commonly used due to their simplicity and low computational complexity. Several works have been devoted to first-order methods such as the gradient descent [1,2] and its variance-reduced forms [3–5], Nesterov's Accelerated Gradient Descent (NAG) [6], AdaGrad [7], RMSprop, [8] and Adam [9]. However, second order methods have shown to have better convergence, with the only drawbacks being high computational and storage costs. Thus, several approximations have been proposed under Newton [10,11] and quasi-Newton [12] methods to efficiently use the second order information while keeping the computational load minimal. Recently there has been a surge of interest in designing efficient second order quasi-Newton variants which are better suited for large scale problems, such as in [13–16] since in addition to better convergence, second order methods are more suitable for parallel and distributed training. It is notable that among the quasi-Newton methods, the Broyden-Fletcher-Goldfarb-Shanon (BFGS) method is most widely studied for training neural networks. The Symmetric Rank-1 (SR1) quasi-Newton method, though less commonly used in training neural networks, is known to have interesting properties and provide good Hessian approximations when used with a trust-region approach [17,18]. Several works in optimization [19–21] have shown SR1 quasi-Newton methods to be efficient. Recent works such as [22,23] have proposed sampled LSR1 (limited memory) quasi-Newton updates for machine learning and describe efficient ways for distributed training implementation. Recent studies such as [24,25] have shown that the BFGS method can be accelerated by using Nesterov's accelerated gradient and momentum terms. In this paper, we explore if the Nesterov's acceleration can be applied to the LSR1 quasi-Newton method as well. We thus propose a new limited memory Nesterov's acclerated symmetric rank-1 method (L-SR1-N) for training neural networks. We show that the performance of the LSR1 quasi-Newton method can be significantly improved using the trust-region approach and Nesterov's acceleration.

2. Background

Training in neural networks is an iterative process in which the parameters are updated in order to minimize an objective function. Given a subset of the training dataset $X \subseteq T_r$ with input-output pair samples $(x_p, o_p)_{p \in X}$ drawn at random from the training set T_r and error function $E_p(\mathbf{w}; x_p, o_p)$ parameterized by a vector $\mathbf{w} \in \mathbb{R}^d$, the objective function to be minimized is defined as

$$E(\mathbf{w}) = \frac{1}{b} \sum_{p \in X} E_p(\mathbf{w}), \qquad (1)$$

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 $k \in \mathbb{N}$ is the iteration count and $\mathbf{v}_{k+1}$ is the update vector, which is defined for each gradient algorithm.

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}, \quad \text{for} \quad k = 1, 2, ..., k_{max} \in \mathbb{N}. \qquad (2)$$

Notations: We briefly define the notations used in this paper. In general, all vectors are denoted by boldface lowercase characters, matrices by boldface uppercase characters and scalars by simple lowercase characters. The scalars, vectors and matrices at each iteration bear the corresponding iteration index k as a subscript. Below is a list of notations used.

- iteration index $k \in \mathbb{N} : k = 1, 2, ..., k_{max} \in \mathbb{N}$
- n is the number of total samples in T_r and is given by $|T_r|$.
- b is the number of samples in the minbatch $X \subset T_r$ and is given by $|X|$.
- d is the number of parameters of the neural network.
- m_L is the limited memory size.
- α_k is the learning rate or step size.
- μ_k is the momentum coefficient, chosen in the range (0,1).
- $E(\mathbf{w})$ is the error evaluated at $\mathbf{w}$.

- $\nabla E(\mathbf{w})$ is the gradient of the error function evaluated at $\mathbf{w}$.

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

2.1. First-Order Gradient Descent and Nesterov's Accelerated Gradient Descent Methods

The gradient descent (GD) method is one of the earliest and simplest gradient based algorithms. The update vector $\mathbf{v}_{k+1}$ is given as

$$\mathbf{v}_{k+1} = -\alpha_k \nabla E(\mathbf{w}_k). \quad (3)$$

The learning rate α_k determines the step size along the direction of the gradient $\nabla E(\mathbf{w}_k)$. The step size α_k is usually fixed or set to a simple decay schedule.

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

$$\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \alpha_k \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k), \quad (4)$$

where $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ is the gradient at $\mathbf{w}_k + \mu_k \mathbf{v}_k$ and is referred to as Nesterov's accelerated gradient. The momentum coefficient μ_k is a hyperparameter chosen in the range (0,1). Several adaptive momentum and restart schemes have also been proposed for the choice of the momentum [26,27]. The algorithms of GD and NAG are as shown in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 GD Method

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

2.2. Second-Order Quasi-Newton Methods

Second order methods such as the Newton's method have better convergence than first order methods. The update vector of second order methods take the form

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

However, computing the inverse of the Hessian matrix $\mathbf{H}_k = \mathbf{B}_k^{-1}$ incurs a high computational cost, especially for large-scale problems. Thus, quasi-Newton methods are widely used where the inverse of the Hessian matrix is approximated iteratively.

2.2.1. BFGS Quasi-Newton Method

The Broyden-Fletcher-Goldfarb-Shanon (BFGS) algorithm is one of the most popular quasi-Newton methods for unconstrained optimization. The update vector of the BFGS quasi-Newton method is given as $\mathbf{v}_{k+1} = \alpha_k \mathbf{g}_k$, where $\mathbf{g}_k = -\mathbf{H}_k^{\mathrm{BFGS}} \nabla E(\mathbf{w}_k)$ is the search direction. The hessian matrix $\mathbf{H}_k^{\mathrm{BFGS}}$ is symmetric positive definite and is iteratively approximated by the following BFGS rank-2 update formula [28].

$$\mathbf{H}_{k+1}^{\mathrm{BFGS}} = \left(\mathbf{I} - \frac{\mathbf{p}_k \mathbf{q}_k^{\mathrm{T}}}{\mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k}\right) \mathbf{H}_k^{\mathrm{BFGS}} \left(\mathbf{I} - \frac{\mathbf{q}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k}\right) + \frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k}, \tag{6}$$

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

$$\mathbf{p}_k = \mathbf{w}_{k+1} - \mathbf{w}_k \text{ and } \mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k). \tag{7}$$

2.2.2. Nesterov's Accelerated Quasi-Newton Method

The Nesterov's Accelerated Quasi-Newton (NAQ) [24] method introduces Nesterov's acceleration to the BFGS quasi-Newton method by approximating the quadratic model of the objective function at $\mathbf{w}_k + \mu_k \mathbf{v}_k$ and by incorporating Nesterov's accelerated gradient $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$ in its Hessian update. The update vector of NAQ can be written as:

$$\mathbf{v}_{k+1} = \mu_k \mathbf{v}_k + \alpha_k \mathbf{g}_k, \tag{8}$$

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

$$\mathbf{H}_{k+1}^{\mathrm{NAQ}} = \left(\mathbf{I} - \frac{\mathbf{p}_k \mathbf{q}_k^{\mathrm{T}}}{\mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k}\right) \mathbf{H}_k^{\mathrm{NAQ}} \left(\mathbf{I} - \frac{\mathbf{q}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k}\right) + \frac{\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}}}{\mathbf{q}_k^{\mathrm{T}} \mathbf{p}_k}, \tag{9}$$

where

$$\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k \mathbf{v}_k) \text{ and } \mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k). \tag{10}$$

(9) is derived from the secant condition $\mathbf{q}_k = (\mathbf{H}_{k+1}^{\mathrm{NAQ}})^{-1} \mathbf{p}_k$ and the rank-2 updating formula [24]. It is proven that the Hessian matrix $\mathbf{H}_{k+1}^{\mathrm{NAQ}}$ updated by (9) is a positive definite symmetric matrix, given $\mathbf{H}_k^{\mathrm{NAQ}}$ is initialized to identity matrix [24]. It is shown in [24] that NAQ has similar convergence properties to that of BFGS.

The algorithms of BFGS and NAQ are as shown in Algorithm 3 and Algorithm 4, respectively. 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. Often, as the scale of the neural network model increases, the $O(d^2)$ cost of storing and updating the Hessian matrices $\mathbf{H}_k^{\mathrm{BFGS}}$ and $\mathbf{H}_k^{\mathrm{NAQ}}$ become expensive. Hence, limited memory variants LBFGS and LNAQ were proposed, and the respective Hessian matrices were updated using only the last m_L curvature information pairs $\{\mathbf{p}_i, \mathbf{q}_i\}_{i=k-1}^{k-m_L-1}$, where m_L is the limited memory size and is chosen such that $m_L \ll b$.

Algorithm 3 BFGS Method

```
Require: $\varepsilon$ and $k_{max}$
Initialize: $\mathbf{w}_k \in \mathbb{R}^d$ and $\mathbf{H}_k = \mathbf{I}$.
1: $k \leftarrow 1$
2: Calculate $\nabla E(\mathbf{w}_k)$
3: while $||E(\mathbf{w}_k)|| > \varepsilon$ and $k < k_{max}$ do
4:     $\mathbf{g}_k \leftarrow -\mathbf{H}_k^{\text{BFGS}} \nabla E(\mathbf{w}_k)$
5:     Determine $\alpha_k$ by line search
6:     $\mathbf{v}_{k+1} \leftarrow \alpha_k \mathbf{g}_k$
7:     $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + \mathbf{v}_{k+1}$
8:     Calculate $\nabla E(\mathbf{w}_{k+1})$
9:     Update $\mathbf{H}_{k+1}^{\text{BFGS}}$ using (6)
10:    $k \leftarrow k + 1$
11: end while
```

Algorithm 4 NAQ Method

```
Require: $0 < \mu_k < 1$, $\varepsilon$ and $k_{max}$
Initialize: $\mathbf{w}_k \in \mathbb{R}^d$, $\mathbf{H}_k = \mathbf{I}$ and $\mathbf{v}_k = 0$.
1: $k \leftarrow 1$
2: while $||E(\mathbf{w}_k)|| > \varepsilon$ and $k < k_{max}$ do
3:     Calculate $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$
4:     $\mathbf{g}_k \leftarrow -\mathbf{H}_k^{\text{NAQ}} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$
5:     Determine $\alpha_k$ by line search
6:     $\mathbf{v}_{k+1} \leftarrow \mu_k \mathbf{v}_k + \alpha_k \mathbf{g}_k$
7:     $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + \mathbf{v}_{k+1}$
8:     Calculate $\nabla E(\mathbf{w}_{k+1})$
9:     Update $\mathbf{H}_k^{\text{NAQ}}$ using (9)
10:    $k \leftarrow k + 1$
11: end while
```

2.2.3. SR1 Quasi-Newton Method

While the BFGS and NAQ methods update the Hessian using rank-2 updates, the Symmetric Rank-1 (SR1) method performs rank-1 updates [28]. The Hessian update of the SR1 method is given as

$$\mathbf{H}_{k+1}^{\text{SR1}} = \mathbf{H}_k^{\text{SR1}} + \frac{(\mathbf{p}_k - \mathbf{H}_k^{\text{SR1}}\mathbf{q}_k)(\mathbf{p}_k - \mathbf{H}_k^{\text{SR1}}\mathbf{q}_k)^T}{(\mathbf{p}_k - \mathbf{H}_k^{\text{SR1}}\mathbf{q}_k)^T\mathbf{q}_k}, \tag{11}$$

where,

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

Unlike the BFGS or NAQ method, the Hessian generated by the SR1 update may not always be positive definite. Also, the denominator can vanish or become zero. Thus, SR1 methods are not popularly used in neural network training. However, SR1 methods are known to converge faster towards the true Hessian than the BFGS method, and have computational advantages for sparse problems [17]. Furthermore, several strategies have been introduced to overcome the drawbacks of the SR1 method, resulting in them performing almost on par with, if not better than, the BFGS method.

Thus, in this paper, we investigate if the performance of the SR1 method can be accelerated using Nesterov's gradient. We propose a new limited memory Nesterov's accelerated symmetric rank-1 (L-SR1-N) method and evaluate its performance in comparison to the conventional limited memory symmetric rank-1 (LSR1) method.

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 condition and then show the derivation of the proposed Nesterov Accelerated Symmetric Rank-1 Quasi-Newton Method.

Nesterov Accelerated Symmetric Rank-1 Quasi-Newton Method

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

$$E(\mathbf{w}_k + \mathbf{d}) \approx m_k(\mathbf{d}) \approx E(\mathbf{w}_k) + \nabla E(\mathbf{w}_k)^T \mathbf{d} + \frac{1}{2}\, \mathbf{d}^T \nabla^2 E(\mathbf{w}_k)\mathbf{d}. \tag{13}$$

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

$$\mathbf{d}_k = -\nabla^2 E(\mathbf{w}_k)^{-1} \nabla E(\mathbf{w}_k) = -\mathbf{B}_k^{-1} \nabla E(\mathbf{w}_k)\,. \tag{14}$$

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

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \mathbf{B}_k^{-1} \nabla E(\mathbf{w}_k), \tag{15}$$

and the quadratic model at the new iterate is given as

$$E(\mathbf{w}_{k+1} + \mathbf{d}) \approx m_{k+1}(\mathbf{d}) \approx E(\mathbf{w}_{k+1}) + \nabla E(\mathbf{w}_{k+1})^T \mathbf{d} + \frac{1}{2}\, \mathbf{d}^T \mathbf{B}_{k+1} \mathbf{d}\,, \tag{16}$$

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,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu_k \mathbf{v}_k - \alpha_k \mathbf{B}_k^{-1} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k), \tag{17}$$

$$= \mathbf{w}_k + \mu_k \mathbf{v}_k + \alpha_k \mathbf{d}_k. \tag{18}$$

In order to show that the Nesterov accelerated updates also satisfy the secant condition, we require that the gradient of 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 $\mathbf{B}_{k+1}$,

$$\nabla m_{k+1}|_{\mathbf{d}=0} = \nabla E(\mathbf{w}_{k+1} + \mathbf{d})|_{\mathbf{d}=0} = \nabla E(\mathbf{w}_{k+1}), \tag{19}$$

$$\nabla m_{k+1}|_{\mathbf{d}=-\alpha_k \mathbf{d}_k} = \nabla E(\mathbf{w}_{k+1} + \mathbf{d})|_{\mathbf{d}=-\alpha_k \mathbf{d}_k} = \nabla E(\mathbf{w}_{k+1} - \alpha_k \mathbf{d}_k) = \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k). \tag{20}$$

From (16),

$$\nabla m_{k+1}(\mathbf{d}) = \nabla E(\mathbf{w}_{k+1}) + \mathbf{B}_{k+1} \mathbf{d}. \tag{21}$$

Substituting $\mathbf{d} = 0$ in (21), the condition in (19) is satisfied. From (20) and substituting $\mathbf{d} = -\alpha_k \mathbf{d}_k$ in (21), we have

$$\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) = \nabla E(\mathbf{w}_{k+1}) - \alpha_k \mathbf{B}_{k+1} \mathbf{d}_k. \tag{22}$$

Substituting for $\alpha_k \mathbf{d}_k$ from (18) in (22), we get

$$\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) = \nabla E(\mathbf{w}_{k+1}) - \mathbf{B}_{k+1}(\mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k \mathbf{v}_k)). \tag{23}$$

On rearranging the terms, we have the secant condition

$$\mathbf{y}_k = \mathbf{B}_{k+1}\mathbf{s}_k, \tag{24}$$

where,

$$\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \text{ and } \mathbf{s}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k \mathbf{v}_k) = \alpha_k \mathbf{d}_k. \tag{25}$$

We have thus shown that the Nesterov accelerated QN update satisfies the secant condition. The update equation of $\mathbf{B}_{k+1}$ for SR1-N can be derived similarly to that of the classic SR1 update [28]. The secant condition requires $\mathbf{B}_k$ to 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

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \sigma \mathbf{u}\mathbf{u}^T, \tag{26}$$

where σ and $\mathbf{u}$ are chosen such that they satisfy the secant condtion in (24). Substituting (26) in (24), we get

$$\mathbf{y}_k = \mathbf{B}_k \mathbf{s}_k + (\sigma \mathbf{u}^T \mathbf{s}_k)\mathbf{u}. \tag{27}$$

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

$$(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k) = \sigma \delta^2 [\mathbf{s}_k^T (\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)](\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k), \tag{28}$$

where

$$\sigma = \text{sign}[\mathbf{s}_k^T (\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)] \quad \text{and} \quad \delta = \pm |[\mathbf{s}_k^T (\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)]|^{1/2}. \tag{29}$$

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

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T}{(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T \mathbf{s}_k}. \tag{30}$$

Note that the Hessian update is performed only if the below condition in (31) is satisfied, otherwise $\mathbf{B}_{k+1} = \mathbf{B}_k$.

$$|\mathbf{s}_k^T (\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)| \geq \rho \, ||\mathbf{s}_k|| \, ||\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k||. \tag{31}$$

By applying the Sherman-Morrison-Woodbury Formula [28], we can find $\mathbf{B}_{k+1}^{-1} = \mathbf{H}_{k+1}$ as

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)^T}{(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)^T \mathbf{y}_k}, \tag{32}$$

where,

$$\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \text{ and } \mathbf{s}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k \mathbf{v}_k) = \alpha_k \mathbf{d}_k. \tag{33}$$

The proposed algorithm is as shown in Algorithm 5. We implement the proposed method in its limited memory form, where the Hessian is updated using the recent m_L curvature information pairs satisfying (31). Here m_L denotes the limited memory size and is chosen such that $m_L \ll b$. The proposed method uses the trust-region approach where the subproblem is solved using the CG-Steihaug method [28] as shown in Algorithm 6. Also 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 [25,29] as a linear combination of past gradients as shown below.

$$\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \approx (1 + \mu_k)\nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1}). \tag{34}$$

Thus we have the momentum accelerated symmetric rank-1 (L-MoSR1) method by approximating the Nesterov's gradient in L-SR1-N.

Algorithm 5 Proposed Algorithm

```
1:  while $||\nabla E(\mathbf{w}_k)|| > \epsilon$ and $k < k_{\max}$ do
2:      Determine $\mu_k$
3:      Compute $\nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$
4:      Find $\mathbf{s}_k$ by CG-Steihaug subproblem solver in Algorithm (6)
5:      Compute $\rho_k = \frac{E(\mathbf{w}_k+\mu_k\mathbf{v}_k)-E(\mathbf{w}_k+\mu_k\mathbf{v}_k+\mathbf{s}_k)}{m_k(0)-m_k(\mathbf{s}_k)}$
6:      if $\rho_k \geq \eta$ then
7:          Set $\mathbf{v}_{k+1} = \mu_k\mathbf{v}_k + \mathbf{s}_k, \mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$
8:      else
9:          Set $\mathbf{v}_{k+1} = \mathbf{v}_k, \quad \mathbf{w}_{k+1} = \mathbf{w}_k$, reset $\mu_k$
10:     end if
11:     $\Delta_{k+1} = \text{adjustTR}(\Delta_k, \rho_k)$
12:     Compute $\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu_k\mathbf{v}_k) + \zeta\mathbf{s}_k$
13:     Update $(\mathbf{S}_k, \mathbf{Y}_k)$ buffer with $(\mathbf{s}_k, \mathbf{y}_k)$ if (31) is satisfied
14: end while
```

Algorithm 6 CG-Steihaug

```
Require: Gradient $\nabla E(\mathbf{w}_k + \mu_k\mathbf{v}_k)$, tolerance $\epsilon_k > 0$, and trust-region radius $\Delta_k$.
Initialize: Set $\mathbf{z}_0 = 0, \ \mathbf{r}_0 = \nabla E(\mathbf{w}_k + \mu_k\mathbf{v}_k), \ \mathbf{d}_0 = -\mathbf{r}_0 = -\nabla E(\mathbf{w}_k + \mu_k\mathbf{v}_k)$
1:  if then$||\mathbf{r}_0|| < \epsilon_k$
2:      return $\mathbf{s}_k = \mathbf{z}_0 = 0$
3:  end if
4:  for $i = 0, 1, 2, ...$ do
5:      if $\mathbf{d}_i^T\mathbf{B}_k\mathbf{d}_i \leq 0$ then
6:          Find $\tau$ such that $\mathbf{s}_k = \mathbf{z}_i + \tau\mathbf{d}_i$ minimizes (41) and satisfies $||\mathbf{s}_k|| = \Delta_k$
7:          return $\mathbf{s}_k$
8:      end if
9:      Set $\alpha_i = \frac{\mathbf{r}_i^T\mathbf{r}_i}{\mathbf{d}_i^T\mathbf{B}_k\mathbf{d}_i}$
10:     Set $\mathbf{z}_{i+1} = \mathbf{z}_i + \alpha_i\mathbf{d}_i$
11:     if $||\mathbf{z}_{i+1}|| \geq \Delta_k$ then
12:         Find $\tau \geq 0$ such that $\mathbf{s}_k = \mathbf{z}_i + \tau\mathbf{d}_i$ satisfies $||\mathbf{s}_k|| = \Delta_k$
13:         return $\mathbf{s}_k$
14:     end if
15:     Set $\mathbf{r}_{i+1} = \mathbf{r}_i + \alpha_i\mathbf{B}_k\mathbf{d}_i$
16:     if $||\mathbf{r}_{i+1}|| < \epsilon_k$ then
17:         return $\mathbf{s}_k = \mathbf{z}_{i+1}$
18:     end if
19:     Set $\beta_{i+1} = \frac{\mathbf{r}_i^T\mathbf{r}_{i+1}}{\mathbf{r}_i^T\mathbf{r}_i}$
20:     Set $\mathbf{d}_{i+1} = -\mathbf{r}_{i+1} + \beta_{i+1}\mathbf{d}_i$
21: end for
```

4. Convergence Analysis

In this section we discuss the convergence proof of the proposed Nesterov accelerated Symmetric Rank-1 (L-SR1-N) algorithm in its limited memory form. 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, we write $\mathbf{w}_k + \mu_k\mathbf{v}_k = \hat{\mathbf{w}}_k \ \forall \ k = 1, 2, ..., k_{max} \in \mathbb{N}$. In the limited memory scheme, the Hessian matrix can be implicitly constructed using the recent m_L number of curvature information pairs $\{\mathbf{s}_i, \mathbf{y}_i\}_{i=k-1}^{k-m_L-1}$. At a given iteration k, we define matrices $\mathbf{S}_k$ and $\mathbf{Y}_k$ of dimensions $d \times m_L$ as

$$\mathbf{S}_k = [\mathbf{s}_{k-1}, \mathbf{s}_{k-2}, ..., \mathbf{s}_{k-m_L-1}] \quad \text{and} \quad \mathbf{Y}_k = [\mathbf{y}_{k-1}, \mathbf{y}_{k-2}, ..., \mathbf{y}_{k-m_L-1}], \tag{35}$$

where the curvature pairs $\{\mathbf{s}_i, \mathbf{y}_i\}_{i=k-1}^{k-m_L-1}$ are each vectors of dimensions $d \times 1$. The Hessian approximation in (30) can be expressed in its compact representation form [30] as

$$\mathbf{B}_k = \mathbf{B}_0 + (\mathbf{Y}_k - \mathbf{B}_0\mathbf{S}_k)(\mathbf{L}_k + \mathbf{D}_k + \mathbf{L}_k^T - \mathbf{S}_k^T\mathbf{B}_0\mathbf{S}_k)^{-1}(\mathbf{Y}_k - \mathbf{B}_0\mathbf{S}_k), \tag{36}$$

where $\mathbf{B}_0$ is the initial $d \times d$ Hessian matrix, $\mathbf{L}_k$ is a $m_L \times m_L$ lower triangular matrix and $\mathbf{D}_k$ is a $m_L \times m_L$ diagonal matrix as given below,

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

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

$$\mathbf{D}_k = \text{diag}\,[\mathbf{S}_k^T\mathbf{Y}_k]. \tag{37}$$

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 $\{\mathbf{s}_k\}$ $\forall\ k = 1, 2, ..., k_{max} \in \mathbb{N}$, 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 $(\mathbf{s}_k, \mathbf{y}_k)$ given by (33) are stored in $\mathbf{S}_k$ and $\mathbf{Y}_k$ only if they satisfy the condition in (31), 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 $\hat{\mathbf{w}}_k$ $\forall\ k = 1, 2, ..., k_{max} \in \mathbb{N}$ 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*

$$||\nabla E(\mathbf{w}_{k+1}) - \nabla E(\hat{\mathbf{w}}_k)|| \leq L||\mathbf{w}_{k+1} - \hat{\mathbf{w}}_k|| \quad \forall\ \mathbf{w}_{k+1},\ \hat{\mathbf{w}}_k \in \mathbb{R}^d. \tag{38}$$

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

$$\rho \leq ||\mathbf{B}_k|| \leq M \quad \forall\ k = 1, 2, ..., k_{max} \in \mathbb{N}. \tag{39}$$

and for each iteration k

$$|\mathbf{s}_k^T(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)| \geq \rho\ ||\mathbf{s}_k||\ ||\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k||. \tag{40}$$

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,*

$$\min_{\mathbf{d}}\ m_k(\mathbf{d}) = E(\hat{\mathbf{w}}_k) + \mathbf{d}^T\nabla E(\hat{\mathbf{w}}_k) + \frac{1}{2}\mathbf{d}^T\mathbf{B}_k\mathbf{d}, \tag{41}$$

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

$$\left|\nabla E(\hat{\mathbf{w}}_k)^T\mathbf{s}_k + \frac{1}{2}\mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k\right| \geq \frac{1}{2}||\nabla E(\hat{\mathbf{w}}_k)||\ \min\left\{\Delta_k, \frac{||\nabla E(\hat{\mathbf{w}}_k)||}{||\mathbf{B}_k||}\right\}. \tag{42}$$

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* $\mathbf{s}_k$ *be an optimal solution to the trust region subproblem given in (41), and if the initial* γ_k *is bounded (i.e.,* $0 \leq \gamma_k \leq \bar{\gamma}_k$*), then for all* $k \geq 0$*, the Hessian update given by Algorithm 5 and (26) is bounded.*

Proof. We begin with the proof for the general case [31], where the Hessian is bounded by

$$||\mathbf{B}_k^{(j)}|| \leq \left(1+\frac{1}{\rho}\right)^j \gamma_k + \left[\left(1+\frac{1}{\rho}\right)^j - 1\right] M. \tag{43}$$

The proof for (43) 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 given by (33) at the kth iteration for $j = 1, 2, ..., m_L$. For $j = 0$, we can see that (43) holds true. Let us assume that (43) holds true for some $j > 0$. Thus for $j + 1$ we have

$$\mathbf{B}_k^{(j+1)} = \mathbf{B}_k^{(j)} + \frac{(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1})(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1})^T}{(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1})^T \mathbf{s}_{k,j+1}} \tag{44}$$

$$||\mathbf{B}_k^{(j+1)}|| \leq ||\mathbf{B}_k^{(j)}|| + \left|\left|\frac{(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1})(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1})^T}{(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1})^T \mathbf{s}_{k,j+1}}\right|\right| \tag{45}$$

$$\leq ||\mathbf{B}_k^{(j)}|| + \frac{||(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1})(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1})^T||}{\rho\, ||(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1})||\ \ ||\mathbf{s}_{k,j+1}||} \tag{46}$$

$$\leq ||\mathbf{B}_k^{(j)}|| + \frac{||(\mathbf{y}_{k,j+1} - \mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1})||}{\rho\, ||\mathbf{s}_{k,j+1}||} \tag{47}$$

$$\leq ||\mathbf{B}_k^{(j)}|| + \frac{||\mathbf{y}_{k,j+1}||}{\rho\, ||\mathbf{s}_{k,j+1}||} + \frac{||\mathbf{B}_k^{(j)}\mathbf{s}_{k,j+1}||}{\rho\, ||\mathbf{s}_{k,j+1}||} \tag{48}$$

$$\leq ||\mathbf{B}_k^{(j)}|| + \frac{||\mathbf{y}_{k,j+1}||}{\rho\, ||\mathbf{s}_{k,j+1}||} + \frac{||\mathbf{B}_k^{(j)}||}{\rho} \tag{49}$$

$$\leq \left(1+\frac{1}{\rho}\right)||\mathbf{B}_k^{(j)}|| + \frac{M}{\rho} \tag{50}$$

$$\leq \left(1+\frac{1}{\rho}\right)\left[\left(1+\frac{1}{\rho}\right)^j \gamma_k + \left[\left(1+\frac{1}{\rho}\right)^j - 1\right] M\right] + \frac{M}{\rho} \tag{51}$$

$$||\mathbf{B}_k^{(j+1)}|| \leq \left(1+\frac{1}{\rho}\right)^{j+1} \gamma_k + \left[\left(1+\frac{1}{\rho}\right)^{j+1} - 1\right] M \tag{52}$$

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

$$||\mathbf{B}_{k+1}|| \leq \left(1+\frac{1}{\rho}\right)^{m_L} \gamma_k + \left[\left(1+\frac{1}{\rho}\right)^{m_L} - 1\right] M \tag{53}$$

We choose $\gamma_k = 0$ as it removes the choice of the hyperparameter for the initial Hessian $\mathbf{B}_k^{(0)} = \gamma_k \mathbf{I}$ and also ensures that the subproblem solver CG algorithm (Algorithm 6) terminates in at most m_L iterations [22]. Thus the Hessian approximation at the kth iteration satisfies (54) and is still bounded.

$$||\mathbf{B}_{k+1}|| \leq \left[\left(1+\frac{1}{\rho}\right)^{m_L} - 1 \right] M \tag{54}$$

This completes the inductive proof. □

Theorem 1. *Given a level set* $\mathbf{\Omega} = \{\mathbf{w} \in \mathbb{R}^d : E(\mathbf{w}) \leq E(\mathbf{w_0})\}$ *that is bounded, let* $\{\mathbf{w}_k\}$ *be the sequence of iterates generated by Algorithm 5. If assumptions (A1) to (A3) holds true, then we have,*

$$\lim_{k\to\infty} ||\nabla E(\mathbf{w}_k)|| = 0. \tag{55}$$

Proof. From the derivation of the proposed L-SR1-N 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 (31) 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 (41). From Theorem 2.2 in [32], 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. We illustrate the performances in both full batch and stochastic/mini-batch setting. The hyperparameters are set to their default values. The momentum coefficient μ_k is set to 0.9 in NAG and 0.85 in oLNAQ [33]. For L-NAQ [34], L-MoQ [35], and the proposed methods, the momentum coefficient μ_k is set adaptively. The adaptive μ_k is obtained from the following equations, where $\theta_k = 1$ and $\eta = 10^{-6}$.

$$\mu_k = \theta_k(1-\theta_k)/(\theta_k^2 + \theta_{k+1}), \tag{56}$$

$$\theta_{k+1}^2 = (1-\theta_{k+1})\theta_k^2 + \eta\theta_{k+1}. \tag{57}$$

5.1. Results of the Levy Function Approximation Problem

Consider the following Levy function approximation problem to be modeled by a neural network.

$$f(x_1 \dots x_p) = \frac{\pi}{p}\left\{ \sum_{i=1}^{p-1} [(x_i-1)^2(1+10\sin^2(\pi x_{i+1}))] \right.$$
$$\left. + 10\sin^2(\pi x_1) + (x_p-1)^2 \right\}, x_i \in [-4,4], \forall i. \tag{58}$$

The performance of the proposed L-SR1-N and L-MoSR1 is evaluated on the Levy function (58) where $p = 5$. Therefore the inputs to the neural network is $\{x_1, x_2, ..., x_5\}$. We use a single hidden layer with 50 hidden neurons. The neural network architecture is thus $5-50-1$. We terminate the training at $k_{\max} = 10{,}000$, and set $\epsilon = 10^{-6}$ and $m_L = 10$. Sigmoid and linear activation functions are used for the hidden and output layers, respectively. Mean squared error function is used. 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 $n = 5000$. Figure 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 the first order methods as well as the conventional LSR1 and rank-2 LBFGS quasi-Newton method. Furthermore, it can be observed that incorporating the Nesterov's gradient in LSR1 has significantly improved the performance, bringing it almost equivalent to the rank-2 Nesterov accelerated L-NAQ and momentum accelerated L-MoQ methods. Thus we can confirm that the limited memory symmetric rank-1 quasi-Newton method can be significantly accelerated using the Nesterov's gradient. From the iterations vs. training error plot, we can observe that the L-SR1-N and L-MoSR1 are almost similar in performance. This verifies that the approximation applied to L-SR1-N in L-MoSR1 is valid, and has an advantage in terms of computation wall time. This can be observed in the time vs. training error plot, where the L-MoSR1 method converges much faster compared to the other first and second order methods under comparison.

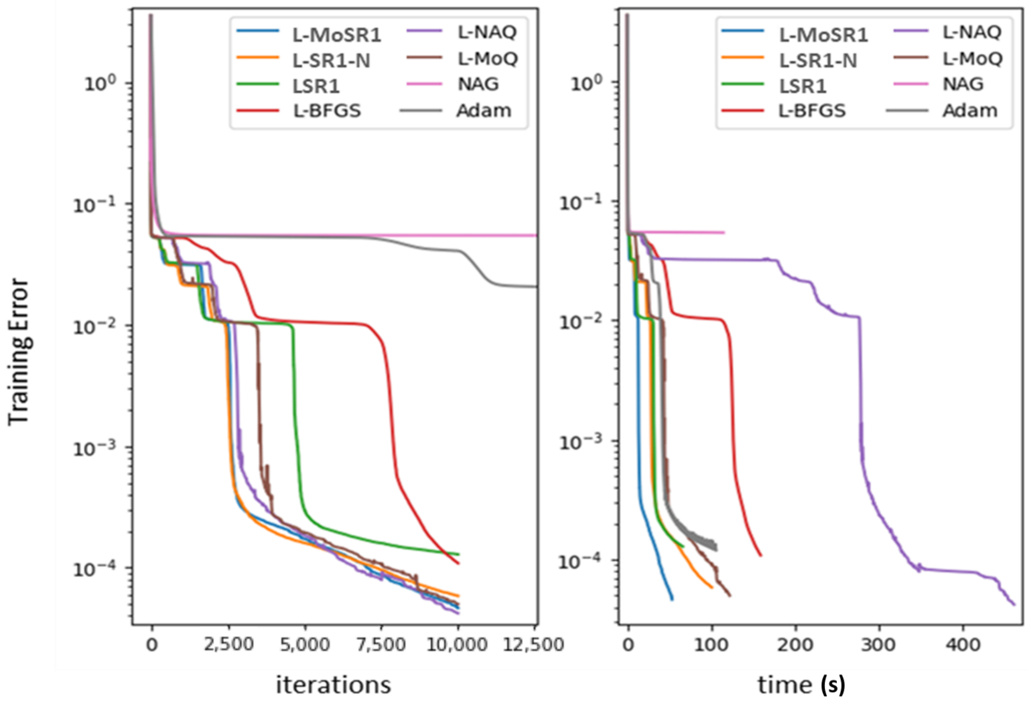

Figure 1. Average results on levy function approximation problem with $m_L = 10$ (full batch).

5.2. Results of MNIST Image Classification Problem

In large scale optimization problems, owing to the massive amount of data and large number of parameters of the neural network model, training the neural network using full batch is not feasible. Hence a stochastic approach is more desirable where the neural networks are trained using a relatively small subset of the training data, thereby significantly reducing the computational and memory requirements. However, getting second order methods to work in a stochastic setting is a challenging task. A common problem in stochastic/mini-batch training is the sampling noise that arises due to the gradients being estimated on different mini-batch samples at each iteration. In this section, we evaluate the performance of the proposed L-SR1-N and L-MoSR1 methods in the stochastic/mini-batch setting. We use the MNIST handwritten digit image classification problem for the evaluation. The MNIST dataset consists of 50,000 train and 10,000 test samples of 28×28 pixel images of handwritten digits from 0 to 9 that needs to be classified. We evaluate the performance of this image classification task on a simple fully connected neural network and LeNet-5 architectures. In a stochastic setting, the conventional LBFGS method is known to be affected by sampling noise and to alleviate this issue, [16] proposed the oLBFGS method that computes two gradients per iteration. We thus compare the performance of our proposed method against both the naive stochastic LBFGS (denoted here as oLBFGS-1) and the oLBFGS proposed in [16].

5.2.1. Results of MNIST on Fully Connected Neural Networks

We first consider a simple fully connected neural network with two hidden layers with 100 and 50 hidden neurons respectively. Thus, the neural network architecture used is $784 - 100 - 50 - 10$. The hidden layers use the ReLU activation function and the loss function used is the softmax cross-entropy loss function. Figure 2 shows the performance comparison with a batch size $b = 128$ and limited memory size of $m_L = 8$. It can be observed that the second order quasi-Newton methods show fast convergence compared to first order methods in the first 500 iterations. From the results we can see that even though the stochastic L-SR1-N (oL-SR1-N) and stochastic MoSR1 (oL-MoSR1) does not perform the best on the small network, it has significantly improved the performance of the stochastic LSR1 (oLSR1) method, and performs better than the oLBFGS-1 method. Since our aim is to investigate the effectiveness of the Nesterov's acceleration on SR1, we focus on the performance comparison of oLBFGS-1, oLSR1 and the proposed oL-SR1-N and oL-MoSR1 methods. As seen from Figure 2, oLBFGS-1, oLSR1 does not further improve the test accuracy or test loss after 1000 iterations. However, incorporating Nesterov's acceleration significantly improved the performance compared to the conventional oL-SR1 and oLBFGS-1, thus confirming the effectiveness of Nesterov's acceleration on LSR1 in the stochastic setting.

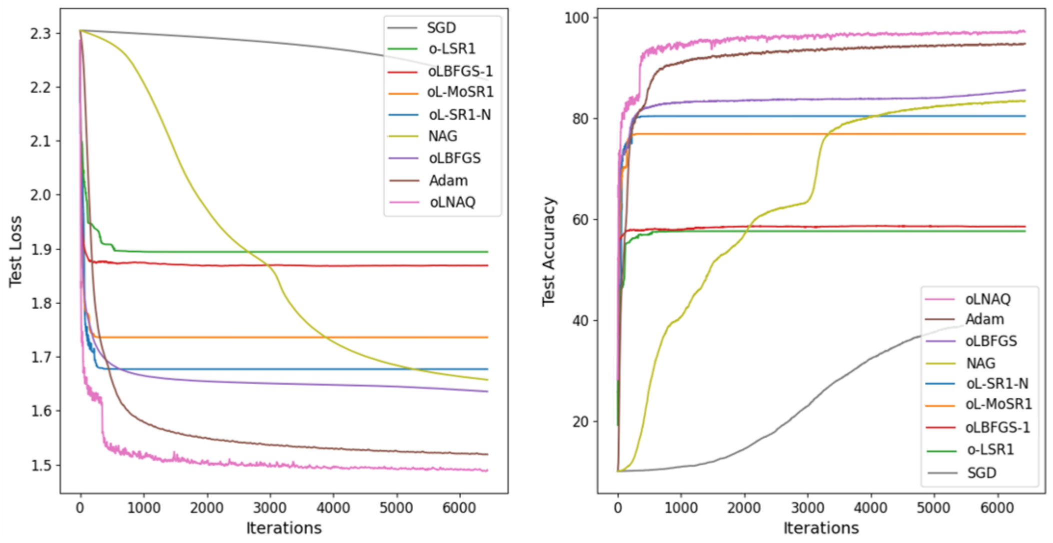

Figure 2. Results of MNIST on fully connected neural network with $b = 128$ and $m_L = 8$.

5.2.2. Results of MNIST on LeNet-5 Architecture

Next, we evaluate the performance of the proposed methods on a bigger network with convolutional layers. The LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier. The number of parameters is $d = 61{,}706$. Figure 3 shows the performance comparison when trained with a batch size of $b = 256$ and limited memory $m_L = 8$. From the results, we can observe that oLNAQ performs the best. However, the proposed oL-SR1-N method performs better compared to both the first order SGD, NAG, Adam and second order oLSR1, oLBFGS-1 and oLBFGS methods. It can be confirmed that incorporating the Nesterov's gradient can accelerate and significantly improve the performance of the conventional LSR1 method, even in the stochastic setting.

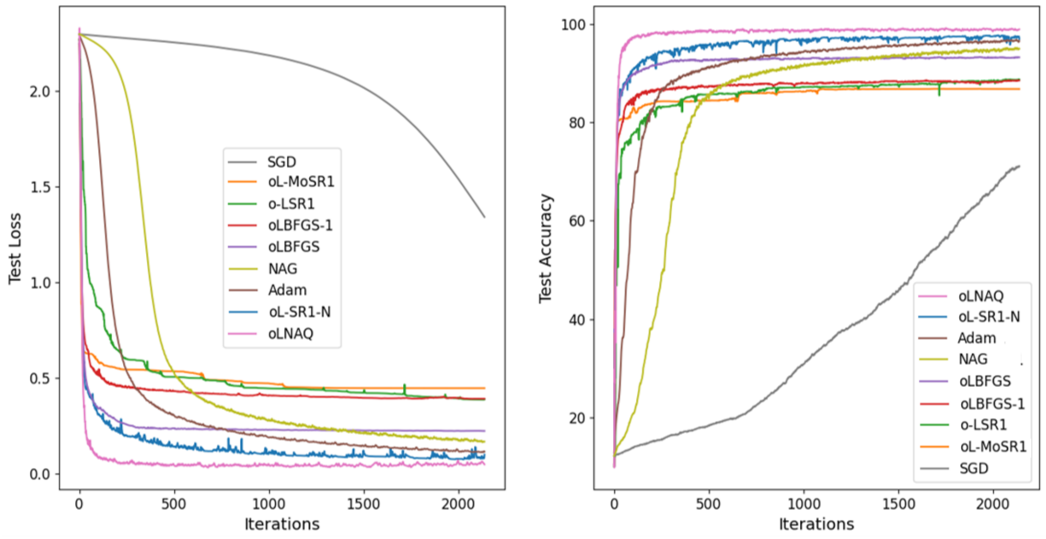

Figure 3. Results of MNIST on LeNet-5 architecture with $b = 256$ and $m_L = 8$.

6. Conclusions and Future Works

Acceleration techniques such as the Nesterov's acceleration have shown to speed up convergence as in the cases of NAG accelerating GD and NAQ accelerating the BFGS methods. Second order methods are said to achieve better convergence compared to first order methods and are more suitable for parallel and distributed implementations. While the BFGS quasi-Newton method is the most extensively studied method in the context of deep learning and neural networks, there are other methods in the quasi-Newton family, such as the Symmetric Rank-1 (SR1), which are shown to be effective in optimization but not extensively studied in the context of neural networks. SR1 methods converge towards the true Hessian faster than BFGS and have computational advantages for sparse or partially separable problems [17]. Thus, investigating acceleration techniques on the SR1 method is significant. The Nesterov's acceleration is shown to accelerate convergence as seen in the case of NAQ, improving the performance of BFGS. We investigate whether the Nesterov's acceleration can improve the performance of other quasi-Newton methods such as SR1 and compare the performance among second-order Nesterov's accelerated variants. To this end, we have introduced a new limited memory Nesterov accelerated symmetric rank-1 (L-SR1-N) method for training neural networks. We compared the results with LNAQ to give a sense of comparison of how the Nesterov's acceleration affects the two methods of the quasi-Newton family, namely BFGS and SR1. The results confirm that the performance of the LSR1 method can be significantly improved in both the full batch and the stochastic settings by introducing Nesterov's accelerated gradient. Furthermore, it can be observed that the proposed L-SR1-N method is competitive with LNAQ and is substantially better than the first order methods and second order LSR1 and LBFGS method. It is shown both theoretically and empirically that the proposed L-SR1-N converges to a stationary point. From the results, it can also be noted that, unlike in the full batch example, the performance of oL-SR1-N and oL-MoSR1 do not correlate well in the stochastic setting. This can be regarded as due to the sampling noise, similar to that of oLBFGS-1 and oLBFGS. In the stochastic setting, the curvature information vector $\mathbf{y}_k$ of oL-MoSR1 is approximated based on the gradients computed on different mini-batch samples. This could introduce sampling noise and hence result in oL-MoSR1 not being a close approximation of the stochastic oL-SR1-N method. Future works could involve solving the sampling noise problem with multi-batch strategies such as in [36], and further improving the performance of L-SR1-N. Furthermore, a detailed study on larger networks and problems with different hyperparameter settings could test the limits of the proposed method.

Author Contributions: Conceptualization, S.I. and S.M.; Methodology, S.I.; Software, S.I.; formal analysis, S.I., S.M. and H.N; validation, S.I., S.M., H.N. and T.K.; writing—original draft preparation, S.I.; writing—review and editing, S.I. and S.M.; resources, H.A.; supervision, H.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Data Availability Statement: Source code is available on: https://github.com/indra-ipd/sr1-n (accessed on 23 December 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bottou, L.; Cun, Y.L. Large scale online learning. *Adv. Neural Inf. Process. Syst.* **2004**, *16*, 217–224.
2. Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 177–186.
3. Robbins, H.; Monro, S. A stochastic approximation method. *Ann. Math. Stat.* **1951**, *22*, 400–407. [CrossRef]
4. Peng, X.; Li, L.; Wang, F.Y. Accelerating minibatch stochastic gradient descent using typicality sampling. *IEEE Trans. Neural Networks Learn. Syst.* **2019**, *31*, 4649–4659. [CrossRef]
5. Johnson, R.; Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 315–323.
6. Nesterov, Y.E. A method for solving the convex programming problem with convergence rate O(1/k^2). *Dokl. Akad. Nauk Sssr* **1983**, *269*, 543–547.
7. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
8. Tieleman, T.; Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural Netw. Mach. Learn.* **2012**, *4*, 26–31.
9. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
10. Martens, J. Deep learning via Hessian-free optimization. *ICML* **2010**, *27*, 735–742.
11. Roosta-Khorasani, F.; Mahoney, M.W. Sub-sampled Newton methods I: Globally convergent algorithms. *arXiv* **2016**, arXiv:1601.04737.
12. Dennis, J.E., Jr.; Moré, J.J. Quasi-Newton methods, motivation and theory. *SIAM Rev.* **1977**, *19*, 46–89. [CrossRef]
13. Mokhtari, A.; Ribeiro, A. RES: Regularized stochastic BFGS algorithm. *IEEE Trans. Signal Process.* **2014**, *62*, 6089–6104. [CrossRef]
14. Mokhtari, A.; Ribeiro, A. Global convergence of online limited memory BFGS. *J. Mach. Learn. Res.* **2015**, *16*, 3151–3181.
15. Byrd, R.H.; Hansen, S.L.; Nocedal, J.; Singer, Y. A stochastic quasi-Newton method for large-scale optimization. *SIAM J. Optim.* **2016**, *26*, 1008–1031. [CrossRef]
16. Schraudolph, N.N.; Yu, J.; Günter, S. A stochastic quasi-Newton method for online convex optimization. *Artif. Intell. Stat.* **2007**, *26*, 436–443.
17. Byrd, R.H.; Khalfan, H.F.; Schnabel, R.B. Analysis of a symmetric rank-one trust region method. *SIAM J. Optim.* **1996**, *6*, 1025–1039. [CrossRef]
18. Brust, J.; Erway, J.B.; Marcia, R.F. On solving L-SR1 trust-region subproblems. *Comput. Optim. Appl.* **2017**, *66*, 245–266. [CrossRef]
19. Spellucci, P. A modified rank one update which converges Q-superlinearly. *Comput. Optim. Appl.* **2001**, *19*, 273–296. [CrossRef]
20. Modarres, F.; Hassan, M.A.; Leong, W.J. A symmetric rank-one method based on extra updating techniques for unconstrained optimization. *Comput. Math. Appl.* **2011**, *62*, 392–400. [CrossRef]
21. Khalfan, H.F.; Byrd, R.H.; Schnabel, R.B. A theoretical and experimental study of the symmetric rank-one update. *SIAM J. Optim.* **1993**, *3*, 1–24. [CrossRef]
22. Jahani, M.; Nazari, M.; Rusakov, S.; Berahas, A.S.; Takáč, M. Scaling up quasi-newton algorithms: Communication efficient distributed sr1. In Proceedings of the International Conference on Machine Learning, Optimization, and Data Science, Siena, Italy, 19–23 July 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 41–54.
23. Berahas, A.; Jahani, M.; Richtarik, P.; Takáč, M. Quasi-Newton methods for machine learning: Forget the past, just sample. *Optim. Methods Softw.* **2021**, *36*, 1–37. [CrossRef]
24. Ninomiya, H. A novel quasi-Newton-based optimization for neural network training incorporating Nesterov's accelerated gradient. *Nonlinear Theory Its Appl. IEICE* **2017**, *8*, 289–301. [CrossRef]
25. Mahboubi, S.; Indrapriyadarsini, S.; Ninomiya, H.; Asai, H. Momentum acceleration of quasi-Newton based optimization technique for neural network training. *Nonlinear Theory Its Appl. IEICE* **2021**, *12*, 554–574. [CrossRef]
26. Sutskever, I.; Martens, J.; Dahl, G.E.; Hinton, G.E. On the importance of initialization and momentum in deep learning. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; Volume 28, pp. 1139–1147.
27. O'donoghue, B.; Candes, E. Adaptive restart for accelerated gradient schemes. *Found. Comput. Math.* **2015**, *15*, 715–732. [CrossRef]

28. Nocedal, J.; Wright, S.J. *Numerical Optimization*, 2nd ed.; Springer Series in Operations Research; Springer: Berlin/Heidelberg, Germany, 2006.
29. Mahboubi, S.; Indrapriyadarsini, S.; Ninomiya, H.; Asai, H. Momentum Acceleration of Quasi-Newton Training for Neural Networks. In *Pacific Rim International Conference on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 268–281.
30. Byrd, R.H.; Nocedal, J.; Schnabel, R.B. Representations of quasi-Newton matrices and their use in limited memory methods. *Math. Program.* **1994**, *63*, 129–156. [CrossRef]
31. Lu, X.; Byrd, R.H. A Study of the Limited Memory Sr1 Method in Practice. Ph.D. Thesis, University of Colorado at Boulder, Boulder, CO, USA, 1996.
32. 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 J. Numer. Anal.* **1985**, *22*, 47–67. [CrossRef]
33. Indrapriyadarsini, S.; Mahboubi, S.; Ninomiya, H.; Asai, H. A Stochastic Quasi-Newton Method with Nesterov's Accelerated Gradient. In *ECML-PKDD*; Springer: Berlin/Heidelberg, Germany, 2019.
34. 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 Int. J. Adv. Softw.* **2018**, *11*, 323–334.
35. Indrapriyadarsini, S.; Mahboubi, S.; Ninomiya, H.; Takeshi, K.; Asai, H. A modified limited memory Nesterov's accelerated quasi-Newton. In Proceedings of the NOLTA Society Conference, IEICE, Online, 6–8 December 2021.
36. Crammer, K.; Kulesza, A.; Dredze, M. Adaptive regularization of weight vectors. *Adv. Neural Inf. Process. Syst.* **2009**, *22*, 414–422. [CrossRef]