A Novel Training Algorithm based on Limited-Memory quasi-Newton Method with Nesterov's Accelerated Gradient for Neural Networks

Shahrzad Mahboubi and Hiroshi Ninomiya
Department of Information Science, Shonan Institute of Technology
Email: mahboubi.shaa@gmail.com, ninomiya@info.shonan-it.ac.jp

***Abstract*—This paper describes a novel training algorithm based on Limited-memory quasi-Newton method (LQN) with Nesterov's accelerated gradient for faster training of neural networks.**

Keywords–Limited-memory quasi-Newton method; Nesterov's accelerated gradient method; neural networks; training algorithm.

I. INTRODUCTION

Neural networks have been recognized as a useful tool for the function approximation problems with high-nonlinearity [1][2]. Training is the most important step in developing a neural network model. Gradient algorithm based on the first order approximation such as Steepest gradient (SG), Momentum and Nesterov's accelerated gradient (NAG) methods are popularity used for this purpose [1]-[3]. With the progress of AI technologies the characteristics between inputs and desired outputs of the training samples become increasingly complex. Then neural networks have to train the highly-nonlinear functions. Under such circumstances the first order methods converge too slowly and optimization error cannot be effectively reduced within finite time in spite of its advantage [2]. The quasi-Newton (QN) training, which is one of the most effective optimization based on the second order approximation [4] is widely utilized as the robust training algorithm for highly-nonlinear function approximation. However, the QN iteration includes the approximated Hessian, that is QN needs the massive computer resources of memories as the scale of neural network becomes larger. To deal with this problem, it is noteworthy that QN incorporating Limited-memory scheme is effective for large-scale problems.

In this paper, the acceleration technique of Limited-memory QN (LQN) is proposed using Nesterov's accelerated gradient. In [2], the QN training was drastically accelerated by Nesterov's accelerated gradient that is called Nesterov's accelerated quasi-Newton (NAQ) training. Therefore, the proposed algorithm can be accelerated in cooperating the similar scheme of NAQ into LQN. The method is referred to as Limited-memory NAQ (LNAQ). The proposed algorithm is demonstrated through the computer simulations for a benchmark problem compared with the conventional training methods.

II. FORMULATION OF TRAINING AND LIMITED-MEMORY QUASI-NEWTON METHOD

This section describes an error function of the neural network training and conventional training algorithms beased on gradients such as Back propagetion and Limited-memory quasi-Newton method.

A. Formulation of training

Let $\mathbf{d}_p$, $\mathbf{o}_p$ and $\mathbf{w} \in \mathbb{R}^D$ be the p-th desired, output, and weight vectors, respectively the error function $E(\mathbf{w})$ is defined as the mean squared error (MSE) of

$$E(\mathbf{w}) = \frac{1}{|T_r|}\sum_{p\in T_r} E_p(\mathbf{w}), \quad E_p(\mathbf{w}) = \frac{1}{2}\|\mathbf{d}_p - \mathbf{o}_p\|^2, \quad (1)$$

where T_r denotes a training data set $\{\mathbf{x}_p, \mathbf{d}_p\}, p \in T_r$ and $|T_r|$ is the number of training samples. Among the gradient-based algorithms, (1) is minimized by $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$, where k is the iteration count and $\mathbf{v}_{k+1}$ is the update vector. SG, so-called Back propagation method has $\mathbf{v}_{k+1} = -\alpha_k \nabla E(\mathbf{w}_k)$ with the step size α_k and the gradient vector at $\mathbf{w}_k$ of $\nabla E(\mathbf{w}_k)$.

B. Limited-memory quasi-Newton training (LQN)

The update vector of QN is defined as

$$\mathbf{v}_{k+1} = -\alpha_k \mathbf{H}_k \nabla E(\mathbf{w}_k). \quad (2)$$

where $\mathbf{H}_k$ is the symmetric positive definite matrix and iteratively approximated by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula [4]. For the purpose of reducing the amount of memory used in QN a sophisticated technique incorporating the limited-memory scheme is widely utilized for the calculation of $\mathbf{v}_{k+1}$ of LQN. Specifically, this method is useful for solving problems whose $\mathbf{H}_k$ (inverse of approximated Hessian) matrices in (2) cannot be computed at a reasonable cost [4]. Furthermore, instead of storing $D \times D$ matrix of $\mathbf{H}_k$, only $2 \times t$ vectors of the dimension D ($2 \times t \times D$) have to be stored. Here, t is defined by users and $t \ll D$.

III. PROPOSED ALGORITHM -LNAQ

Nesterov's accelerated quasi-Newton (NAQ) training was derived by the quadratic approximation of (1) around $\mathbf{w}_k + \mu\mathbf{v}_k$, and μ was the momentum coefficient whereas QN used the approximation of (1) around $\mathbf{w}_k$ [2]. NAQ drastically improved the convergence speed of QN using the gradient vector at $\mathbf{w}_k + \mu\mathbf{v}_k$ of $\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$ called Nesterov's accelerated gradient vector [3]. In this paper, we apply the limited-memory method into NAQ, that is called LNAQ. The proposed method can be expected to cope with large-scale optimization problems like LQN, maintaining the fast training of NAQ. The update vector of NAQ is

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

 ISBN: 978-1-61208-608-8

and the matrix $\hat{\mathbf{H}}_k$ was updated by,

$$\hat{\mathbf{H}}_{k+1} = \hat{\mathbf{H}}_k - \frac{(\hat{\mathbf{H}}_k\mathbf{q}_k)\mathbf{p}_k^{\mathrm{T}} + \mathbf{p}_k(\hat{\mathbf{H}}_k\mathbf{q}_k)^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k} + \left(1 + \frac{\mathbf{q}_k^{\mathrm{T}}\hat{\mathbf{H}}_k\mathbf{q}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}\right)\frac{\mathbf{p}_k\mathbf{p}_k^{\mathrm{T}}}{\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k}, \tag{4}$$

where

$$\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu\mathbf{v}_k), \tag{5}$$

$$\mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu\mathbf{v}_k). \tag{6}$$

(4) was equivalent to the BFGS formula of [4] by replacing $\mathbf{p}_k$ and $\mathbf{q}_k$ into $\mathbf{s}_k = \mathbf{w}_{k+1} - \mathbf{w}_k$ and $\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k)$, respectively. Therefore, the limited-memory scheme can be straightly applied to NAQ as illustrated in Algorithms 1 and 2. In Algorithm 1, two calculations of gradient vectors of $\nabla E(\mathbf{w}_k+\mu\mathbf{v}_k)$ and $\nabla E(\mathbf{w}_{k+1})$ were needed within a training loop whereas LQN was calculated once. This is a disadvantage of LNAQ, but the algorithm can farther shorten the iteration counts to cancel out the effect of this shortcoming.

Algorithm1: The proposed LNAQ
1. $k = 1$;
2. $\mathbf{w}_1 = rand[-0.5, 0.5]$(uniform random numbers);
3. **While**$(E(\mathbf{w}_k) > \varepsilon$ and $k < k_{max})$
   (a) Calculate $\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$;
   (b) Calculate the direction vector $\hat{\mathbf{c}}_k$ using Algorithm 2;
   (c) Calculate α_k using Armijo's condition;
   (d) Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \mu\mathbf{v}_k - \alpha_k\hat{\mathbf{c}}_k$;
   (e) Calculate $\nabla E(\mathbf{w}_{k+1})$;
   (f) $k = k + 1$;
4. **return** $\mathbf{w}_k$;

Algorithm2: Direction Vector of LNAQ
1. $\hat{\mathbf{c}}_k = -\nabla E(\mathbf{w}_k + \mu\mathbf{v}_k)$;
2. for $i : k, k-1, \ldots, k - \min(k, (t-1))$;
   (a) $\hat{\beta}_i = \mathbf{p}_i^{\mathrm{T}}\hat{\mathbf{c}}_k / \mathbf{p}_i^{\mathrm{T}}\mathbf{q}_i$;
   (b) $\hat{\mathbf{c}}_k = \hat{\mathbf{c}}_k - \hat{\beta}_i\mathbf{q}_i$;
3. if $k > 1, \hat{\mathbf{c}}_k = (\mathbf{p}_k^{\mathrm{T}}\mathbf{q}_k / \mathbf{q}_k^{\mathrm{T}}\mathbf{q}_k)\hat{\mathbf{c}}_k$;
4. for $i : k - \min(k, (t-1)), \ldots, k, k-1$;
   (a) $\hat{\tau} = \mathbf{q}_i^{\mathrm{T}}\hat{\mathbf{c}}_k / \mathbf{q}_i^{\mathrm{T}}\mathbf{p}_i$;
   (b) $\hat{\mathbf{c}}_k = \hat{\mathbf{c}}_k - (\hat{\beta}_i - \hat{\tau})\mathbf{p}_i$;
5. **return** $\hat{\mathbf{c}}_k$;

IV. Simulation results

In this paper, we demonstrated the effectiveness of the proposed LNAQ for the training of neural networks. Levy function as shown in (7) is used for the function approximation problem. Levy function is a multimodal function and has the highly-nonlinear characteristic [2].

$$f(x_1 \ldots x_n) = \frac{\pi}{n}\left\{\sum_{i=1}^{n-1}[(x_i - 1)^2(1 + 10\sin^2(\pi x_{i+1}))] + 10\sin^2(\pi x_1) + (x_n - 1)^2\right\}, x_i \in [-4, 4], \forall i. \tag{7}$$

Here the structure of neural network is 5-50-1, that is, the network has 5 inputs, 1 output and 50 hidden neurons. The dimension of $\mathbf{w}$ is 351. The number of training data is $|T_r| = 5000$, which are generated by uniformly random number in $x_i \in [-4, 4]$. The storage amount of t is 30 for LQN and LNAQ. Ten simulation runs are performed from different initial values. The propose LNAQ is compared with NAG, ADAM and LQN. ADAM is one of the latest and the most effective first order training method [5]. The momentum coefficient μ of NAG and LNAQ is experimentally set to 0.95 in the simulations. The termination conditions are set to $E(\mathbf{w}) \leq 10^{-4}$ and $k_{max} = 5\times10^5$. Figure 1 shows the training error $E(\mathbf{w})$ of NAG, ADAM, LQN and LNAQ for the iteration count in the early stage of training. From this figure, it can be seen that NAG cannot converge within 5×10^4 iterations and the iteration counts of LNAQ is drastically improved. Next, the averages of iteration counts and CPU times (sec) for NAG, ADAM, LQN and LNAQ are illustrated in Table 1. In the simulations, the trainings continued until each training error of $E(\mathbf{w})$ was less than 1×10^{-4}. This is an important point of the function approximation problem. That is, the trained network with the small MSE of $E(\mathbf{w})$ can become an accurate neural network model. All algorithms can obtain the neural network model with the small training error. However, the first order methods (NAG and ADAM) need more iteration counts and CPU times than LQN and LNAQ. Furthermore, LNAQ was able to improve the convergence speed compare with LQN. As a result, it is confirmed that the proposed LNAQ is efficient and practical for the training of neural networks.

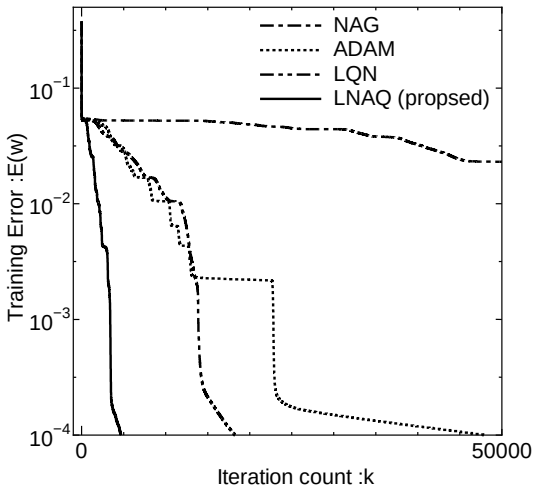

Figure 1. The average training errors for iteration count.

TABLE 1. SIMULATION RESULTS FOR THE AVERAGE ITERATION COUNTS AND CPU TIMES.

Algorithm	NAG	ADAM	LQN	LNAQ
Iteration counts	335,403	48,680	17,892	4,514
CPU times (sec)	1,820	263	142	60

V. Conclusions

In this research, we proposed a novel training algorithm called LNAQ which was developed based on Limited-memory method of QN using Nesterov's accelerated gradients. The effectiveness of the proposed LNAQ was demonstrated through

the computer simulations compared with the conventional algorithms such as NAG, ADAM and LQN. It helps provide accurate neural network models much fast. In the future, the validity of the proposed algorithm for more highly nonlinear function approximation problems and the much huge scale problems including deep networks will be demonstrated.

ACKNOWLEDGMENT

This work was supported by Japan Society for the Promotion of Science (JSPS), KAKENHI (17K00350).

REFERENCES

[1] S. Haykin, Neural Networks and Learning Machines 3^{rd}, Pearson, 2009.

[2] H. Ninomiya, "A novel quasi-Newton-Optimization for neural network training incorporating Nesterov's accelerated gradient", *IEICE NOLTA Journal*, **vol.E**8-**N**, no.4, p.289-301, Oct. 2017.

[3] Y. Nesterov, "*Introductory Lectures on Convex Optimization: A Basic Course*", Kluwer Boston, 2004.

[4] J. Nocedal, and S.J. Wright., *Numerical Optimization Second Edition,* Springer, 2006.

[5] S. Ruder,"An overview of gradient descent optimization algorithms", arXiv 1609.04747, 15. Jun. 2017.

 ISBN: 978-1-61208-608-8