A novel quasi-Newton-based optimization for neural network training incorporating Nesterov’s accelerated gradient

Hiroshi Ninomiya¹a)

¹ Department of Information Science, Faculty of Engineering, Shonan Institute of Technology
1-1-25 Tsujido-Nishikaigan, Fujisawa 251-8511, Japan

a) ninomiya@info.shonan-it.ac.jp

Received January 24, 2017; Revised May 27, 2017; Published October 1, 2017

Abstract: This paper describes a novel quasi-Newton (QN) based accelerated technique for training of neural networks. Recently, Nesterov’s accelerated gradient method has been utilized for the acceleration of the gradient-based training. In this paper the acceleration of the QN training algorithm is realized by the quadratic approximation of the error function incorporating the momentum term as Nesterov’s method. It is shown that the proposed algorithm has a similar convergence property with the QN method. Neural network trainings for the function approximation and the microwave circuit modeling problems are presented to demonstrate the proposed algorithm. The method proposed here drastically improves the convergence speed of the conventional QN algorithm.

Key Words: neural networks, training algorithm, Nesterov’s accelerated gradient, quasi-Newton method

1. Introduction

Neural networks have been recognized as a useful tool for the function approximation problems with high-nonlinearity [1]. For example, the techniques are useful for microwave modeling and design in which neural networks can be trained from Electro-Magnetic (EM) data over a range of geometrical parameters and trained neural networks become models providing fast solutions of the EM behavior it learned [2–4]. This is useful for modeling where formulas are not available or original model is computationally too expensive. A neural model is trained once and can be used again and again. This avoids repetitive EM simulations where a change in the physical dimension requires a re-simulation of the EM structure. However these modelings have strong nonlinearity to itself and need lower training error. Therefore the second-order training methods are more suitable than the first-order algorithms [4].

Training is the most important step in developing a neural network model. Gradient based algorithms are popularly used for the training [1]. For the purpose of this paper gradient-based training algorithms can be divided into two categories: first order methods, so called Steepest gradient (SG),
Momentum (classical momentum, CM) and Nesterov’s accelerated gradient (NAG) methods, and second (or super-linear) order methods, namely (quasi-)Newton methods. The formers are popular methods used to train the neural networks. Especially, CM has been utilized for training the large scale neural networks [1, 7]. CM is a technique for accelerating SG that accumulates an update vector in directions of persistent reduction in the objective across iterations. NAG has been the subject of much recent study in convex optimization theory [8], arguing that NAG can be viewed as a simple modification of CM, and can sometimes provide a distinct improvement in performance for acceleration of neural network training [7]. However, when applied to highly nonlinear function modeling, these first order methods still converge too slowly and optimization error cannot be effectively reduced within finite time in spite of its advantage [5].

On the other hand, quasi-Newton (QN) method which is one of the most efficient optimization techniques [10] is widely utilized as the robust training algorithm for highly nonlinear function approximation [2–6]. Furthermore several improvements for QN have been done [3, 5]. These methods could obtain the stronger local convergent property than QN, over a long simulation time. Although they improved asymptotic convergence rates, the methods were still often slow in practice.

This paper describes the acceleration technique of QN incorporating Nesterov’s accelerated gradient. The proposed algorithm is referred to as Nesterov’s accelerated quasi-Newton (NAQ) method. Hessian of the error function is approximated using both of the normal and Nesterov’s accelerated gradient vectors in NAQ. This technique is realized by the quadratic approximation of the error function incorporating the inertial term. The iteration during the NAQ training can be shortened without loss of the strong ability to search a global minimum in QN. The proposed algorithm is demonstrated through the computer simulations for the benchmark problems such as the function approximation and the microwave circuit modeling with high-nonlinearity.

2. Formulation of training and gradient algorithms

2.1 Formulation of training

Let \( d_p, o_p \), and \( w \) be the \( p \)-th desired, \( p \)-th output, and weight vectors, respectively, the error function \( E(w) \) is defined as

\[
E(w) = \frac{1}{|T_r|} \sum_{p \in T_r} E_p(w), \quad \text{and} \quad E_p(w) = \frac{1}{2} ||d_p - o_p||^2,
\]

where \( T_r \) denotes a training data set \( \{x_p, d_p\}, p \in T_r \) and \( |T_r| \) is the number of training samples. Among the training, (1) is minimized by the following iterative formula:

\[
w_{k+1} = w_k + v_{k+1},
\]

where \( k \) is the iteration count and \( v_{k+1} \) is the update vector which is individually defined in each gradient algorithm. The simple steepest gradient (SG) algorithm that is the original back-propagation has \( v_{k+1} = -\alpha_k \nabla E(w_k) \) with the step size \( \alpha_k \) and \( \nabla E(w_k) \) that is the gradient vector at \( w_k \).

2.2 Momentum and Nesterov’s accelerated gradient methods

Given an error function \( E(w) \) to be minimized, the update vector of the classical momentum (CM) method is given by

\[
v_{k+1} = \mu v_k - \alpha_k \nabla E(w_k),
\]

where \( \mu \in (0, 1) \) is the momentum coefficient. CM is a technique for accelerating the gradient descent that accumulates a previous update vector in directions of persistent reduction in the training [1, 7].

Nesterov’s accelerated gradient (NAG) method has been the subject of much recent attention [8, 9]. The NAG update can be written as:

\[
v_{k+1} = \mu v_k - \alpha_k \nabla E(w_k + \mu v_k).
\]

In analogy with CM, NAG is a first-order optimization method with better convergence rate than gradient descent in certain situations. From (2–4) it is confirmed that both CM and NAG compute
the new update vector by applying a gradient-based correction to the previous one decaying by the
momentum coefficient \( \mu \). But while CM computes the gradient vector from the current position \( w_k \),
NAG first performs a partial update to \( w_k \), computing \( w_k + \mu v_k \), and then computes the gradient at
\( w_k + \mu v_k \). Figures 1 (a) and (b) show diagrams which illustrate the update vectors geometrically.

2.3 quasi-Newton algorithm

The update vector of quasi-Newton (QN) is defined as

\[
v_{k+1} = -\alpha_k H_k \nabla E(w_k),
\]

where \( H_k \) is the symmetric positive definite matrix and iteratively approximated by the following
BFGS (Broyden-Fletcher-Goldfarb-Shanno) formula [10]:

\[
H_{k+1} = H_k - \frac{(H_k y_k) s_k^T + s_k (H_k y_k)^T}{s_k^T y_k} + \left( 1 + \frac{y_k^T H_k y_k}{s_k^T y_k} \right) \frac{s_k s_k^T}{s_k^T y_k},
\]

where \( s_k = w_{k+1} - w_k \) and \( y_k = \nabla E(w_{k+1}) - \nabla E(w_k) \) and T denotes the transpose. QN is one of the
most efficient gradient-based optimization algorithms [10, 11] and commonly-used training method for
highly nonlinear function problems [2–6]. Note that, the secant equation,

\[
y_k = B_{k+1} s_k = (H_k^{-1}) s_k
\]

is guaranteed in QN algorithm. This will be possible only if \( s_k \) and \( y_k \) satisfy the curvature condition
\( s_k^T y_k > 0 \). The diagram of the update vector of QN is illustrated in Fig. 1 (c). From Fig. 1 (c) it is
confirmed that \( v_{k+1} \) is given by the quadratic approximation of \( E(w) \) at \( w_k \) using the displacement
\( s_k \) and the change of gradient \( y_k \). The algorithm of QN is shown in Algorithm 1.

**Algorithm 1: QN**

1. \( k = 1 \)
2. Initialize \( w_1 \) and \( H_1 \) by uniform random numbers and the unit matrix \( I \), respectively;
3. Calculate \( \nabla E(w_1) \);
4. **While** (\( ||\nabla E(w_k)|| > \epsilon \) and \( k < k_{\text{max}} \))
   5. Execute Line search to decide the step size \( \alpha_k \) using Armijo’s condition;
   6. Update \( v_{k+1} = -\alpha_k H_k \nabla E(w_k) \);
   7. Update \( w_{k+1} = w_k + v_{k+1} \);
   8. Calculate \( \nabla E(w_{k+1}) \);
   9. Update \( H_{k+1} \) using (6);
10. \( k = k + 1 \)
11. **EndWhile**

3. Nesterov’s accelerated quasi-Newton training

3.1 Proposed algorithm: Nesterov’s accelerated quasi-Newton method (NAQ)

Several improvements for QN have been done [3, 5]. Although these methods could obtain the stronger
convergent property than that of the conventional QN, the methods are still often slow in practice. In
this paper QN is accelerated by the quadratic approximation of (1) around the position of \( w_k + \mu v_k \)
and Hessian approximation incorporating Nesterov’s accelerated gradient vector of \( \nabla E(w_k + \mu v_k) \)
whereas QN is the second-order approximation of (1) around the position of \( w_k \). The detailed derivation
is described in the next subsection.
The update vector of the proposed NAQ is given by
\[ v_{k+1} = \mu v_k - \alpha_k \hat{H}_k \nabla E(w_k + \mu v_k), \] (7)
where \( \hat{H}_k \) is the symmetric positive definite matrix and iteratively approximated by the following formula which is also derived in the next subsection:
\[ \hat{H}_{k+1} = \hat{H}_k - \frac{(\hat{H}_k q_k) p_k^T + p_k (\hat{H}_k q_k)^T}{p_k^T q_k} + \left( 1 + \frac{q_k^T \hat{H}_k q_k}{p_k^T q_k} \right) p_k p_k^T q_k, \] (8)
where \( p_k = w_{k+1} - (w_k + \mu v_k) \) and \( q_k = \nabla E(w_{k+1}) - \nabla E(w_k + \mu v_k) \). Note that (8) is equivalent to BFGS formula of (6) by replacing \( p_k \) and \( q_k \) by \( s_k \) and \( y_k \), respectively. This means that NAQ can be easily applied to other modifications of QN-based algorithms. The algorithm and diagram of NAQ are illustrated in Algorithm 2 and Fig. 2, respectively. The matrix of \( \hat{H}_k \) is updated using both of the normal gradient \( \nabla E(w_{k+1}) \) and Nesterov’s accelerated gradient \( \nabla E(w_k + \mu v_k) \) in NAQ. Therefore the computational cost is equivalent to QN except that the gradient is calculated two times within a training loop whereas QN needs one calculation of the normal gradient. This is a disadvantage of NAQ, but the algorithm can further shorten the iteration counts to cancel out the effect of the shortcoming compared with QN.

**Algorithm 2: NAQ**

1. \( k = 1 \)
2. Initialize \( w_1 \) and \( \hat{H}_1 \) by uniform random numbers and the unit matrix \( I \), respectively;
3. \( v_1 = 0 \);
4. While(\( ||\nabla E(w_k)|| > \epsilon \) and \( k < k_{max} \))
   5. Calculate \( \nabla E(w_k + \mu v_k) \);
   6. Execute Line search to decide the step size \( \alpha_k \) using Armijo’s condition;
   7. Update \( v_{k+1} = \mu v_k - \alpha_k \hat{H}_k \nabla E(w_k + \mu v_k) \);
   8. Update \( w_{k+1} = w_k + v_{k+1} \);
   9. Calculate \( \nabla E(w_{k+1}) \);
10. Update \( \hat{H}_{k+1} \) using (8);
11. \( k = k + 1 \)
12. EndWhile

![Fig. 2. Geometrical view of NAQ.](image)

**3.2 Derivation and convergence of NAQ**

First of all, let \( \Delta w \) be the vector \( \Delta w = w - (w_k + \mu v_k) \), the quadratic approximation of (1) at \( w_k + \mu v_k \) is derived as
\[
E(w) \approx E(w_k + \mu v_k) + \nabla E(w_k + \mu v_k)^T \Delta w + \frac{1}{2} \Delta w^T \nabla^2 E(w_k + \mu v_k) \Delta w. \] (9)

The minimizer of this quadratic function is explicitly given by \( \Delta w = -\nabla^2 E(w_k + \mu v_k)^{-1} \nabla E(w_k + \mu v_k) \). Therefore the new iterate is defined as
\[ w_{k+1} = (w_k + \mu v_k) - \nabla^2 E(w_k + \mu v_k)^{-1} \nabla E(w_k + \mu v_k). \] (10)
This iteration is considered as Newton method with the momentum term $\mu v_k$. Here Hessian $\nabla^2 E(w_{k+1} + \mu v_k)$ is approximated by $B_{k+1}$ and the rank-2 updating formula of this matrix is derived.

Let $p_k = w_{k+1} - (w_k + \mu v_k)$ and $q_k = \nabla E(w_{k+1}) - \nabla E(w_k + \mu v_k)$, the relation which is called the secant condition $q_k = B_{k+1} p_k$ results from (10) and the optimally condition of (9). The suitable rank-2 updating formula for $B_{k+1}$ is derived as follows. The matrix $B_{k+1}$ is defined using arbitrary vectors $t$ and $u$ and constants $a$ and $b$ as,

$$B_{k+1} = \hat{B}_k + att^T + buu^T.$$  \hspace{1cm} (11)

Substitute (11) into the secant condition of $q_k = B_{k+1} p_k$,

$$q_k = (\hat{B} + att^T + buu^T) p_k = \hat{B}_k p_k + at (t^T p_k) + bu (u^T p_k).$$  \hspace{1cm} (12)

Since $t^T p_k$ and $u^T p_k$ are scalars, both of conditions $t = q_k$ and $u = -\hat{B}_k p_k$ are necessary to the secant condition of (12). Furthermore scalars $a$ and $b$ are given by $a (t^T p_k) = 1$ and $b (u^T p_k) = 1$, respectively. As a result the rank-2 updating formula for NAQ is defined as

$$\hat{B}_{k+1} = \hat{B}_k + \frac{q_k q_k^T}{q_k^T p_k} - \frac{\hat{B}_k p_k p_k^T \hat{B}_k}{p_k^T B_k p_k}.$$  \hspace{1cm} (13)

Applying the Sherman-Morrison-Woodbury formula [10] to (13), the update formula of (8) for the inverse Hessian approximation $\tilde{H}_{k+1} = \hat{B}_{k+1}^{-1}$ is derived. Note that the above derivation is based on BFGS formula [10, 11].

Next, it is shown that $\hat{B}_{k+1}$ which is derived from (13) is the symmetric positive definite matrix under the Definition: $B_k$ is the symmetric positive definite matrix. Here the following conditions are guaranteed for the above:

(a): $B_{k+1}$ of (13) satisfies the secant condition $q_k = B_{k+1} p_k$.

(b): If $B_k$ is symmetry, $B_{k+1}$ is also symmetry.

(c): If $B_k$ is the positive definite matrix, $B_{k+1}$ is also the positive definite matrix.

Proof of (a):

Form (13) and the secant condition $q_k = B_{k+1} p_k$:

$$\hat{B}_{k+1} p_k = \left( \hat{B} + \frac{q_k q_k^T}{q_k^T p_k} - \frac{B_k p_k p_k^T B_k}{p_k^T B_k p_k} \right) p_k = \hat{B}_k p_k + \frac{q_k q_k^T}{q_k^T p_k} p_k - \frac{B_k p_k p_k^T B_k}{p_k^T B_k p_k} p_k = q_k$$

\( \square \)

Proof of (b): This is clear from (13).

Proof of (c):

First, $q_k^T p_k > 0$ will be shown. When the step size $\alpha_k$ is calculated by the exact line search, that is, from $dE(w_{k+1})/d\alpha_k = -\nabla E(w_{k+1})^T \hat{H}_k \nabla E(w_k + \mu v_k) = 0$,

$$q_k^T p_k = \alpha_k \nabla E(w_k + \mu v_k)^T \hat{H}_k \nabla E(w_k + \mu v_k) > 0,$$  \hspace{1cm} (14)

is derived. It is guaranteed in (14) that $\hat{H}_k$ is the positive definite matrix because it is the inverse matrix of $B_k$, and $\nabla E(w_k + \mu v_k) \neq 0$.

Second, the positive definiteness of $\hat{B}_{k+1}$, that is, let $r \neq 0$ be an arbitrary vector, $r^T \hat{B}_{k+1} r > 0$ will be shown. Because $B_k$ is the positive definite matrix, it can be divided as $B_k = CC^T$ using an arbitrary non-singular matrix $C$. Let $\tilde{t} = C^T r (\neq 0)$ and $\tilde{u} = C^T p_k (\neq 0)$, it is shown that

$$r^T \hat{B}_{k+1} r = \frac{(\tilde{t}^T \tilde{t})(\tilde{u}^T \tilde{u}) - (\tilde{t}^T \tilde{u})^2}{\tilde{u}^T \tilde{u}} + \frac{(r^T q_k)^2}{q_k^T p_k} \geq 0,$$  \hspace{1cm} (15)

with the Cauchy-Schwarz inequality [10] and the condition of (14). In (15) the equal condition is satisfied, if and only if $(\tilde{t}^T \tilde{t})(\tilde{u}^T \tilde{u}) - (\tilde{t}^T \tilde{u})^2 = 0$ and $r^T q_k = 0$. The former equation holds when $\tilde{t} = \gamma \tilde{u}$ with the arbitrary scalar $\gamma (\neq 0)$. When $\tilde{t} = \gamma \tilde{u}$, then $r = \gamma p_k$. Therefore the later equation is transformed as $r^T q_k = \gamma p_k^T q_k = 0$. This contradicts (14). Then the equal condition of (15) is not
It is confirmed that the proposed NAQ has a similar convergence property with QN because $\tilde{B}_{k+1}$ updated by (13) holds symmetry and positive definiteness and $\tilde{H}_{k+1}$ is the inverse matrix of $B_{k+1}$.

4. Simulation results

Computer simulations are conducted in order to demonstrate the validity of NAQ. The structure of feedforward neural network considered here has a hidden layer. Each neuron has a sigmoid function as $\text{sig}(\beta) = 1/(1+\exp(-\beta))$. The performance of NAQ is compared with the performances of SG, NAG, Quickprop (Qprop) [12] which are first order algorithms and QN. Thirty (30) independent runs were performed for the function approximation of example 1 and microwave circuit modeling problems, and 10 runs were conducted for the function approximation of example 2 with different starting values of $\mathbf{w}$ which are initialized by uniform random numbers within $[-0.5, 0.5]$. Simulations were performed on the computer which has an Intel Core i7 3930K 3.2GHz processor with 16GB memory. Each trained neural network was estimated by the average, best and worst of $E(\mathbf{w}) \times 10^3$, the average of computational time $T(\text{sec})$ and the average of iteration counts (epochs). The errors for all algorithms as shown in (16) and (17). In this paper, the parameter $\xi$ in the Armijo’s condition is set to $1 \times 10^{-6}$ for function approximation problems of ex.1 and $\xi = 1 \times 10^{-8}$ for ex.2 and two microwave circuit modeling problems. The step size $\alpha$ is determined by Armijo’s conditions for all algorithms as shown in (16) and (17). In this paper, the parameter $\xi$ in the Armijo's condition is set to $1 \times 10^{-3}$. SG and NAG were also simulated using the fixed stepsize as $\alpha = 0.01$ which was determined by some experiments because these first-order methods need many back-tracking in Armijo’s condition. That may cause wasteful time consuming in some cases.

Find a stepsise $\alpha_k > 0$ satisfying the Armijo’s-type line search condition:

For SG, Qprop and QN

$$E(\mathbf{w}_k + \alpha_k \mathbf{g}_k) \leq E(\mathbf{w}_k) + \xi \alpha_k \nabla E(\mathbf{w}_k)^T \mathbf{g}_k, \quad (16)$$

For NAG and NAQ

$$E(\mathbf{w}_k + \mu \mathbf{v}_k + \alpha_k \mathbf{g}_k) \leq E(\mathbf{w}_k + \mu \mathbf{v}_k) + \xi \alpha_k \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)^T \mathbf{g}_k, \quad (17)$$

where $0 < \xi < 1$ and $\mathbf{g}_k$ denotes the direction vector of each algorithm. That is, $\mathbf{g}_k$s of SG, NAG, Qprop, QN, and NAQ are $-\nabla E(\mathbf{w}_k), -\nabla E(\mathbf{w}_k + \mu \mathbf{v}_k), -\lambda \nabla E(\mathbf{w}_k), -\mathbf{H} \nabla E(\mathbf{w}_k)$ and $-\mathbf{H} \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$, respectively. $\lambda$ is the diagonal matrix in which $i$-th element given by $\lambda_{ii,k} = s_{i,k}/y_{i,k}$, where $s_{i,k}$ and $y_{i,k}$ are $i$-th elements of vectors $\mathbf{s}_k$ and $\mathbf{y}_k$, respectively.

4.1 Simulation results for function approximation problems

Example 1

First, (18) is used as a function approximation problem with high-linearity [5,15],

$$f_1(x) = 1 + (x + 2x^2) \sin(-x^2), \quad |x| \leq 4. \quad (18)$$

The number of hidden neurons is 7. The input and output are $x$ and $f_1(x)$, respectively. $T_r$ and $T_e$ include 400 training and 10,000 testing points, respectively. $T_r$ and $T_e$ are generated with 0.02 intervals and the random sampling in $x \in [-4, 4]$, respectively. The results are presented in Table I. The average, best and worst training and testing errors are shown in this table. Several momentum parameters are tested for NAG and the proposed NAQ as done in [7]. From the Table I, it is confirmed that NAQ with $\mu = 0.95$ is the fastest algorithm among the simulations, but the quality of solutions was inferior compared with QN and NAQ with different $\mu$s. NAQs with $\mu = 0.85$ and 0.9 can obtain
the same quality of solutions keeping the fast simulation time compared with QN. Namely NAQ is faster than QN but the quality of the solutions depends on the momentum parameter $\mu$. On the other hand first order methods could not obtain enough small errors even if the simulations utilized a lot of times and epochs. Especially, SG and NAG with the fixed stepsize never obtained an enough small error. This means that the neural modeling of (18) with 7 hidden neurons can be regarded as a relatively difficult function approximation problem with high-nonlinearity [5]. Next the validity of $\mu$ is considered. Table II illustrates the averages of the training errors, the simulation times and epochs for NAQ and NAG with variable $\mu$. From this table it is confirmed that NAG is efficient with larger $\mu$ as shown in [7]. On the other hand the convergence speed becomes faster as $\mu$ increases in NAQ. Therefore larger $\mu$s that are 0.8, 0.85, 0.9 and 0.95 are selected in the simulations. The effectiveness of the momentum term is studied by showing the average training errors $E_{\text{train}}(w)$ among 30 runs without the termination condition of QN and NAQs as shown in Fig. 3. From this figure it is shown that the errors of NAQs drastically decrease in the early stage of training compared with QN. As a result it is confirmed that the simulation times of NAQs are faster than QN in spite of time consuming training loop because of twice calculation for the gradient compared with QN. On the other hand it is also confirmed that the quality of solutions for NAQ depends on $\mu$ from the final stage in Fig. 3.

| stepsize | $\mu$ | Training error $E_{\text{train}}(w)$ Ave./Best/Worst | Ave. T(sec) | Ave. epochs | Testing error $E_{\text{test}}(w)$ Ave./Best/Worst |
|----------|-------|----------------------------------------------------|-------------|-------------|-----------------------------------------------|
| SG       | fixed | 60.1/60.0/60.1 9.95                                  | 100,000     | 61.9/61.7/61.9 |
| Armijo   | -     | 54.6/52.6/58.1 34.4                                  | 100,000     | 56.5/54.6/59.5 |
| NAG      | fixed | 59.2/58.2/60.1 10.4                                  | 100,000     | 60.8/59.7/61.9 |
|          | 0.8   | 58.8/58.1/60.1 10.4                                  | 100,000     | 60.4/59.6/61.9 |
|          | 0.9   | 58.4/58.0/59.9 10.4                                  | 100,000     | 59.9/59.5/61.6 |
|          | 0.95  | 57.9/57.4/58.3 10.3                                  | 100,000     | 59.4/58.9/59.7 |
| Armijo   | 0.8   | 27.0/7.95/51.5 33.0                                  | 100,000     | 27.4/8.08/53.4 |
|          | 0.85  | 31.7/5.93/60.2 24.3                                  | 71,700      | 32.3/6.17/61.9 |
|          | 0.9   | 37.5/4.57/60.2 19.5                                  | 56,876      | 38.4/4.75/61.9 |
|          | 0.95  | 48.6/3.33/60.2 9.11                                  | 26,707      | 49.9/3.37/61.9 |
| Qprop    | Armijo| 13.6/1.21/31.3 31.7                                  | 100,000     | 13.6/1.21/31.6 |
| QN       | Armijo| 4.84/0.439/19.1 6.31                                  | 19,874      | 4.69/0.439/18.6 |
| NAQ      | Armijo| 6.70/0.411/18.7 2.22                                  | 4.668       | 6.56/0.414/18.2 |
|          | 0.85  | 4.60/0.442/18.6 1.97                                  | 4.152       | 4.53/0.438/18.2 |
|          | 0.9   | 4.53/0.438/18.2 1.57                                  | 3.309       | 4.09/0.415/18.5 |
|          | 0.95  | 11.5/0.469/60.1 0.898                                 | 1.894       | 11.4/0.465/61.9 |

<Example 2>

Second a larger problem for the approximation of (19) is utilized to show the validity of the proposed NAQ.

$$f_2(x_1, \cdots, x_n) = \frac{\pi}{n} \left\{ \sum_{i=1}^{n-1} \left[ (x_i - 1)^2 \left( 1 + 10 \sin^2 (\pi x_{i+1}) \right) \right] + 10 \sin^2 (\pi x_1) + (x_n - 1)^2 \right\},$$

$$x_i \in [-4, 4], \forall i.$$  \hspace{1cm} (19)

$f_2$ is Levy function [14]. This is usually used as a benchmark problem for the multimodal function optimization. This means that the function has a huge number of local minima and can be regarded as a highly nonlinear function for neural modeling in this paper. The simulations were conducted under following definitions. The dimension $n$ of input vector $x$ is set to 5. The input and output are $x_1, \cdots, x_5$ and $f_2(x_1, \cdots, x_5)$, respectively. The trained network has 150 hidden neurons, namely the dimension of $w$ is 1,051. 15,000 points are generated by uniformly random number in $x_i \in [-4, 4]$.
Table II. Effect of the parameter $\mu$.

| $\mu$ | NAQ with Armijo | NAG with Armijo |
|-------|-----------------|-----------------|
|       | $E_{\text{train}}(w)$ Ave/Best/Worst | $E_{\text{train}}(w)$ Ave/Best/Worst |
|       | T(sec) epochs | Ave/Best/Worst T(sec) epochs |
| 0.05  | 8.11/0.421/18.1 8.99 18,969 | 54.7/52.7/58.0 34.4 100,000 |
| 0.1   | 7.82/0.448/19.1 8.74 18,434 | 54.7/52.7/57.9 44.1 100,000 |
| 0.15  | 6.67/0.397/19.2 8.83 18,629 | 54.6/52.7/57.6 34.3 100,000 |
| 0.2   | 7.82/0.452/18.7 7.57 15,970 | 54.6/52.7/57.7 34.4 100,000 |
| 0.25  | 7.49/0.437/18.7 7.45 14,157 | 54.4/52.4/56.4 33.2 100,000 |
| 0.3   | 7.61/0.429/18.7 6.90 14,557 | 54.2/52.3/55.9 34.2 100,000 |
| 0.35  | 7.99/0.414/18.7 6.65 14,035 | 54.1/52.0/56.0 33.0 100,000 |
| 0.4   | 8.50/0.440/18.7 5.90 12,453 | 53.9/51.8/56.8 33.0 100,000 |
| 0.45  | 8.31/0.497/18.7 5.52 11,650 | 53.6/51.6/55.3 34.1 100,000 |
| 0.5   | 7.99/0.620/19.1 5.20 10,970 | 53.5/51.4/54.7 34.0 100,000 |
| 0.55  | 9.74/0.692/19.1 4.63 9,773 | 53.0/50.0/54.4 34.5 100,000 |
| 0.6   | 9.50/0.464/18.7 3.95 8,324 | 52.6/47.9/54.1 34.0 100,000 |
| 0.65  | 9.35/0.381/19.1 3.60 7,595 | 48.2/16.0/53.6 33.6 100,000 |
| 0.7   | 10.8/0.443/19.3 2.98 6,293 | 44.9/9.84/53.1 33.6 100,000 |
| 0.75  | 7.71/0.532/18.6 2.96 6,255 | 38.2/12.8/52.5 33.5 100,000 |
| 0.8   | 6.70/0.411/18.7 2.22 4,685 | 27.0/7.95/51.5 33.0 100,000 |
| 0.85  | 4.60/0.442/18.6 1.97 4,151 | 31.7/5.93/60.2 24.5 100,000 |
| 0.9   | 4.53/0.438/18.2 1.57 3,309 | 37.5/4.57/60.2 19.5 56,876 |
| 0.95  | 11.5/0.469/60.1 0.898 1,894 | 48.6/3.33/60.2 9.11 26,707 |

Fig. 3. The average training errors for iteration count.

and 10,000 and 5,000 points are utilized for $T_r$ and $T_e$, respectively. The simulation results are summarized in Table III. The first-order algorithms of SG and NAG could not converge within the maximum iteration of $k_{\text{max}}$ whereas second-order methods could converge to the small errors within $k_{\text{max}}$. NAQ could obtain good training and testing errors regardless of the parameters of $\mu$ in the simulations. Furthermore, it is confirmed that NAQ is faster than other algorithms.

4.2 Simulation results for microwave circuit modeling problems

Neural networks can be trained using measured or simulated microwave device data, such as EM and physical data. The trained neural networks can be used as models of microwave devices in place of CPU-intensive EM/physics models to significantly speed up circuit design while maintaining EM/physics-level accuracies [2, 4]. Neural network based modeling has been used to model a variety
### Table III. Summary of simulation results of example 2.

| stepsize | μ   | Training error $E_{\text{train}}(w)$ Ave/Best/Worst | Ave. T(sec) | Ave. epochs | Testing error $E_{\text{test}}(w)$ Ave/Best/Worst |
|----------|-----|--------------------------------------------------|-------------|-------------|--------------------------------------------------|
| SG       | -   | 52.5/53.3/53.8                                  | 4,180       | 100,000     | 53.9/53.7/54.2                                    |
| Armijo   | -   | 52.2/52.2/52.2                                  | 7,550       | 100,000     | 52.6/52.6/52.6                                    |
| NAG      | 0.8 | 52.2/52.2/52.3                                  | 4,250       | 100,000     | 52.6/52.6/52.7                                    |
|          | 0.85| 52.2/52.2/52.2                                 | 4,210       | 100,000     | 52.6/52.6/52.6                                    |
|          | 0.9 | 52.2/52.2/52.2                                 | 4,210       | 100,000     | 52.6/52.6/52.6                                    |
|          | 0.95| 52.1/52.1/52.1                                 | 4,220       | 100,000     | 52.5/52.5/52.5                                    |
| Armijo   | 0.8 | 52.1/52.1/52.1                                 | 7,940       | 100,000     | 52.5/52.5/52.5                                    |
|          | 0.85| 52.1/52.1/52.1                                 | 7,940       | 100,000     | 52.5/52.5/52.5                                    |
|          | 0.9 | 52.0/51.6/52.1                                 | 7,940       | 100,000     | 52.4/52.5/52.5                                    |
|          | 0.95| 35.9/21.6/42.6                                 | 7,970       | 100,000     | 36.4/21.7/43.3                                    |
| Qprop    | Armijo | 0.162/0.09/0.23                           | 5,610       | 100,000     | 0.167/0.0922/0.23                                 |
| QN       | Armijo | 0.0101/0.00525/0.0164 | 2,239       | 38,156      | 0.013/0.00823/0.0189                              |
| NAQ      | Armijo | 0.08/0.00649/0.00159  | 1,572       | 16,114      | 0.0131/0.00935/0.0191                             |
|          | 0.85 | 0.0102/0.00621/0.0148                          | 1,411       | 14,366      | 0.0132/0.00923/0.0177                             |
|          | 0.9  | 0.00842/0.00526/0.015                          | 1,350       | 13,752      | 0.0114/0.00824/0.018                              |
|          | 0.95 | 0.00956/0.0054/0.0142                          | 1,458       | 14,856      | 0.0125/0.00843/0.0171                             |

![Fig. 4. Layout of rectangular waveguide filter (WGF).](image)

Of microwave circuit components at both device and circuit levels.

<Neural modeling of waveguide filter>

In this example the proposed NAQ can be applied to develop a neural network model of a rectangular waveguide filter (WGF) [2–6, 15] showing the effectiveness of NAQ for a real-world problems. The layout of the waveguide structure is shown in Fig. 4. The length ($x$-axis) of the waveguide is 35mm, its width ($y$-axis) is 7.11mm and its height ($z$-axis) is 3.55mm. The length of the middle post is 3.47mm, with a width of 1.00mm and a height of 3.55mm. Each side post is 1.00mm long, 1.00mm wide and 3.55mm high. The distance between the side posts and the middle post is varied to obtain different center frequencies. Nine different sets of data were generated, each representing a different conducting post distance $d$ from 3.88 to 4.04mm. Five of the sets ($d = 3.88, 3.92, 3.96, 4.00$ and $4.04$mm) were used for training and the other four sets ($d = 3.90, 3.94, 3.98$ and $4.02$mm) were used for testing. The frequency range of each set was 35 to 39GHz and contained 251 frequency points. The numbers of training and testing samples were 1,255 and 1,004, respectively. The training and testing data were generated using HFSS [15], which is the industry-standard tool for simulating 3D full-wave electromagnetic field. HFSS provides E- and H-fields, currents, S-parameters, and near- and far-radiated-field results. HFSS utilizes a 3D full-wave finite element method [16] to compute the electrical behavior of high-frequency and high-speed components. The inputs of the neural network, $x_1$ and $x_2$ are the post distance $d$ and frequency $f$, respectively. The outputs, $o_1$ and $o_2$ are the magnitudes of the S-parameters, $|S_{11}|$ and $|S_{21}|$, respectively. Training data set is illustrated in Fig. 5. The number of hidden neurons is 8. The simulation results are shown in Table IV. The table shows that the first-order methods could not obtained enough small errors. Especially, NAGs with $\mu = 0.9$ and $\mu = 0.95$
often resulted in huge training errors. NAQ with $\mu = 0.8$ has the same quality of training and testing errors compared with QN while keeping fast simulation time. NAQ with $\mu = 0.9$ is fastest among all simulations whereas the quality of solutions is inferior. Note that the testing errors of $E_{test}(w)$ are showing a very good match between the models trained by QN and NAQ with $\mu = 0.8$ and the test data. As a result, neural networks with 8 hidden neurons can be used as practical models providing a rapid analysis of EM behavior for microwave-circuit CAD as shown in [3, 5].

**Table IV.** Summary of simulation results of rectangular waveguide filter (WGF).

| stepsize | $\mu$ | Training error $E_{train}(w)$ | Ave. | Ave. | Testing error $E_{test}(w)$ | Ave. | Ave. | Ave. |
|----------|-------|-------------------------------|------|------|----------------------------|------|------|------|
|          | Ave/Best/Worst | T(sec) | epochs | Ave/Best/Worst | T(sec) | epochs | Ave/Best/Worst | T(sec) | epochs |
| SG       | fixed | - | 13.6/12.0/17.4 | 29.4 | 100,000 | 15.3/13.6/19.2 | 29.4 | 100,000 | 15.3/13.6/19.2 |
|          | Armijo | - | 5.12/5.07/5.28 | 50.7 | 100,000 | 6.21/6.17/6.35 | 50.7 | 100,000 | 6.21/6.17/6.35 |
| NAG      | fixed | 0.8 | 9.06/6.76/11.6 | 27.5 | 100,000 | 10.5/7.96/13.2 | 27.5 | 100,000 | 10.5/7.96/13.2 |
|          | 0.85 | 7.63/6.30/10.9 | 30.5 | 100,000 | 8.94/7.48/12.5 | 30.5 | 100,000 | 8.94/7.48/12.5 |
|          | 0.9 | 6.30/5.83/7.53 | 30.7 | 100,000 | 7.48/6.96/8.87 | 30.7 | 100,000 | 7.48/6.96/8.87 |
|          | 0.95 | 5.55/5.29/6.56 | 27.4 | 100,000 | 6.64/6.36/7.77 | 27.4 | 100,000 | 6.64/6.36/7.77 |
| Armijo   | 0.8 | 6.41/5.38/11.0 | 43.3 | 100,000 | 7.55/6.48/12.2 | 43.3 | 100,000 | 7.55/6.48/12.2 |
|          | 0.85 | 6.71/5.06/13.2 | 34.6 | 100,000 | 7.90/6.22/14.8 | 34.6 | 100,000 | 7.90/6.22/14.8 |
|          | 0.9 | 27.8/5.70/107.0 | 31.7 | 90,002 | 29.3/6.82/109.0 | 31.7 | 90,002 | 29.3/6.82/109.0 |
|          | 0.95 | 58.2/2.09/114.0 | 21.8 | 60,014 | 59.8/2.79/115.0 | 21.8 | 60,014 | 59.8/2.79/115.0 |
| Qprop    | Armijo | - | 1.31/0.904/3.23 | 38.0 | 100,000 | 1.93/1.44/4.16 | 38.0 | 100,000 | 1.93/1.44/4.16 |
| QN       | Armijo | - | 0.926/0.779/1.41 | 12.1 | 24566 | 1.47/1.35/2.06 | 12.1 | 24566 | 1.47/1.35/2.06 |
| NAQ      | Armijo | 0.8 | 0.947/0.857/1.42 | 4.49 | 7442 | 1.47/1.36/2.00 | 4.49 | 7442 | 1.47/1.36/2.00 |
|          | 0.85 | 1.03/0.779/3.21 | 4.95 | 7331 | 1.58/1.35/4.06 | 4.95 | 7331 | 1.58/1.35/4.06 |
|          | 0.9 | 1.15/0.856/3.45 | 3.57 | 5857 | 1.71/1.34/4.31 | 3.57 | 5857 | 1.71/1.34/4.31 |
|          | 0.95 | 8.90/0.855/87.7 | 6.97 | 8202 | 9.41/1.34/86.1 | 6.97 | 8202 | 9.41/1.34/86.1 |

**<Neural network modeling of microstrip low-pass filter>**

Finally the proposed algorithm is applied to develop a neural network model of a microstrip low-pass filter (LPF) [2–6, 15] illustrated in Fig. 6. The dielectric constant and height of the substrate of LPF are 9.3 and 1mm, respectively. The length $D$ ranges 12-20mm for training data and 13-19mm for testing data at intervals of 2mm. The frequency range was 0.1 to 4.5GHz. Each set contains 221 samples. The numbers of samples $|T_r|$ and $|T_e|$ are 1,105 and 884, respectively. The training and
Fig. 6. Layout of microstrip low-pass filter (LPF).

Fig. 7. Training data set of microstrip low-pass filter (LPF).

Testing data were generated using Sonnet [17], which is a simulation tool for predominantly planar (3D planar) circuits and antennas. 3D planar circuits consist of layers of metal traces embedded in a stratified dielectric material. Sonnet uses the method of moments [18] which is applied directly to Maxwell’s equations to solve planar problems. The inputs of the neural network, $x_1$ and $x_2$ are length $D$ and frequency $f$, respectively. The outputs, $o_1$ and $o_2$ are the magnitudes of $S$-parameters, $|S_{11}|$ and $|S_{21}|$, respectively. Training data is illustrated in Fig. 7. As shown in Fig. 7 there are many irregularly aligned poles in S-parameters and the modeling of the poles is the most important in microwave circuit problems. Therefore the microwave circuit modeling is a strong nonlinear problem and needs very small training and testing errors. The number of hidden neurons is 45. The summary of results is shown in Table V. The table shows that the first-order methods could not obtain enough small errors. Especially, NAG with $\mu = 0.95$ was the fastest among the simulations but often had very large training errors. Furthermore it is confirmed that the second-order algorithms can improve the training errors compared with the first-order ones. NAQ was much faster than QN and was able to obtain sufficiently small errors. Especially, when $\mu = 0.95$, NAQ is the fastest among the simulations of the second-order methods. On the other hand the results also depend on the parameter of the momentum. Therefore the careful tuning of the momentum parameter is necessary for the proposed NAQ training.

5. Conclusions
In this research a novel quasi-Newton-based training technique of neural networks has been presented. The proposed algorithm called NAQ could be accelerated by the quadratic approximation of the error function incorporating Nesterov’s accelerated gradient. Furthermore it has been confirmed that the derivation process and the convergent property of NAQ were similar to the conventional QN. From the simulation results for two function approximation problems and two microwave circuit problems which are real world problems, it has been confirmed that the inertial term is effective to accelerate the QN training just as it is successful in the first-order algorithms and the method is faster than the conventional method without compromising quality of training solutions. It helps provide accurate neural network models much fast.
Table V. Summary of simulation results of microstrip low-pass filter (LPF).

| stepsize | Training error $E_{train}(w)$ | Ave./Best/Worst | Ave. T(sec) | Ave. epochs | Testing error $E_{test}(w)$ | Ave./Best/Worst |
|-----------|-------------------------------|----------------|-----------|-------------|----------------------------|----------------|
| SG fixed  | 26.7/26.6/26.7                | 95.1           | 100,000   | 22.5/22.5/22.6 |
| Armijo    | 25.4/25.2/25.5                | 174            | 100,000   | 21.7/21.6/21.8 |
| NAG fixed | 25.9/25.8/26.0                | 96.2           | 100,000   | 21.9/21.9/22.0 |
| 0.8       | 25.7/25.7/25.9                | 96.0           | 100,000   | 21.9/21.8/21.9 |
| 0.9       | 25.6/25.5/25.7                | 97.4           | 100,000   | 21.8/21.8/21.9 |
| 0.95      | 25.2/24.9/25.5                | 96.0           | 100,000   | 21.6/21.4/21.8 |
| Armijo    | 23.1/15.8/29.2                | 126            | 95,100    | 20.0/13.0/25.5 |
| 0.85      | 21.5/14.3/29.5                | 119            | 90,231    | 18.5/11.5/26.0 |
| 0.9       | 62.7/12.3/83.2                | 97.4           | 75,003    | 59.1/9.54/81.9 |
| 0.95      | 30.4/11.3/108.0               | 52.4           | 40,041    | 26.9/9.25/101.0 |
| Qprop     | 10.4/6.79/18.7                | 140            | 100,000   | 8.33/4.48/18.5 |
| QN        | 0.762/0.623/1.26              | 149            | 98,212    | 1.33/0.22/5.19 |
| NAQ       | 0.720/0.583/1.09              | 132            | 53,035    | 1.88/0.23/17.8 |
| 0.85      | 0.690/0.533/0.925             | 109            | 45,670    | 0.792/0.422/2.03 |
| 0.9       | 0.699/0.552/0.910             | 97             | 40,326    | 1.64/0.45/11.3 |
| 0.95      | 0.651/0.442/1.00              | 89             | 37,206    | 0.910/0.42/3.16 |

In the future the momentum parameters $\mu$ will be studied. This parameter was analytically determined for the first-order method of NAG in [8,9] whereas the fixed values were used in [7] for the neural training problems in the same way as this paper. Therefore the analytical studies of the momentum parameter for the second-order method of NAQ will be done in the future. Furthermore 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. Note that Radial Basis Function (RBF) networks may work very well for the modeling of stronger nonlinear functions [1]. Therefore NAQ will be demonstrated for RBF networks. NAQ proposed here was a batch mode process in which the gradient was calculated two times within a loop. The online (stochastic) form was the most frequently used for the training, particularly for huge and redundant data problems [19, 20]. Typical QN with online process was oBFGS in [19]. oBFGS was necessary to calculate the gradient two times within a loop because the training samples (minibatch) are changed with respect to each loop. Therefore its algorithm will be very compatible with NAQ. That is, the stochastic mode NAQ will improve the convergence speed compared with the stochastic mode QNs. On the other hand, QN needed the limited-memory methods such as LBFGS for the large scale neural network training to reduce the required memories of the approximated Hessian [9]. Therefore the limited-memory technique will be applied to NAQ for the training of large scale networks.

Acknowledgments
The author thanks Prof. Q.J. Zhang at Carleton University, Canada, for his support of microwave circuit models. This work was supported by Japan Society for the Promotion of Science (JSPS), KAKENHI (26330287).

References
[1] S. Haykin, “Neural networks and learning machines 3rd,” Pearson, 2009.
[2] Q.J. Zhang, K.C. Gupta, and V.K. Devabhaktuni, “Artificial neural networks for RF and microwave design-from theory to practice,” IEEE Trans. Microwave Theory and Tech., vol. 51, pp. 1339–1350, April 2003.
[3] H. Ninomiya, S. Wan, H. Kabir, X. Zhang, and Q.J. Zhang, “Robust training of microwave neural network models using combined global/local optimization techniques,” IEEE MTT-S International Microwave Symposium (IMS) Digest, pp. 995–998, June 2008.
[4] H. Kabir, L. Zhang, M. Yu, P.H. Aaen, J. Wood, and Q.J. Zhang, “Smart modeling of microwave devices,” IEEE Microwave Magazine, pp. 105–118, May 2010.

[5] H. Ninomiya, “An improved online quasi-Newton method for robust training and its application to microwave neural network models,” Proc. IJCNN’10, pp. 792–799, July 2010.

[6] H. Ninomiya, “Microwave neural network models using improved online quasi-Newton training algorithm,” Journal of Signal Processing, vol. 15, no. 6, pp. 483–488, November 2011.

[7] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” Proc. ICML’13, 2013.

[8] Y. Nesterov, “Introductory lectures on convex optimization: A basic course,” Kluwer, 2004.

[9] S. Ghadimi and G. Lan, “Accelerated gradient methods for nonconvex nonlinear and stochastic programming,” Math. Program. Series A, pp. 1–41, online 21, February 2015.

[10] J. Nocedal and S.J. Wright, “Numerical optimization 2nd,” Springer Series in Operations Research, 2006.

[11] W. Forst and D. Hoffmann, “Optimization - theory and practice,” Springer, 2010.

[12] M.N. Vrahatis, G.D. Magoulas, and V.P. Plagianakos, “Globally convergent modification of the Quickprop method,” Neural Processing Letters, vol. 12, pp. 159–169, 2000.

[13] H. Ninomiya, “Neural network training based on quasi-Newton method using Nesterov’s accelerated gradient,” Proc. IEEE/TENCON 2016, pp. 51–54, November 2016.

[14] D. Gao, N. Ruan, and W. Xing, editors, Advances in Global Optimization, Springer Proceedings in Mathematics & Statistics, 2014.

[15] HFSS, “Simulation software for high-performance electric design,” ANSOFT Corp.

[16] J.L. Volakis, A. Chatterjee, and L.C. Kempel, Finite Element Method for Electromagnetics, IEEE Press, 1998.

[17] Sonnet, “Full-wave 3D planar electromagnetic field solver software for high frequency EM simulation,” Sonnet Software, Inc.

[18] R.F. Harrington, Field Computation by Moment Method, IEEE Press, 1993.

[19] N.N. Schraudolph, J. Yu, and S. Gunter, “A stochastic quasi-Newton method for online convex optimization,” Proc. 11th Intl. Conf. Artificial Intelligence and Statistics, 2007.

[20] R.H. Byrd, S.L. Hansen, J. Nocedal, and Y. Singer, “A stochastic quasi-Newton method for large-scale optimization,” arXiv: 1401.7020v2, February 19, 2015 and SIAM Journal on Optimization, 2016.