Dynamic Sample Size Selection based quasi-Newton Training for Highly Nonlinear Function Approximation using Multilayer Neural Networks

Hiroshi Ninomiya, Member, IEEE

Abstract— This paper describes a novel robust training algorithm based on quasi-Newton iteration. The size of training samples for each iteration is dynamically and analytically determined by variance estimates during the computation of its gradient in the mini-batch based online training methodology. Furthermore, the size of mini-batch is controlled by a parameter to ensure that the number of samples in a mini-batch changes from a portion of samples (online) to all ones (batch) as quasi-Newton iteration progressed. As a result, the iteration during online mode can be shortened compared with previous quasi-Newton-based methods in which the gradient of error function for the training step was improved.

Keywords- neural networks; quasi-Newton training; online and batch training methods; Sample Size Selection; highly-nonlinear function modeling;

I. INTRODUCTION

Neural network techniques 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 [2]-[5].

Training is the most important step in developing a neural network model. Gradient based algorithms are popularly used for this purpose [1]. Especially, quasi-Newton method, which is one of the most efficient optimization techniques [6] is widely utilized as the robust training algorithm for highly nonlinear function approximation using feedforward neural networks [1]-[5]. For a given set of training samples, the gradient algorithm operates in one of two modes: online (stochastic) or batch. Despite its disadvantages, the online form is the most frequently used for the training, particularly for large-scale problems. Moreover, the online training follows noisy gradients, and so might move out of a local minima [1][7][8]. On the other hand, the quasi-Newton training was usually operated as batch mode. In [9], the online quasi-Newton based training algorithm referred to as oBFGS (Online quasi-Newton based Broyden-Fletcher-Goldfarb-Shanno formula [6]) was introduced as the algorithm for machine learning with huge training data set. oBFGS worked with gradients which obtained from small subsamples (mini-batches) of training data and could greatly reduce computational requirements:

(H. Ninomiya is with the Department of Information Science, Shonan Institute of Technology, 1-1-25, Tsujido-Nishikaigan, Fujisawa, Kanagawa, 251-8511, Japan (phone: +81-466-30-0218; fax: +81-466-34-5932; e-mail: ninomiya@info.shonan-it.ac.jp).

on huge, redundant data sets. However, when applied to highly nonlinear function modeling, oBFGS still converges too slowly and optimization error cannot be effectively reduced within finite time in spite of its advantage [10].

Recently, two robust training algorithms for highly nonlinear function approximation were proposed, improving the gradient of error function [10]-[13]. In the online mode, the use of a small sample size allows rapid progress in the early stages, while a larger sample yields high accuracy in the solution. Therefore these algorithms gradually changed from online to batch methods during iteration. First, the improved online quasi-Newton method based on BFGS formula (ioBFGS) was developed for robust neural network training [10][11]. The gradient was calculated by a mini-batch which is a portion of all training samples in oBFGS, but the gradient of ioBFGS was calculated by variable training samples which were automatically and heuristically increased from a mini-batch to all samples as quasi-Newton iteration progressed. The other was Parameterized Online BFGS (poBFGS) [12][13]. poBFGS was introduced, incorporating Simulated Annealing concept. In the algorithm, online and batch error functions were associated by a weighting coefficient. Then the coefficient was adjusted to ensure that the algorithm gradually changes from online to batch. These methods could obtain the strong local convergent property of the batch-mode quasi-Newton algorithm, over a long transition time. Because the change of mini-batch size had to be very slow in ioBFGS, and much more iteration was needed for the cooling schedule of Simulated Annealing in poBFGS.

On the other hand, a methodology for using varying sample sizes in training algorithms for huge scale machine learning problems was introduced in [14]. A mini-batch size that is the number of samples in itself was dynamically and analytically determined by variance estimates during the computation of its gradient in the methodology.

This paper proposes a novel robust training algorithm based on quasi-Newton method in which the size of training samples is dynamically and analytically determined by the similar methodology of [14]. Meanwhile, the mini-batch size was heuristically determined in ioBFGS which was previously proposed [10][11]. Furthermore, a parameter which controls the condition to derive the mini-batch size is introduced to ensure the transition from online to batch as iteration progressed. As a result, the proposed algorithm has strong ability to search a global minimum without being trapped into local minimum. Moreover the iteration during online mode can be shortened compared with previous methods such as ioBFGS and poBFGS. The algorithm is tested on a function approximation problem with high-nonlinearity in this research.
This paper is organized as follows. In section 2 the problem formulation and the conventional quasi-Newton methods are described. A novel training algorithm based on quasi-Newton is proposed in Section 3. The computer simulation for the proposed algorithm is demonstrated in Section 4. Section 5 includes a conclusion.

II. FORMULATION OF TRAINING AND IMPROVED ONLINE
QUASI-NEWTON TRAINING ALGORITHM

A. Formulation of Training

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

\[
E(\mathbf{w}) = \frac{1}{P_T} \sum_{p \in P_T} E_p(\mathbf{w}),
\]

where \( P_T \in \{1, \ldots, P_T\} \) denotes a training data set and \( P_T \) is the number of training samples within \( P_T \).

Among the gradient-based algorithms, (2) is minimized by the following iterative formula:

\[
\mathbf{w}^{k+1} = \mathbf{w}^k + \alpha^k \mathbf{g}^k(\mathbf{w}^k),
\]

where \( k \) is the iteration count and \( \mathbf{g}^k(\mathbf{w}^k) \) is the negative gradient vector. The gradient vector of the online and batch methods are defined as \( \mathbf{g}^k(\mathbf{w}^k) = -\nabla E(\mathbf{w}^k) \) and \( \mathbf{g}^k(\mathbf{w}^k) = -\nabla E(\mathbf{w}^k) \), respectively. In the online mode a training data set \( P_T \) is divided into \( \text{Seg} \) subsamples (mini-batches). \( \text{Seg} \) denotes the number of mini-batches within \( P_T \). A mini-batch is called a “segment” and includes \( P_{T,r}\text{-updates} \) of weight vector \( \mathbf{w} \) using a mini-batch. Next, a mini-batch size of \( \text{Seg} \)-updates of weight vector \( \mathbf{w} \), all training data are used. In the next \( \text{Seg} \)-iterations, (\( \text{Seg}-\text{Seg}\)) is changed to \( \text{Seg} \)-updates of weight vector \( \mathbf{w} \), all training data are used. The algorithm became BFGS. The details of the increasing strategy of mini-batch size are as follows:

During first \( \text{Seg} \)-iterations (\( \text{Seg} \)-updates) of \( \mathbf{w} \), training data \( P_T \) is divided into \( \text{Seg} \)-segments as

\[
T_r = \{T_{r,1}, T_{r,2}, \ldots, T_{r,(\text{Seg}-1)}, T_{r,\text{Seg}}\},
\]

where these steps are indicated as \( \text{Seg} \)-operations. Note that, in the first \( \text{Seg} \)-updates of weight vector \( \mathbf{w} \), all training data are used. In the next \( \text{Seg} \)-iterations, (\( \text{Seg}-\text{Seg}\)), (6) is changed to

\[
T_r = \{T_{r,1}, T_{r,2}, \ldots, T_{r,\text{Seg}}\},
\]

where two segments are grouped with the subsequent segment, and a mini-batch size doubles overlapping a segment. Note that, the training data set \( P_T \) still contains \( \text{Seg} \) data sets. After \( \text{Seg} \)-grouping operations, that is \( \text{Seg} \)-iterations (\( \text{Seg} \)-\( \text{Seg} \)) iterations (\( \text{Seg} \)-operations (\( \text{Seg} \)-\( \text{Seg} \))), the training data \( P_T \) is divided into

\[
T_r = \{T_{r,\text{Grp}(c,1)}, \ldots, T_{r,\text{Grp}(c)}\},
\]

where \( \text{Seg} \)-mini-batch in \( T_r \) is denoted as

\[
\text{Seg}_{\text{Grp}(c)} = \{T_{r,\text{Grp}(c)}\},
\]

which contains \( \text{Seg} \) segments. In (9), \% is modulo operation. Here, the training subsamples of the \( i \)-th mini-batch are

\[
\{x_{p,\text{Grp}(c)}|d_p, \mathbf{o}_p, \mathbf{w} \in \text{Seg}_{\text{Grp}(c)}\},
\]

After \( \text{Seg} \)-grouping operations, namely \( \text{Seg} \)-updates of \( \mathbf{w} \), the training data set \( T_r \) becomes a group. As a result, \( \text{Seg} \)-\( \text{Seg} \)-becomes the batch BFGS. The error function of \( i \)-th mini-batch in \( \text{Seg} \)-grouping is redefined as

\[
E_{\text{Grp}(c)}(\mathbf{w}, \{x^i_{p,\text{Grp}(c)}\}) = \frac{1}{P_{T,\text{Grp}(c)}} \sum_{p \in P_{T,\text{Grp}(c)}} E_p(\mathbf{w}),
\]

and \( E_{\text{Grp}(\text{Seg})}(\mathbf{w}, \{x^i_{p,\text{Grp}(\text{Seg})}\}) = E(\mathbf{w}) \). The algorithm of ioBFGS is illustrated in Algorithm 1.
Algorithm 1: ioBFGS
1. \( k = 1, c = 1; \)
2. Initialize \( w^1 \) and \( H^1 \) by uniform random numbers and the unit matrix \( I \), respectively;
3. \( \text{While}(k < k_{max} \text{ or } \|\delta E(w^k)/\delta w^k\| > \varepsilon) \)
   4. Calculate \( g(w^k) = -\delta E_{Gr}(w^k)/\delta w^k; \)
   5. Execute Line search with Armijo’s condition to decide the step size \( \eta^k; \)
6. Update \( w^{k+1} = w^k + \eta^k H^k g(w^k); \)
7. Update \( H^{k+1} \) by using BFGS formula of (5);
8. \( \text{if } (k \% \text{Seg} == 0) \text{ then update grouping counter } c; \)
   \( \text{if } (c < \text{Seg}) \text{ then } c = c + 1 ;\text{oBFGS}<c>\)
   \( \text{else } c = \text{Seg}; \) (BFGS)
9. \( k = k + 1; \)
10. EndWhile

step8 of the algorithm is excluded, this algorithm becomes online BFGS (oBFGS) [9].

III. THE PROPOSED ALGORITHM – DYNAMIC SAMPLE SIZE
BFGS (dssBFGS)

ioBFGS was a robust training algorithm, however it is necessary to slow the transition from online to batch overlapping a segment as shown in Algorithm 1. Moreover the increasing strategy of ioBFGS was heuristic.

In this research, the size of mini-batch for a training step is dynamically and analytically determined by variance estimates in the evaluation of the gradients throughout the progression of the algorithm. At every iteration, the method chooses a subset \( T_{r,s} \) of the training set \( T_r \), and applies one step of quasi-Newton training to the objective function \( E_\theta(w) \) of (4). At the start of algorithm, a relatively small sample size \( P_{T_{r,s}}(=P_{Tr}/\text{Seg}) \) is chosen, and it is judged that the training step is likely to reduce the “target” objective function \( E(w) \) of (2), then the sample size is kept unchanged, and a next training step is computed by using a larger sample size. Otherwise, the algorithm chooses a larger sample size, selects a corresponding sample \( T_{r,s} \), and computes the next step. The quality of the sample subset \( T_{r,s} \) is measured by using the variance in the gradient for this purpose. Furthermore a parameter which controls the condition to derive the mini-batch size is introduced to ensure the transition from online to batch. Finally whole training sample is used for a step as the batch mode. The proposed algorithm is referred to as Dynamic Sample Size quasi-Newton method based on BFGS formula (dssBFGS). The method of dynamic sample size selection [14] and the details of dssBFGS are shown as follows.

First of all, the dynamic sample size selection method [14] in which the variance estimates of the gradients were utilized is introduced. Each step of dssBFGS enforces descent in \( E_\theta(w) \), at the same time, the target objective function \( E(w) \) has to be also reduced sufficiently often to converge the solution. In [17], the following condition to confirm the above motivation was defined: the gradient vector \( g(w) \) is a descent direction for \( E(w) \) if
\[
\delta_\theta(w) \equiv \|E_\theta(w) - E(w)\|_2 \leq \theta \|E_\theta(w)\|_2. \quad (11)
\]

where \( \theta \in [0, 1] \). The quantity \( \delta_\theta(w) \) is, however not available, since the batch mode gradient, \( \sqrt{E(w)} \) cannot be calculated in the online mode. Therefore, \( \delta_\theta(w) \) is approximated by an estimate of the variance of \( \sqrt{E(w)} \). For a given \( w \), the expected value of \( \sqrt{E_\theta(w)} \) over all possible samples \( T_{r,s} \subset [1, \ldots, P_r] \) equals \( \sqrt{E(w)} \), so that the quantity \( \delta_\theta(w) \) satisfies
\[
\text{Ex}[\delta_\theta(w)^2] = \text{Ex}[\|\sqrt{E_\theta(w)} - \sqrt{E(w)}\|^2_2] = \|\text{Var}(\sqrt{E_\theta(w)})\|_2^2, \quad (12)
\]
where \( \text{Var}(\sqrt{E_\theta(w)}) \) is a vector of the same length as \( w \). Therefore the variance of \( \sqrt{E_\theta(w)} \) has to be estimated, where \( T_{r,s} \) is a sample subset without replacement. \( \text{Var}(\sqrt{E_\theta(w)}) \) is defined using the finite population correction [18] as
\[
\text{Var}(\sqrt{E_\theta(w)}) = \frac{\text{Var}(\sqrt{E_p(w)})}{P_{Tr,s}/P_{Tr}}. \quad (13)
\]
Since the population variance \( \text{Var}(\sqrt{E_p(w)}) \) is too expensive to compute at every iteration, it is approximated with the sample variance
\[
\text{Var}(\sqrt{E_p(w)}) \approx \frac{\text{Var}_{P_{Tr,s}}(\sqrt{E_p(w)})}{P_{Tr,s}/P_{Tr}} \leq \theta^2 \|\sqrt{E_\theta(w)}\|^2_2. \quad (16)
\]
dssBFGS requires satisfying the condition (16) at each iteration. If (16) is not satisfied, the number of samples in \( T_{r,s} \) has to be increased. Therefore \( \hat{P}_{Tr,s} \) which is slightly larger than the current sample size \( P_{Tr,s} \) is calculated under assumption that the change in sample size is gradual enough, that is,
\[
\|\text{Var}_{P_{Tr,s}}(\sqrt{E_p(w)})\|_2 \approx \text{Var}_{P_{Tr,s}}(\sqrt{E_p(w)})\|_2 \quad \text{and} \quad \|\sqrt{E_\theta(w)}\|_2 \|\sqrt{E_\theta(w)}\|_2 \leq \theta^2 \|\sqrt{E_\theta(w)}\|^2_2
\]
Therefore \( \hat{P}_{Tr,s} \) is derived by the equal condition of (16) and given by
\[
\hat{P}_{Tr,s} = \frac{V_x \cdot P_{Tr}}{\|E_\theta(w)\|_2^2} \quad (17)
\]
where \( V_x = \text{Var}_{P_{Tr,s}}(\sqrt{E_p(w)})\) and \( \|E_\theta(w)\|_2^2 \).

Furthermore the parameter \( \theta \) is controlled in this research to ensure the transition from online to batch as follows. The initial value of \( \theta \) is set to \( \theta \approx 1 \). Namely, when \( \theta \approx 1 \), (11) will be almost satisfied with any \( P_{Tr,s} \) (online) and when \( \theta = 0 \), \( P_{Tr,s} = P_{Tr} \) in (17), that is batch.
Algorithm 2: dssBFGS

1. \(k = 1, T_k = Seg \times D, \rho = 1 - 1/Seg;\)
2. Initialize \(w^1\) and \(H^1\) by uniform random numbers and the unit matrix \(I\), respectively;
3. While \(k < k_{\text{max}}\) or \(|\theta E(w^k)/\theta w^k| > \epsilon\) do:
   4. Calculate \(g(w^k) = -\theta E_{P_{tr}}(w^k)/\theta w^k;\)
5. Execute Line search with Armijo’s condition to decide the step size \(\eta^k;\)
6. Update \(w^{k+1} = w^k + \eta^k \nabla E_{P_{tr}}(w^k);\)
7. Update \(H^{k+1}\) by using BFGS formula of (5);
8. Update \(\theta_k\) using (18);
9. If condition (16) is not satisfied, a new sample size, \(P_{Tr,x}\) is calculated by using (17);
10. \(k = k + 1;\)
11. EndWhile

mode. Therefore, dssBFGS starts at \(\theta = 1\), and then \(\theta\) is gradually decreased with progressing iteration. Finally, \(\theta\) approaches 0. In that process, dssBFGS progressively changes from online to batch in the similar manner of ibBFGS. The idea of this methodology can be carried out by the following equations for the \(k\)-th iteration:

\[
\theta_k = \exp(-1/T_k) \quad \text{and} \quad T_k = \rho T_{k-1},
\]

where \(T_k\) and \(\rho(0 < \rho < 1)\) are parameters. This approach is similar to the concept of Simulated Annealing [19]. Typically, above update equations need a number of parameters which must be tuned. In order to avoid having too many parameters to tune, \(\rho\) and the initial value of \(T_k\) are defined as

\[
\rho = 1 - 1/Seg \quad \text{and} \quad T_1 = Seg \times D,
\]

respectively. In (19), \(D\) denotes the dimension of \(w\). Note that if \(Seg\) is large, the algorithm changes slowly from online to batch. Then the turning parameter of dssBFGS is only \(Seg\) as with ibBFGS. The algorithm of dssBFGS is illustrated in Algorithm 2. As a result, dssBFGS substantially improves the quality of solutions during global optimization compared with the other algorithms based on quasi-Newton method.

IV. SIMULATION RESULTS

Computer simulations are conducted in order to demonstrate the validity of dssBFGS. The structure of feedforward neural network considered here is 3-layer, that is, a network has a hidden layer. The performance of dssBFGS is compared with the performances of BFGS [1], oBFGS [9], ibBFGS [10][11], and poBFGS [12][13]. Thirty (30) independent runs were performed to all algorithms with different starting values of \(w\). Each trained neural network was estimated by average of \(E(w)\times 10^3\) and the average of computational time \(T(\text{sec})\).

In this research, (20) is used as a function approximation problem with high-nonlinearity[10]-[13][20],

\[
f(x) = 1 + (x + 2x^2)\sin(-x^2),
\]

where \(x \in [-4.4]\) is an input variable for the neural network. The number of hidden layer’s neurons is 7. The neural modeling of (20) with 7 hidden neurons can be regarded as a relatively difficult function approximation problem with high-nonlinearity because the radial basis function network with 20 hidden units was used for this problem in [20]. Furthermore, the EM behavior of microwave circuit is quite similar to the test function [2]-[5]. The maximum iteration count \(k_{\text{max}}\) is set to 1\times 10^5. \(T\) includes 400 training points. The training errors \(E(w)\) of the neural model, which are represented by the average, best, and worst errors among the 30 runs, are presented in Table 1 in which \(Seg\) in parenthesis denotes the initial number of segment and the initial mini-batch size is \(P_{Tr,x}(= P_{Tr}/Seg)\).

Several Eqs are tested for oBFGS, ibBFGS and dssBFGS. From the table, it is demonstrated that dssBFGS can obtain smaller average, best, and worst errors than that from the existing BFGS-based algorithms. Furthermore, the proposed dssBFGS with \(Seg = 20\) which is indicated as dssBFGS(20) can also obtain much smaller difference between the best and the worst errors, showing the quality of solution by dssBFGS is more certain (respect to randomly chosen starting point) than other algorithms. This means that the proposed algorithm has strong ability to search a global minimum without being trapped into local minimum. As a result, it is confirmed that dssBFGS is a robust algorithm less dependent on the choice of initial guesses of \(w\) on the other hand the best and worst errors from conventional BFGS methods except for oBFGS differ significantly, a clear indication that the previous BFGS methods depend on initial values of \(w\), with good result only if initial guess is suitable. The generalization abilities of trained neural networks are considered to ensure the above discussion by a testing data set which contains inputs \(x_i\) never seen during training (i.e. \(i \in \mathcal{T}_r\)). A comparison of the function (20) of the network models which have the best and worst testing errors of dssBFGS(20) and the worst errors of ibBFGS(100) and oBFGS(40) versus original test data is given in Fig.1, showing accuracies of the trained models for the test data. Note that, the best and worst training errors \(E(w)\) of dssBFGS(20) are \(0.2 \times 10^{-3}\) and \(0.777 \times 10^{-3}\), respectively. The worst errors of ibBFGS(100) and oBFGS(40) are \(6.3 \times 10^{-3}\) and \(1.2 \times 10^{-3}\), respectively. From Fig.1 it is confirmed that the neural models which

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Seg</th>
<th>(E(w)(\times 10^3))</th>
<th>Ave/Best/Worst</th>
<th>(T(\text{sec}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFGS</td>
<td>−</td>
<td>3.85/0.20/14.0</td>
<td>42.9</td>
<td></td>
</tr>
<tr>
<td>oBFGS(30)</td>
<td>30</td>
<td>5.78/0.35/6.2</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>oBFGS(60)</td>
<td>60</td>
<td>5.72/0.32/6.0</td>
<td>41.1</td>
<td></td>
</tr>
<tr>
<td>dssBFGS(30)</td>
<td>30</td>
<td>3.83/0.24/14.0</td>
<td>46.3</td>
<td></td>
</tr>
<tr>
<td>dssBFGS(60)</td>
<td>60</td>
<td>3.28/0.20/14.0</td>
<td>46.6</td>
<td></td>
</tr>
<tr>
<td>poBFGS</td>
<td>−</td>
<td>1.01/0.20/14.0</td>
<td>46.2</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Summary of simulation results.
have the training errors both of $0.2 \times 10^{-3}$ and $0.777 \times 10^{-3}$ are very good match for the test data. On the other hand, the trained networks with larger training errors cannot approximate the original test function. This means that the proposed dssBFGS can always provide reliable neural models regardless of initial values.

Next, the mini-batch sizes of ioBFGS and dssBFGS for iteration count are illustrated in Fig.2. ioBFGS needs $\text{Seg} \times \text{Seg}$ iteration for the transition from online to batch at all runs. On the other hand, it is confirmed in Fig.2 that the transition process of dssBFGS is controlled in such a way as to end at similar timing regardless of $\text{Seg}$. Furthermore, the iteration during online mode of dssBFGS can be shortened compared with the previous ioBFGS.

Finally, the effectiveness of the dynamic sample size selection method is studied by showing the average training errors $E(w)$ among 30 runs of dssBFGS(20), ioBFGS(20) and BFGS for iteration count in Fig.3. From Fig.3 it is shown that both of ioBFGS and dssBFGS take on the aspect of stochastic process in the early stage of quasi-Newton iteration because of the online mode. However, $E(w)$ of ioBFGS in stochastic process varies more intensely than one of dssBFGS. The detail of this process is shown in Fig.4 which includes the training sample size of mini-batch for iteration count. The mini-batch size of ioBFGS is automatically increased between 10 and 100 epochs, namely the early stage of iteration, though $E(w)$ of ioBFGS has changed enormously. On the other hand, the mini-batch size of dssBFGS remains virtually unchanged between 10 and 100 iteration, while achieving low swing of $E(w)$. As a result, the proposed dssBFGS can obtain the small training error with much certainty, incorporating the effectiveness of stochastic process in which swing of $E(w)$ is relatively kept low by the condition of (16).

\section*{V. CONCLUSION}

In this research a robust training technique of feedforward neural networks has been presented. The size of training samples for an update $w$ was dynamically and analytically determined by variance estimates during the computation of its gradient in the mini-batch based online training methodology. Furthermore, a parameter which controls the condition to derive the mini-batch size was introduced to ensure the transition from online to batch. From the simulation results for the benchmark problem it has been confirmed that the method is robust, and provides high quality training solutions regardless of initial values. It helps provide accurate neural network models for function approximation problem.

In the future the validity of the proposed algorithm for more highly nonlinear function approximation problems and the real world problems such as microwave circuit modeling [2]-[5] will be demonstrated.

\section*{ACKNOWLEDGMENT}

This work was supported by Japan Society for the Promotion of Science (JSPS), KAKENHI (23500189).

\section*{REFERENCES}

Fig.3 The average errors $E(w)$ for each iteration.

Fig.4 The average errors $E(w)$ and mini-batch size for each iteration.