IPSJ Transactions on Mathematical Modeling and Its Applications Vol.6 No.3 100–105 (Dec. 2013)

Regular Paper

Theoretical Analysis of Learning Speed in Gradient Descent Algorithm Replacing Derivative with Constant

Kazuyuki Hara[1,a)] Kentaro Katahira[2,3,b)]

Received: November 6, 2012, Revised: December 28, 2012,
Accepted: April 30, 2013

Abstract: In on-line gradient descent learning, the local property of the derivative term of the output function can slowly converge. Improving the derivative term, such as by using the natural gradient, has been proposed for speeding up the convergence. Beside this sophisticated method, we propose an algorithm that replaces the derivative term with a constant and show that this greatly increases convergence speed when the learning step size is less than 2.7, which is near the optimal learning step size. The proposed algorithm is inspired by linear perceptron learning and can avoid locality of the derivative term. We derived the closed deterministic differential equations by using a statistical mechanics method and show the validity of theoretical results by comparing them with computer simulation solutions. In real problems, the optimum learning step size is not given in advance. Therefore, the learning step size must be small. The proposed method is useful in this case.

Keywords: learning speed, derivative, gradient descent algorithm, simple perceptron, statistical mechanics

1. Introduction

Learning in neural networks can be formulated as optimization of an objective function that quantifies the system's performance. An important property of feed-forward network is their ability to learn a rule from examples. Statistical mechanics has been successfully used to study this property, mainly for the so-called simple perceptron [1], [2], [3]. A compact description of the learning dynamics can be obtained by using the statistical mechanics, which uses a large input dimension N and provides an accurate model of mean behavior for realistic N [2], [3], [4].

In the learning using feed-forward network, on-line learning and off-line learning (batch learning) are used. The corresponding objective function measures the performance of the learning network (the student) on a given set of examples. This is called off-line learning. It stores entire examples, so it would be expected to be efficient in terms of the number of examples needed for good generalization. However, it is costly. On-line learning is a common method for learning multi-layer feed-forward neural networks, especially for large systems and for problems requiring rapid and adaptive data processing. Only the latest in a sequence of examples determines the update of student weights in an iterative learning process. No explicit storage of an example set is required and the student's performance on earlier examples is not taken into account. (For comparison between batch learning and on-line learning, see Murata [5].)

There are some works that correspond to the acceleration of learning processes [6], [7], [8]. The main problem in slow learning is *plateau*, which occurs in learning processes using the gradient descent algorithm. Other works have considered local property of a derivative of the output function. In gradient descent learning, the parameters are updated in the direction of the steepest descent of the objective function. To calculate the direction, a derivative of the output function is used. When the output function is linear, the derivative is 1 and is not a function of the inner potential of output unit. However, when the output function is a sigmoid-like function, the derivative becomes a Gaussian function and is a localized function of inner potential of the output unit. The Gaussian function is non-zero for the mean and damps exponentially. Therefore, update of the parameter becomes small for large absolute value of inner potential, causing slow convergence. We scope accelerating the learning process on modifying derivative of the output function while conventional methods scope the optimization of the learning step size [9], [10].

In this paper, we propose the gradient descent algorithm replacing the derivative of the output function with a constant value, and then we analyze the learning dynamics of the proposed method by using the statistical mechanics methods. The rest of this paper is as follows. We formulate the networks, the input, the output function, and the gradient descent algorithm in Section 2. We employ teacher-student formulation that is also introduced in this section. In Section 3, we introduce some theoretical results of a conventional gradient descent method [2]. Then we propose an acceleration method and derive closed differential equations that depict dynamical behavior of the system in Section 4. In Section 5, we compare the numerical calculation of theoretical results of the proposed method with those of computer simulations, and in Section 6, we compare the generalization error of the proposed

[1] College of Industrial Technology, Nihon University, Narashino, Chiba 275–8575, Japan
[2] Center for Evolutionary Cognitive Sciences, The University of Tokyo, Meguro, Tokyo 113–8654, Japan
[3] Brain Science Institute, RIKEN, Wako, Saitama 351–0198, Japan
[a)] hara.kazuyuki@nihon-u.ac.jp
[b)] katahira@ecs.c.u-tokyo.ac.jp

method with that of conventional method [2]. We summarize the results and conclude in Section 7.

2. Formulations

In this section, we formulate the teacher and student networks, and the gradient descent algorithm employing a teacher-student formulation. We assume the teacher and student networks receive N-dimensional input $\boldsymbol{\xi}^m = (\xi_1^m, \ldots, \xi_N^m)$ at the mth learning iteration as shown in **Fig. 1**. Here, we assume the existence of a teacher weight vector $\boldsymbol{B}$ that produces desired outputs, so the teacher output $\tau(\boldsymbol{\xi})$ is a target of the student output $\sigma(\boldsymbol{J}, \boldsymbol{\xi})$. This is called teacher-student formulation.

The teacher network shown in Fig. 1 has N inputs and an output and is the same structure as the perceptron [11]. The learning iteration m is ignored in the figure. The student network has the same architecture as the teacher network. $\boldsymbol{B}$ denotes the weight vector of a teacher network with N elements. $\boldsymbol{J}^m$ denotes the weight vector of a student network with N elements at mth the learning iteration. We also assume that the elements ξ_i^m of independently drawn input $\boldsymbol{\xi}^m$ are uncorrelated random variables with zero mean and unit variance; that is, the ith element of input is drawn from a probability distribution $\mathrm{P}(\xi_i)$. In this paper, the thermodynamic limit of $N \to \infty$ is assumed. In the thermodynamic limit, the law of large numbers and the central limit theorem can apply. We can then depict the system behavior by using a small number of parameters. Statistics of the inputs at the thermodynamic limit are as follows.

$$\langle \xi_i^m \rangle = 0, \ \left\langle (\xi_i^m)^2 \right\rangle = 1, \ \|\boldsymbol{\xi}^m\| = \sqrt{N}, \tag{1}$$

where $\langle \cdots \rangle$ denotes average and $\|\cdot\|$ denotes the norm of a vector.

A perceptron is used as the teacher network and is not subject to training. Thus, the weight vector $\boldsymbol{B}$ is fixed in the learning process. The output of the teacher $\tau(\boldsymbol{\xi}^m)$ for N-dimensional input $\boldsymbol{\xi}^m$ at the mth learning iteration is

$$\tau(\boldsymbol{\xi}^m) = g(y_m) = g\left(\sum_{i=1}^{N} B_i \xi_i^m\right), \tag{2}$$

$$g(y_m) = \mathrm{erf}\left(\frac{y_m}{\sqrt{2}}\right) = \frac{1}{\sqrt{2\pi}} \int_{-y_m}^{y_m} dt \exp\left(-\frac{t^2}{2}\right) \tag{3}$$

where teacher weight vector $\boldsymbol{B} = (B_1, \ldots, B_N)$ is N-dimensional vector, and each element B_i, $i = 1 \sim N$ of teacher weight vector $\boldsymbol{B}$, is drawn from a probability distribution of zero mean and $1/N$ variance. $y_m = \boldsymbol{B} \cdot \boldsymbol{\xi}^m$ is internal potential of the teacher. g is the sigmoid function commonly used in non-linear perceptron. Assuming the thermodynamic limit of $N \to \infty$, statistics of the teacher weight vector are

$$\langle B_i \rangle = 0, \ \left\langle (B_i)^2 \right\rangle = \frac{1}{N}, \ \|\boldsymbol{B}\| = 1. \tag{4}$$

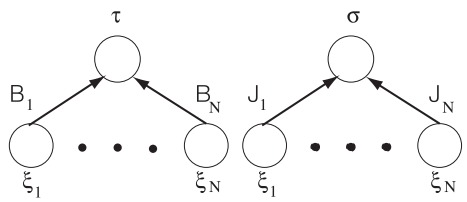

Fig. 1 Network architecture of teacher (left) and student (right) networks, both with the same network structure.

The distribution of the inner potential of the teacher network follows a Gaussian distribution of zero mean and unit variance in the thermodynamic limit.

The perceptron is used as a student network, which has the same architecture as the teacher network. For the sake of analysis, we assume that each element of J_i^0, which is the initial value of the student weight vector $\boldsymbol{J}^m$, is drawn from a probability distribution of zero mean and $1/N$ variance. The norm of the initial student weight vector $\|\boldsymbol{J}^0\|$ is 1 in the thermodynamic limit of $N \to \infty$. Statistics of the student weight vector are

$$\left\langle \boldsymbol{J}_i^0 \right\rangle = 0, \ \left\langle \left(J_i^0\right)^2 \right\rangle = \frac{1}{N}. \tag{5}$$

The student output $\sigma(\boldsymbol{J}, \boldsymbol{\xi}^m)$ for the N-dimensional input $\boldsymbol{\xi}^m$ at the mth learning iteration is

$$\sigma(\boldsymbol{J}, \boldsymbol{\xi}^m) = g(x_m) = g\left(\sum_{i=1}^{N} J_i^m \xi_i^m\right), \tag{6}$$

where $x_m = \boldsymbol{J}^m \cdot \boldsymbol{\xi}^m$ is the internal potential of the student. The distribution of the inner potential of the student network follows a Gaussian distribution of zero mean and variance Q^2 in the thermodynamic limit. Here, $Q^2 = \boldsymbol{J} \cdot \boldsymbol{J}$ is the squared norm of the student weight vector.

Next, we formulate the gradient descent algorithm. We follow Biehl and Schwarze's formulations of the learning [2]. For the possible inputs $\{\boldsymbol{\xi}\}$, we want to train the student network to produce desired outputs $\tau(\boldsymbol{\xi}^m) = \sigma(\boldsymbol{J}, \boldsymbol{\xi}^m)$. We employ the squared error as an error function. The squared error is defined by

$$\epsilon(\boldsymbol{J}, \boldsymbol{\xi}) = \frac{1}{2}[\sigma(\boldsymbol{J}, \boldsymbol{\xi}) - \tau(\boldsymbol{\xi})]^2 \tag{7}$$

At each learning step m, a new uncorrelated input $\boldsymbol{\xi}^m$ is presented. The current student weight vector $\boldsymbol{J}^m$ is updated in the direction of the greatest decrease of $\epsilon(\boldsymbol{J}, \boldsymbol{\xi})$.

$$\begin{aligned} \boldsymbol{J}^{m+1} &= \boldsymbol{J}^m - \frac{\eta}{N} \left[\sigma\left(\boldsymbol{J}^m, \boldsymbol{\xi}^m\right) - \tau\left(\boldsymbol{\xi}^m\right)\right] \nabla_{\boldsymbol{J}} \sigma(\boldsymbol{J}^m, \boldsymbol{\xi}^m) \\ &= \boldsymbol{J}^m + \frac{\eta}{N} \left[g(y_m) - g(x_m)\right] g'(x_m) \boldsymbol{\xi}^m. \end{aligned} \tag{8}$$

Here, η is the so-called learning step size and $g'(x)$ is derivative of the output function $g(x)$. $g'(x) = \sqrt{2/\pi} \exp(-x^2/2)$.

The generalization error of student $\boldsymbol{J}$ is defined as

$$\epsilon_g = \langle \epsilon(\boldsymbol{J}, \boldsymbol{\xi}) \rangle . \tag{9}$$

3. Theoretical Results for Conventional Method

In this section, we introduce some theoretical results given by Biehl and Schwarze [2]. Formulations the same as those in the previous section are used. The generalization error of true gradient descent is given by

$$\epsilon_g = \frac{1}{\pi} \sin^{-1}\left(\frac{1}{2}\right) + \frac{1}{\pi} \sin^{-1}\left(\frac{Q^2}{1+Q^2}\right) - \frac{2}{\pi} \sin^{-1}\left(\frac{R}{\sqrt{2(1+Q^2)}}\right). \tag{10}$$

Here, $R = \boldsymbol{B} \cdot \boldsymbol{J}$ is the overlap of the teacher weight vector $\boldsymbol{B}$

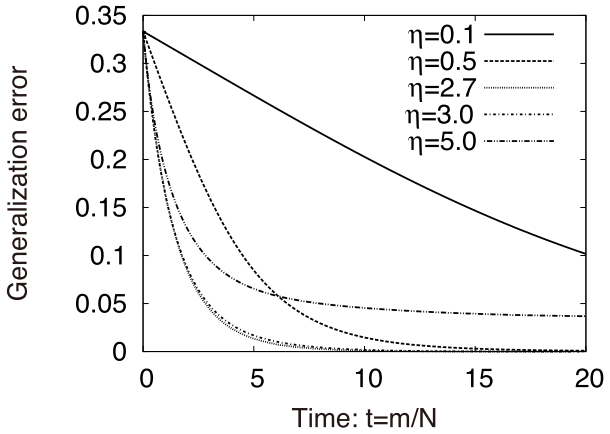

Fig. 2 Generalization error of the perceptron using true gradient for different learning rates. The analytical results are used. All curves are for initial conditions $R(0) = 0$ and $Q(0) = 1$. $\eta = 0.1, 0.5, 2.7, 3.0$ or 5.0 is used.

and student weight vector $\boldsymbol{J}$. ϵ_g is a function of continuous time $t = m/N$ in the thermodynamic limit of $N \to \infty$. These parameters are so-called order parameters that depict dynamics of the learning system. The order parameters obey the following differential equations:

$$\frac{dR}{dt} = \frac{2}{\pi}\frac{\eta}{1+Q^2}\left[\frac{1+Q^2-R^2}{\sqrt{2(1+Q^2)-R^2}} - \frac{R}{\sqrt{1+2Q^2}}\right], \tag{11}$$

$$\frac{dQ^2}{dt} = \frac{4}{\pi}\frac{\eta}{1+Q^2}\left[\frac{R}{\sqrt{2(1+Q^2)-R^2}} - \frac{Q^2}{\sqrt{1+2Q^2}}\right] + \frac{4}{\pi^2}\frac{\eta^2}{\sqrt{1+2Q^2}}\left[\sin^{-1}\left(\frac{Q^2}{1+3Q^2}\right)+\sin^{-1}\left(\frac{1+2(Q^2-R^2)}{2(1+2Q^2-R^2)}\right) - 2\sin^{-1}\left(\frac{R}{\sqrt{2(1+2Q^2-R^2)}\sqrt{1+3Q^2}}\right)\right]. \tag{12}$$

Figure 2 shows the generalization error ϵ_g. In the figure, horizontal axis is time t and the vertical axis is the generalization error ϵ_g. This figure is obtained by solving Eqs. (11) and (12) at each time step, and substituting them into Eq. (10). Initial values are $R(0) = 0$ and $Q(0) = 1$. Learning step size is set to $0.1, 0.5, 2.7, 3.0$ or 5.0. From the figure, the generalization error approaches zero as t increases for small learning step size. Biehl and Schwarze show that if large η is selected, the learning process slows down until a critical learning step size $\eta_c \approx 4.06$ is reached [2]. For $\eta > \eta_c$, the generalization error no longer decays to zero but approaches a value $\epsilon_g > 0$. They also show that there is the optimum learning step size $\eta_{opt} \approx 2.704$. Therefore, the fastest asymptotic decay of ϵ_g is achieved for η_{opt}.

4. Proposed Method and Theory

In this section, we introduce derivative of the output function and show why local property of derivative of the output function causes slow convergence. Then, we propose an acceleration method.

We first investigate the effect of local property of derivative of output function. The variance of inner potential $P(x)$ at the initial is 1, so most of the inner potential is less than 3. Keeping this in mind, we expand $g'(x) = \sqrt{2/\pi}\exp(-x^2/2) \sim \sqrt{2/\pi}(1 - x^2/2 + x^4/8 \cdots)$, and used (1) the first term, (2) the first and second terms for $-\sqrt{2} \le x \le \sqrt{2}$ and 0 otherwise, (3) the first term for $-1 \le x \le 1$ and 0 otherwise, (4) the first term for $-\sqrt{2} \le x \le \sqrt{2}$ and 0 otherwise, (5) the first term for $-2 \le x \le 2$ and 0 otherwise, or (6) the first term for $-3 \le x \le 3$ and 0 otherwise, instead of $g'(x)$. We also used (7) $g'(x) = \sqrt{2/\pi}\exp(-x^2/2)$. Results are shown in **Fig. 3**. Computer simulation results of setting $N = 1000$ and the learning step size $\eta = 0.1$ are used.

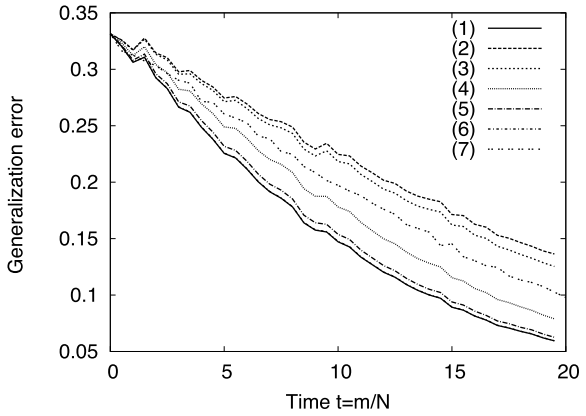

Fig. 3 Comparison of convergence speed using different localities.

In the figure, the top line shows result of (2), and the following lines show results of (3), (7), (4), (5), (6) and (1), respectively. From the results, the generalization error decay slows when (2) or (3) are used. When we reduce locality by expand the region of x as from (4) to (6), the decay speeds are increased. The fastest decay can be achieved by (1) and (6). Therefore, we replace $\sqrt{2/\pi}\exp(-x^2/2)$ with the constant $\sqrt{2/\pi}$.

A better approach might be to use a constant value, "a", instead of "$\sqrt{2/\pi}$" (the first term). We thus modify the learning equation to include a constant term:

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m + \frac{\eta a}{N}\left(\mathrm{erf}\left(\frac{y_m}{\sqrt{2}}\right) - \mathrm{erf}\left(\frac{x_m}{\sqrt{2}}\right)\right)\boldsymbol{\xi}^m = \boldsymbol{J}^m + \frac{\eta a}{N}\delta^m\boldsymbol{\xi}^m. \tag{13}$$

$$\delta^m = \mathrm{erf}\left(\frac{y_m}{\sqrt{2}}\right) - \mathrm{erf}\left(\frac{x_m}{\sqrt{2}}\right) \tag{14}$$

The squared error is defined by Eq. (7) and the generalization error is obtained by average over possible inputs $\{\boldsymbol{\xi}\}$. The generalization error is the same as Eq. (10).

The differential equations that depict behavior of order parameters are given by the next equations. (For derivations, see appendix.) We replace ηa with η' for simplicity.

$$\frac{dR}{dt} = \frac{\eta'}{\sqrt{\pi}}\left(1 - \frac{2R}{\sqrt{2(1+Q^2)}}\right) \tag{15}$$

$$\frac{dQ^2}{dt} = \frac{2\eta'}{\sqrt{\pi}}\left(R - \frac{2Q^2}{\sqrt{2(1+Q^2)}}\right) + \frac{2\eta'^2}{\pi}\left[\sin^{-1}\left(\frac{1}{2}\right)+\sin^{-1}\left(\frac{Q^2}{1+Q^2}\right)-2\sin^{-1}\left(\frac{R}{\sqrt{2(1+Q^2)}}\right)\right] = \frac{2\eta'}{\sqrt{\pi}}\left(R - \frac{2Q^2}{\sqrt{2(1+Q^2)}}\right) + 2\eta'^2\epsilon_g \tag{16}$$

From Eq. (16), the second term of r.h.s. is equal to the generalization error. Thus, this term is zero when the generalization error

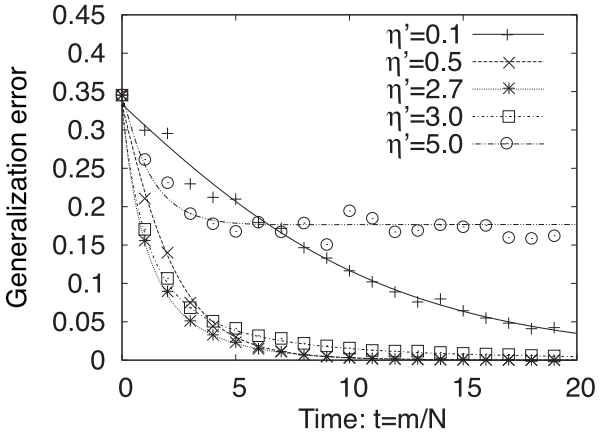

Fig. 4 Comparison of numerical calculation of theoretical results and computer simulations. $\eta' = 0.1, 0.5, 2.7, 3.0$ or 5.0 is used.

converges into zero. These equations form the closed differential equations.

5. Numerical Calculation of Theoretical Results

In the previous section, we derived closed differential equations of the order parameters of the proposed method. In this section, we compare the numerical calculation of theoretical results of the proposed method with those of computer simulations. **Figure 4** shows the results. The learning step size η' is $0.1, 0.5, 2.7, 3.0$, or 5.0. In computer simulations, $N = 1000$, $B_i \sim \mathcal{N}(0, 1/N)$, $J_i^0 \sim \mathcal{N}(0, 1/N)$, and $x_i \sim \mathcal{N}(0, 1)$. Lines show numerical calculation of theoretical results. Symbols show computer simulations: "+" is for $\eta' = 0.1$, "×" is for $\eta' = 0.5$, "∗" is for $\eta' = 2.7$, "□" is for $\eta' = 3.0$, and "○" is for $\eta' = 5.0$. As shown, numerical calculation of theoretical results and computer simulations agreed, validating the theoretical results.

6. Comparison of True Gradient Descent and Proposed Method

In this section, we compare true gradient descent and the proposed method. We used numerical calculation of theoretical results for this purpose. As we introduced in Section 3, the critical learning step size is $\eta_c \approx 4.06$ and the optimum learning step size is $\eta_{opt} \approx 2.7$. Keeping this in mind, we compared the generalization errors for the learning step size $\eta' = 0.1, 0.5, 3.0$, and 5.0. In the following sentences, η is rewritten by η'. **Figure 5** shows the results. In this figure, "(P)" represents the proposed method and "(T)" true gradient descent.

From these figures, in the cases of $\eta' = 0.1$ and 0.5, the generalization error of the proposed method decays faster than that of true gradient descent. The generalization error of true gradient descent decays faster than that of the proposed method when the learning step size is $\eta' = 3.0$. In this case, the generalization error decreases to zero for large t. When $\eta' = 5.0$, the residual error of the proposed method is larger than that of true gradient descent method.

Figure 6 shows the results for $\eta' = \eta'_{opt} = 2.7$. Numerical calculation of theoretical results and computer simulations are used. In the figure, "(T)" means the results obtained by true gradient

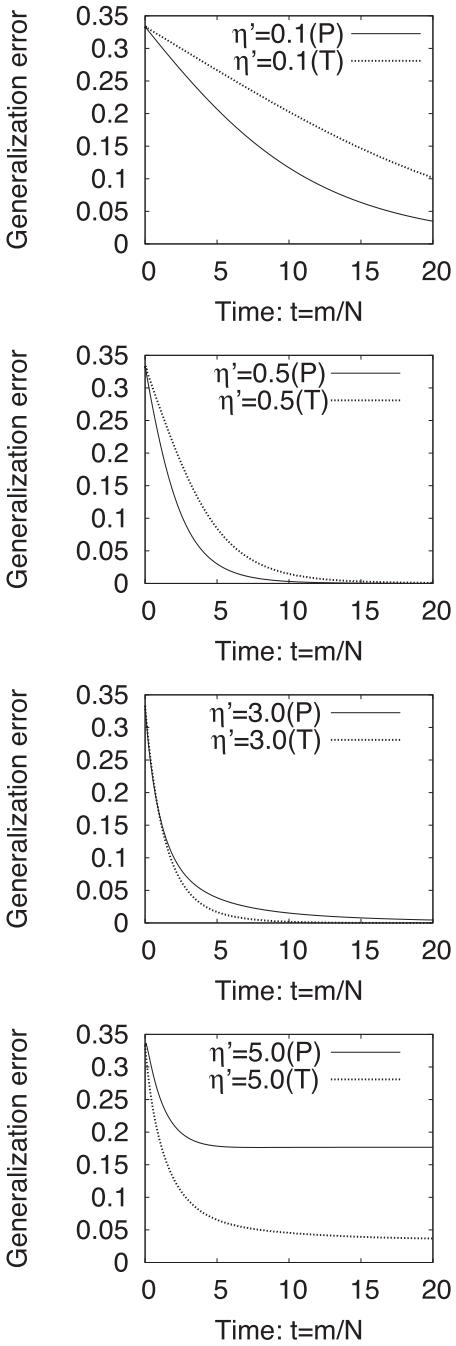

Fig. 5 Comparison of generalization error between true gradient descent and the proposed method. Learning step size η' is 0.1(top), 0.5 (the second), 3.0 (the third), or 5.0 (bottom). "(P)" is for the proposed method, and "(T)" is for true gradient descent. Solid lines represent the proposed method, and broken lines true gradient descent. Numerical calculation of theoretical results are used.

descent, and "(P)" shows the results obtained by the proposed method. Lines show numerical calculation of theoretical results. Symbols show computer simulations: "×" shows results obtained by using true gradient descent. "□" shows the results obtained by using the proposed method. From the figure, numerical calculation of theoretical results agreed with solutions of computer simulations. The generalization errors of both methods decay at the same speed when η'_{opt} is used.

Consequently, the generalization error of proposed method decay faster than true gradient descent when the learning step size is relatively small.

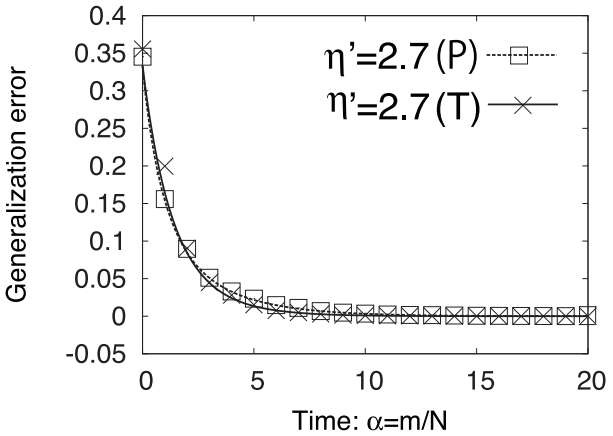

Fig. 6 Comparison of generalization error between proposed method and true gradient descent when $\eta' = 2.7$. Numerical calculation of theoretical results and computer simulation results are used. $N = 1000$ for computer simulations.

7. Conclusions

In this paper, we have proposed a gradient descent algorithm replacing derivative with constant. The idea of replacing $g'(x)$ with the constant value "a" has been inspired by learning equation of the linear perceptron of which the derivative of linear output function $g(x) = x$ is $g'(x) = 1$. We have derived closed order parameter differential equations that depict dynamic behavior of the learning system and solved the generalization error by using numerical calculation of theoretical results. Numerical calculation of theoretical results have been confirmed by computer simulations. From the results, the proposed method can decay faster than the true gradient descent method [2] when learning step size holds $\eta' \leq \eta'_{opt}$. For the case $\eta' > \eta'_{opt}$, in both methods, the generalization error decays slow and residual value of the generalization error remains large. Therefore, when the learning rate η' is smaller than the optimum value, the proposed method can lessen the effect of the estrangement from the optimum value by such a simple substitution. In real problems, the optimum learning step size is not given in advance. Thus, the learning step size must be small. The proposed method is useful in this case.

Acknowledgments We would like to thank Masato Okada and Hayaru Shouno for fruitful discussions.

References

[1] Krogh, A., Hertz, J. and Palmer, R.G.: *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwook City, CA (1991).
[2] Biehl, M. and Schwarze, H.: Learning by on-line gradient descent, *Journal of Physics A: Mathematical and General*, Vol.28, pp.643–656 (1995).
[3] Saad, D. and Solla, S.A.: On-line learning in soft-committee machines, *Physical Review E*, Vol.52, pp.4225–4243 (1995).
[4] Hara, K., Katahira, K., Okanoya, K. and Okada, M.: Statistical Mechanics of On-Line Node-perturbation Learning, *Information Processing Society of Japan, Transactions on Mathematical Modeling and Its Applications*, Vol.4, No.1, pp.72–81 (2011).
[5] Murata, N.: Statistical Study of On-line learning, *On-line Learning in neural networks*, Saad, D. (Ed.), pp.63–92, Cambridge University Press, Cambridge UK (1998).
[6] Fukumizu, K.: A Regularity Condition of the Information Matrix of a Multilayer Perceptron Network, *Neural Networks*, Vol.9, No.5, pp.871–879 (1996).
[7] Rattray, M. and Saad, D.: Incorporating Curvature Information into On-line learning, *On-line Learning in neural networks*, Saad, D. (Ed.), pp.183–207, Cambridge University Press, Cambridge U.K. (1998).
[8] Amari, S.: Natural gradient works efficiently in learning, *Neural Computation*, Vol.10, pp.251–276 (1998).
[9] Kinouchi, O. and Caticha, N.: Optimal generalization in perceptions, *Journal of Physics A: Mathematical and General*, Vol.25, No.23, p.6243 (1992).
[10] Lecun, Y., Simard, P.Y. and Pearlmutler, B.: Automatic Learning Rate Maximization by On-Line Estimation of the Hessian's Eigenvectors, *Advances in Neural Information Processing Systems*, pp.156–163 (1992).
[11] Minsky, M.L. and Papert, S.A.: *Perceptrons*, MIT Press, Cambridge, U.K. (1969).
[12] Williams, C.K.I.: Computation with Infinite Neural Networks, *Neural Computation*, Vol.10, pp.1203–1216 (1998).

Appendix

A.1 Derivation of Differential Equations

The learning equation of the proposed method is

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m + \frac{\eta}{N}\left[g(y_m) - g(x_m)\right]\boldsymbol{\xi}^m \tag{A.1}$$

$$\delta^m = g(y_m) - g(x_m). \tag{A.2}$$

Here, $\boldsymbol{J}^m$ denotes student weight vector, y_m denotes teacher inner potential, and x_m denotes student inner potential at mth interation. By using these equations, the differential equations of the order parameters $Q^2 = \boldsymbol{J}\cdot\boldsymbol{J}$ and $R = \boldsymbol{J}\cdot\boldsymbol{B}$ are given by Ref. [2]

$$\frac{\mathrm{d}Q^2}{\mathrm{d}t} = 2\eta\langle\delta x\rangle + \eta^2\left\langle\delta^2\right\rangle, \tag{A.3}$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \eta\langle\delta y\rangle. \tag{A.4}$$

Then three averages appearing in above equations are

$$\langle\delta x\rangle = \left\langle \mathrm{erf}\left(\frac{y}{\sqrt{2}}\right)x\right\rangle - \left\langle \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)x\right\rangle, \tag{A.5}$$

$$\langle\delta y\rangle = \left\langle \mathrm{erf}\left(\frac{y}{\sqrt{2}}\right)y\right\rangle - \left\langle \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)y\right\rangle, \tag{A.6}$$

$$\left\langle\delta^2\right\rangle = \left\langle \mathrm{erf}\left(\frac{y}{\sqrt{2}}\right)^2\right\rangle + \left\langle \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)^2\right\rangle + 2\left\langle \mathrm{erf}\left(\frac{y}{\sqrt{2}}\right)\mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right\rangle. \tag{A.7}$$

Next, we calculate these averages. We use Williams' results [12] for the above calculations.

$$\frac{1}{(2\pi)^{\frac{d+1}{2}}|\Sigma|^{\frac{1}{2}}}\int \mathrm{erf}(\boldsymbol{u}^T\tilde{\boldsymbol{z}})\mathrm{erf}(\boldsymbol{u}^T\tilde{\boldsymbol{z}}')\exp\left(-\frac{1}{2}\boldsymbol{u}^T\Sigma^{-1}\boldsymbol{u}\right)d\boldsymbol{u} = \frac{2}{\pi}\sin^{-1}\left(\frac{2\tilde{\boldsymbol{z}}^T\Sigma\tilde{\boldsymbol{z}}'}{\sqrt{(1+2\tilde{\boldsymbol{z}}^T\Sigma\tilde{\boldsymbol{z}})}\sqrt{(1+2\tilde{\boldsymbol{z}}'^T\Sigma\tilde{\boldsymbol{z}}')}}\right) \tag{A.8}$$

In calculation of $\left\langle \mathrm{erf}\left(\frac{y}{\sqrt{2}}\right)\mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right\rangle$, we put $\boldsymbol{u} = (x,y)^T$, $\tilde{\boldsymbol{z}} = \left(0, \frac{1}{\sqrt{2}}\right)$, $\tilde{\boldsymbol{z}}' = \left(\frac{1}{\sqrt{2}}, 0\right)$, and $\Sigma = \begin{pmatrix} Q^2 & R \\ R & 1 \end{pmatrix}$. Then we obtain

$$\left\langle \mathrm{erf}\left(\frac{y}{\sqrt{2}}\right)\mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right\rangle = \frac{2}{\pi}\sin^{-1}\left(\frac{R}{\sqrt{2(1+Q^2)}}\right). \tag{A.9}$$

In the same way, we obtain

$$\left\langle \mathrm{erf}\left(\frac{y}{\sqrt{2}}\right)^2\right\rangle = \frac{2}{\pi}\sin^{-1}\left(\frac{1}{2}\right), \tag{A.10}$$

$$\left\langle \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)^2\right\rangle = \frac{2}{\pi}\sin^{-1}\left(\frac{Q^2}{1+Q^2}\right). \tag{A.11}$$

$\langle \delta x \rangle$ can be calculated by using

$$\left\langle \mathrm{erf}(\boldsymbol{u}^T \tilde{\boldsymbol{z}}) \boldsymbol{u}^T \tilde{\boldsymbol{z}}' \right\rangle = \frac{2}{\sqrt{\pi}} \frac{\tilde{\boldsymbol{z}}^T \Sigma \tilde{\boldsymbol{z}}'}{\sqrt{1 + 2\tilde{\boldsymbol{z}}^T \Sigma \tilde{\boldsymbol{z}}}} \tag{A.12}$$

From Eq. (A.5), to calculate $\left\langle \mathrm{erf}\left(\frac{y}{\sqrt{2}}\right) x \right\rangle$, we put $\boldsymbol{u} = (x, y)^T$, $\tilde{\boldsymbol{z}} = \left(0, \frac{1}{\sqrt{2}}\right)$, $\tilde{\boldsymbol{z}}' = (1, 0)$, and $\Sigma = \begin{pmatrix} Q^2 & R \\ R & 1 \end{pmatrix}$. Then

$$\left\langle \mathrm{erf}\left(\frac{y}{\sqrt{2}}\right) x \right\rangle = \frac{R}{\sqrt{\pi}}. \tag{A.13}$$

To calculate $\left\langle \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right) x \right\rangle$, we put the same as the calculation of $\left\langle \mathrm{erf}\left(\frac{y}{\sqrt{2}}\right) x \right\rangle$ except for $\tilde{\boldsymbol{z}} = \left(\frac{1}{\sqrt{2}}, 0\right)$. Then

$$\left\langle \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right) x \right\rangle = \frac{2}{\sqrt{\pi}} \frac{Q^2}{\sqrt{2(1 + Q^2)}}. \tag{A.14}$$

Then, $\langle \delta x \rangle$ is calculated as

$$\langle \delta x \rangle = \frac{1}{\sqrt{\pi}} \left(R - \frac{2Q^2}{\sqrt{2(1 + Q^2)}} \right). \tag{A.15}$$

$\langle \delta y \rangle$ can calculate the same way as $\langle \delta x \rangle$. Then we obtain Eqs. (15) and (16).

Kazuyuki Hara received a B.Eng. and an M.Eng. degrees from Nihon University in 1979 and 1981 respectively and a Ph.D. degree from Kanazawa University in 1997. He was involved in NEC Home Electronics Corporation from 1981 until 1987. He joined to Toyama Polytechnic College in 1987 where he was a lecturer. He joined Tokyo Metropolitan College of Technology in 1998 where he was an associate professor and became a professor in 2005. He became a professor at Nihon University in 2010. His current research interests include statistical mechanics of on-line learning.

Kentaro Katahira received his B.S. degree from Chiba University in 2002 and M.S. and Ph.D. degrees from The University of Tokyo in 2004, 2009, respectively. Currently, he is an associate professor of the Department of Psychology at Nagoya University. His research interests include psychology of learning, decision making and computational neuroscience.