Vol.2012-MPS-91 No.28
Vol.2012-BIO-32 No.28
2012/12/7

Theoretical analysis of learning speed in gradient descent algorithm replacing derivative with constant

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

Abstract: In on-line gradient descent learning, the local property of the derivative term of the output can slow convergence. 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 replace the derivative term with a constant in this paper and showed that this greatly increases convergence speed under some conditions. The proposed algorithm inspired by linear perceptron learning, and it can avoid locality of the derivative term. We derived the closed deterministic differential equations by using a statistical mechanics method and show validity of analytical solutions by comparing that of computer simulations.

1. Introduction

Learning in neural networks can be formulated as optimization of an objective function that quantifies the system's performance. A important property of feed-forward network is their ability to learn a rule from examples. A statistical mechanics have 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 is 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 to be efficient in terms of the number of examples needed for good generalization, however, it is costly. On-line learning is a popular method for learning multi-layer feed-forward neural networks, especially for large system 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 a 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 amount of works that correspond to acceleration of learning process[6], [7], [8]. The main problem of slow learning is *plateau* which occurs in learning process using gradient descent algorithm. On the other hand, local property of derivative of the output function is considered. In gradient descent learning, the parameters are updated to the direction of the steepest descent of the objective function. To calculate the direction, derivative of the output function is used. When the output function is linear, derivative is 1 and is not a function of the inner potential of output unit. However, the output function is non-linear, as like a sigmoidal function, the derivative becomes Gaussian function, and is a localized function of inner potential of the output unit. Gaussian function has value around zero and has very small value at the sides. So update of the parameter becomes very small for large absolute value of inner potential and this causes slow convergence. We scope accelerates the learning process on modifying derivative of the output function while conventional methods scope on 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 proposed method by using statistical mechanics methods. The configuration of this paper is as follows: We formulate the networks, the input, the output function and the gradient descent algorithm in Sec. 2. We employ teacher-student formulation, and is also introduced in the section. In Sec. 3, we introduce some theoretical results of conventional gradient descent method[2]. Then we propose an accelerate method, and derive closed differential equations which depict dynamical behavior of the system in Sec. 4. In Sec. 5, we derive the generalization error of proposed method, and in Sec. 6, we compare the error with that of conventional method[2]. We summarize the results and conclude in Sec. 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 re-

[1] College of Industrial Technology, Nihon University, 1-2-1 Izumi-cho, Narashino, Chiba, 275–8575 Japan
[2] Center for Evolutionary Cognitive Sciences, The University of Tokyo, 3-8-1 Komaba, Meguro-ku, Tokyo 113-8654 Japan
[3] Brain Science Institute, RIKEN
[a)] hara.kazuyuki@nihon-u.ac.jp
[b)] katahira@ecs.c.u-tokyo.ac.jp

ceive N-dimensional input $\boldsymbol{\xi}^m = (\xi_1^m, \ldots, \xi_N^m)$ at the m-th learning iteration as shown in Fig. 1. Here, we assume the existence of a teacher network vector $\boldsymbol{B}$ that produces desired outputs, so the teacher output τ is a target of the student output σ. This is called teacher-student formulation.

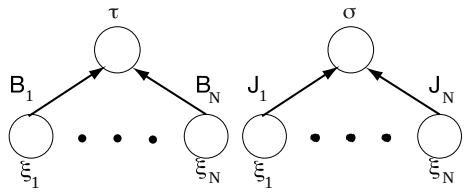

Fig. 1 Network structure of teacher (left) and student (right) networks, both with the same network structure

The teacher network shown in Fig. 1 has N inputs and an output, and is identical to a linear perceptrons. The learning iteration m is ignored in the figure. Student network has the same architecture as the teacher. $\boldsymbol{B}$ denotes the weight matrix of a teacher network with N elements. $\boldsymbol{J}^m$ denotes the weight matrix of a student network with N elements at m-th the learning iteration, the same as the teacher network. We also assume that the elements ξ_i^m of independently drawn input $\boldsymbol{\xi}^m$ are uncorrelated random variables with zero mean and variance 1; that is, the i-th 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 vectors $\{\boldsymbol{B}\}$ is fixed in the learning process. The output of the teacher τ_m for N-dimensional input $\boldsymbol{\xi}^m$ at the m-th learning iteration is

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

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

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

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

The distribution of the output 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, and the student network 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 vector $\boldsymbol{J}^\mu$, is drawn from a probability distribution of zero mean and $1/N$ variance. The norm of the initial student weight vector $\|\boldsymbol{J}_i^0\|$ is 1 in the thermodynamic limit of $N \to \infty$. Statistics of the k-th student weight vector are

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

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

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

$$\mathrm{g}(x) = \mathrm{erf}\left(\frac{x}{\sqrt{2}} \right) = \frac{1}{\sqrt{2\pi}} \int_{-x}^{x} dt \exp\left(-\frac{t^2}{2} \right) \tag{7}$$

$x_m = \boldsymbol{J}^m \cdot \boldsymbol{\xi}^m$ is internal potential of the student. The distribution of the output of the teacher network follows a Gaussian distribution of zero mean and unit variance in the thermodynamic limit.

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

$$\varepsilon(\boldsymbol{J}, \boldsymbol{\xi}) = \frac{1}{2} [\sigma(\boldsymbol{J}, \boldsymbol{\xi}) - \tau(\boldsymbol{\xi})]^2 \tag{8}$$

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

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

Blanket $\langle \cdot \rangle$ denotes average over possible inputs. At each learning step m, a new uncorrelated input ξ^m is presented, the current student weight vector $\boldsymbol{J}^m$ is updated by

$$\begin{aligned} \boldsymbol{J}^{m+1} &= \boldsymbol{J}^m - \frac{\eta}{N} \left[\sigma(\boldsymbol{J}^m \cdot \boldsymbol{\xi}^m) - \tau(\boldsymbol{\xi}^m) \right] \nabla_{\boldsymbol{J}} \sigma(\boldsymbol{J}^m, \boldsymbol{\xi}^m) \\ &= \boldsymbol{J}^m + \frac{\eta}{N} \left[\mathrm{g}(y_m) - \mathrm{g}(x_m) \right] \mathrm{g}'(x_m) \boldsymbol{\xi}^m. \end{aligned} \tag{10}$$

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

3. Theoretical Results for conventional method

In this section, we introduce some theoretical results given by Biehl et. al.[2]. The same formulations as previous section are used. The generalization error of true gradient descent is given by

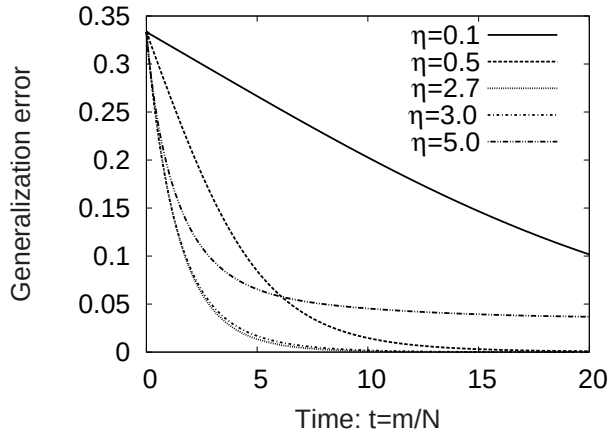

Fig. 2 Generalization error of the perceptron using true gradient for different learning rates.The analytical results are used.All curves are for initial conditions $R(0) = 0$ and $Q(0) = 1$. $\eta = 0.1, 0.5, 2.7, 3.0$ or 5.0 is used.

$$\epsilon_g = \frac{1}{\pi}\sin^{-1}\left(\frac{1}{2}\right) + \frac{1}{\pi}\sin^{-1}\left(\frac{Q^2}{1+Q^2}\right) - \frac{2}{\pi}\sin^{-1}\left(\frac{R}{\sqrt{2(1+Q^2)}}\right). \tag{11}$$

Here, $Q^2 = \boldsymbol{J}\cdot\boldsymbol{J}$ is the norm of the student weight vector, and $R = \boldsymbol{B}\cdot\boldsymbol{J}$ is the overlap of the teacher weight vector $\boldsymbol{B}$ and student weight vector $\boldsymbol{J}$. ϵ_g is function of continuous time $t = m/N$ in the thermodynamic limit of $N \to \infty$. These parameters so-called order parameters that depicts dynamics of the learning system. The order parameters obey the following differential equations,

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

$$\begin{aligned}\frac{dQ^2}{dt} =& \frac{4}{\pi}\frac{\eta}{1+Q^2}\left[\frac{R}{\sqrt{2(1+Q^2)-R^2}} - \frac{Q^2}{\sqrt{1+2Q^2}}\right] \\ &+ \frac{4}{\pi^2}\frac{\eta^2}{\sqrt{1+2Q^2}}\left[\sin^{-1}\left(\frac{Q^2}{1+3Q^2}\right) + \sin^{-1}\left(\frac{1+2(Q^2-R^2)}{2(1+2Q^2-R^2)}\right)\right. \\ &\left. - 2\sin^{-1}\left(\frac{R}{\sqrt{2(1+2Q^2-R^2)}\sqrt{1+3Q^2}}\right)\right]\end{aligned} \tag{13}$$

Here, $t = m/N$ is continuous time in the limit of $N \to \infty$.

Figure2 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 obtained by solving Eqs. (12) and (13) at each time step, then substituting into Eq.(11). 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 to zero with increasing t for small learning rate. Biehl et al. shows 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 does not decay to zero any longer but approaches a value $\epsilon_g > 0$. He also show that there is the optimum learning step size $\eta_{opt} \approx 2.704$. Therefore, the fastest asymptotic decay of ϵ_g is achived 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 causes slow convergence. Then, we propose acceleration method.

Figure 3 shows shape of $g'(x)$ (derivative of output function $g(x)$) respect to x where x is inner potential of the student. We briefly explain the relation between convergence speed and $g'(x)$. As shown in Fig. 3, $g'(x)$ decays quickly along x. As we explained in the previous section. the distribution of inner potential $P(x)$ follows Gaussian distribution of mean zero and unit variance in thermodynamic limit of $N \to \infty$. Then, $g'(x)$ for non-zero x are very small. Therefore, from Eq. (10), update of the student weight is very small and the convergence speed becomes slow.

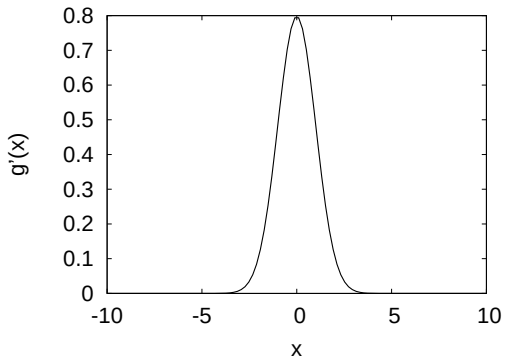

Fig. 3 Shape of $g'(x)$ as a function of inner potential of student.We used $g(x) = \mathrm{erf}(x/\sqrt{2})$.

We expand $g'(x) = \exp(-x^2/\sqrt{2}) \sim 1 - x^2/2 + x^4/8\cdots$ and use the first term. When the first term is constant, the update for non-zero x becomes larger. A better approach might be to use a constant value, "a", instead of "1" (the first term). We thus modify the learning equation to include a constant term:

$$\begin{aligned}\boldsymbol{J}^{m+1} =& \boldsymbol{J}^m + \frac{\eta a}{N}\left(\mathrm{erf}\left(\frac{y_m}{\sqrt{2}}\right) - \mathrm{erf}\left(\frac{x_m}{\sqrt{2}}\right) - n\right)\boldsymbol{\xi}^m \\ =& \boldsymbol{J}^m + \frac{\eta a}{N}\delta\boldsymbol{\xi}.\end{aligned} \tag{14}$$

We replace η' with ηa for simplicity.

The squared error is defined as follows:

$$\epsilon = \frac{1}{2}\left(g(y_m) - g(x_m)\right)^2, \tag{15}$$

and the generalization error is obtained by average over possible inputs $\boldsymbol{\xi}$. The generalization error is as the same as Eq. (11).

The differential equations depict behavior of order parameters are given by next equations. (For derivations, see appendix.)

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

$$\begin{aligned}\frac{dQ^2}{dt} =& \frac{2\eta'}{\sqrt{\pi}}\left(R - \frac{2Q^2}{\sqrt{2(1+Q^2)}}\right) \\ &+ \frac{2\eta'^2}{\pi}\left[\sin^{-1}\left(\frac{1}{2}\right) + \sin^{-1}\left(\frac{Q^2}{1+Q^2}\right) - 2\sin^{-1}\left(\frac{R}{\sqrt{2(1+Q^2)}}\right)\right] \\ =& \frac{2\eta'}{\sqrt{\pi}}\left(R - \frac{2Q^2}{\sqrt{2(1+Q^2)}}\right) + 2\eta'^2\epsilon_g\end{aligned} \tag{17}$$

These equations form the closed differential equations.

5. Computer Calculation of Analytical Results

In the previous section, we derived closed differential equations of the order parameters of the proposed method. In this section, we compare the analytical solutions of proposed method with that of computer simulations. Figure 4 shows the results. The learning step size η' is $0.1, 0.5, 2.7$ or 5.0. In computer simulations, $N = 1000$, $\|\boldsymbol{B}\| = 1/\sqrt{N}$, $\|\boldsymbol{J}^0\| = 1/\sqrt{N}$ and $\|\boldsymbol{x}\| = 1$. Lines show analytical solutions. 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, analytical results and computer simulations are agreed, so validity of the analytical results is shown.

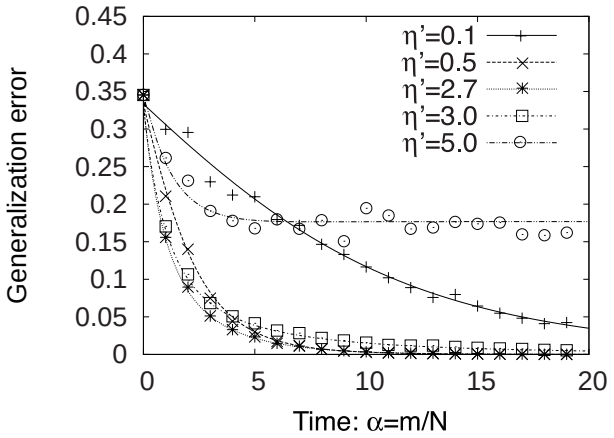

Fig. 4 Comparison of analytical solutions and that of computer simulations. $\eta' = 0.1, 0.5, 2.7, 3.0$ or 5.0 is used.

6. Comparison of True Gradient Descent and Proposed Method

In this section, we compare true gradient descent and proposed method. We used analytical solutions for this purpose. As we introduced, critical learning step size is $\eta_c \approx 4.06$, maximum learning step size is $\eta_d \approx 9.2$ and the optimum learning step size is $\eta_{opt} \approx 2.7$. We keep these in mind, we compared the generalizations for the learning step size $\eta = 0.1, 0.5, 3.0$, and 5.0. Figure 5 shows the results. In these figures, "(P)" is for the proposed method, and "(T)" is for true gradient descent.

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

Figure 6 shows the results for $\eta' = \eta_{opt} = 2.7$. Analytical solutions and computer simulations are used. In the figure, "True" means the results obtained by true gradient descent, and "Proposed" shows the results obtained by proposed method. "(th)" means theoretical results, and "(sim)" means computer simulation results. Lines show analytical solutions. "×" shows results obtained by computer simulation using true gradient descent. "□" shows the results obtained by computer simulation using proposed methods. From the figure, analytical solutions agreed with computer simulations. The generalization error decays the same speed for both methods using η_{opt}.

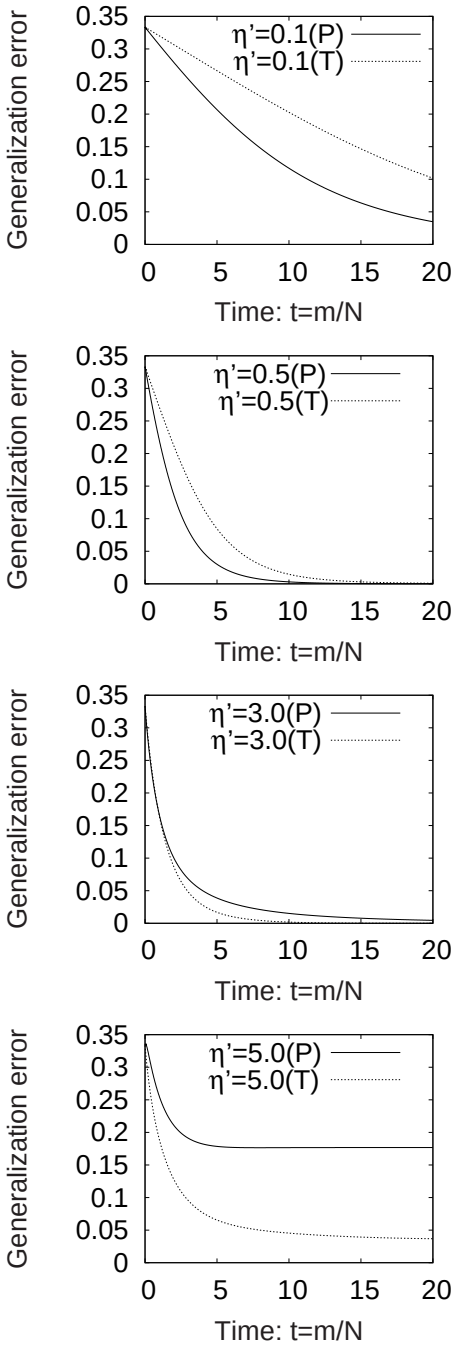

Fig. 5 Comparison of generalization error between true gradient descent and proposed method. Learning step size η' is 0.1(left top), 0.5 (right top), 3.0 (left bottom) or 5.0 (right bottom). "(P)" is for the proposed method, and "(T)" is for true gradient descent. Solid line is proposed method, and break line is true gradient descent. Analytical solutions are used.

Consequently, when the learning step size is $\eta' < \eta_{opt}$, the generalization of proposed method decays faster than true gradient descent, and the generalization error of true gradient descent decays faster than that of proposed method when $\eta' > \eta_{opt}$ is hold.

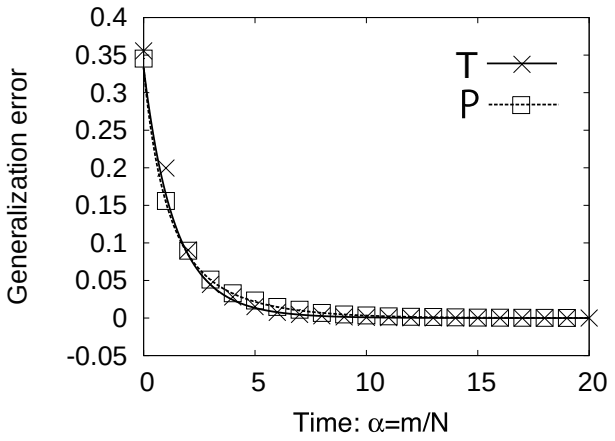

Fig. 6 Comparison of asymptotic property between proposed method and true gradient descent. Analytical solutions and computer simulation results are used. $N = 1000$ for computer simulations.

7. Conclusions

In this paper, we have proposed simple gradient descent method using a constant value "a" instead of derivative $\mathrm{g}'(x)$ of output function $\mathrm{g}(x)$. The idea of replacing $\mathrm{g}'(x)$ with the constant value "a" inspired by learning equation of the linear perceptron. The derivative of linear output function $\mathrm{g}(x) = x$ is $\mathrm{g}'(x) = 1$. We have derived closed order parameter differential equations which depicts dynamic behavior of the learning system, and solved the generalization error by using analytical solutions. Analytical solutions have confirmed by computer simulations. From the results, proposed method can decay faster than true gradient descent method [2] when learning step size holds $\eta' < \eta_{opt}$. For the case $\eta' > \eta_{opt}$, proposed method decays slower and resident error becomes larger than those of true gradient descent method.

Appendix

A.1 Derivation of Differential equations

The learning equation of the proposed method is

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

$$\delta^m = \frac{1}{N}\left[\mathrm{g}(y_m) - \mathrm{g}(x_m)\right]. \tag{A.2}$$

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[2]

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

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

Here, $\langle\cdot\rangle$ denotes average over possible input $\boldsymbol{\xi}$. We define $g(x)$ as $g(x) = \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right) = \frac{1}{\sqrt{2\pi}}\int_{-x}^{x} dt\exp\left(-\frac{t^2}{2}\right)$, then three averages appears 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's result in [12] for the above calculations.

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

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 $\boldsymbol{u} = (x, y)^T$, $\tilde{\boldsymbol{z}} = (0, 1/\sqrt{2})$, $\tilde{\boldsymbol{z}}' = (1/\sqrt{2}, 0)$, $\Sigma = \begin{pmatrix} Q^2 & R \\ R & 1 \end{pmatrix}$, then we get

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

As the same way, we get

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

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

$\langle\delta x\rangle$ and $\langle\delta y\rangle$ can be calculated similar way to the above. Then we get Eqs. (16) and (17).

References

[1] A. Krogh, J. Hertz and R.G. Palmer, *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwook City, CA, 1991.

[2] M. Biehl and H. Schwarze, "Learning by on-line gradient descent", Journal of Physics A: Mathematical and General Physics, **28**, 643–656, 1995.

[3] D. Saad and S. A. Solla, "On-line learning in soft-committee machines", Physical Review E, **52**, pp. 4225-4243 1995.

[4] K. Hara, K. Katahira, K. Okanoya and M. Okada, "Statistical Mechanics of On-Line Node-perturbation Learning", Information Processing Society of Japan, Transactions on Mathematical Modeling and Its Applications, **4**, no. 1, pp. 72-81, 2011.

[5] N. Murata, "Statistical Study of On-line learning" in D. Saad. (Ed) *On-line Learning in neural networks.* (pp. 63-92). Cambridge University Press, Cambridge UK, 1998.

[6] K. Fukumizu, "A Regularity Condition of the Information Matrix of a Multilayer Perceptron Network", Neural Networks, **9**, 5, pp. 871-879 1996 .

[7] M. Rattray and D. Saad, "Incorporating Curvature Information into On-line learning", in D. Saad. (Ed) *On-line Learning in neural networks.* (pp. 183–207). Cambridge University Press, Cambridge UK, 1998.

[8] S. Amari, "Natural gradient works efficiently in learning", Neural Computation, **10**, pp. 251-276 1998.

[9] O Kinouchi and N Caticha, "Optimal generalization in perceptions", Journal of Physics A: mathematical and General, **25**, no. 23, 6243, 1992.
[10] Y. Lecun, P. Y. Simard and B. Pearlmutler, "Automatic Learning Rate Maximization by On-Line Estimation of the Hessian's Eigenvectors", Advances in Neural Information Processing Systems, 156-163, 1992.
[11] S. E. Fahlman, "An empirical study of learning speed in backpropagation networks", http://repository.cmu.edu/compsci/1800, 1988.
[12] C. K. I. Williams, "Computation with Infinite Neural Networks", Neural Computation, **10**. 1203-1216 1998.