IPSJ Transactions on Mathematical Modeling and Its Applications Vol.13 No.2 61–68 (Aug. 2020)

Regular Paper

Node-perturbation Learning Applied for Soft-committee Machine

Kazuyuki Hara[1,a)] Kentaro Katahira[2,b)] Masato Okada[3,4,c)]

Received: November 13, 2019, Revised: January 6, 2020,
Accepted: February 20, 2020

Abstract: Node-perturbation learning is an online stochastic gradient descent method for neural networks. It estimates the gradient of the error surface by calculating the change in error between the perturbed output and the non-perturbed output. Node-perturbation can be applied to problems where the objective function is not defined. Node-perturbation learning is applied to only a simple perceptrons, so we explore the application of node perturbation learning to a multilayer neural network called a soft committee machine and analyze the dynamic properties of the learning process. We conduct computer analysis to show the validity of the proposed method.

Keywords: node-perturbation learning, soft committee machine, on-line learning, perturbation, generalization error, statistical mechanics method

1. Introduction

Supervised learning in neural networks [1] can be formulated as an optimization of an objective function that quantifies the system's performance. The optimization is carried out by calculating the gradient of the objective function explicitly and updating the parameters by a small step in the direction of the locally greatest improvement. However, computing a direct gradient to follow can be problematic. For instance, reinforcement learning has no explicit form of the objective function, so we cannot calculate a gradient for it.

As a solution to this problem, Williams et al. [2] proposed node-perturbation learning [3] (NP learning) based on the online stochastic gradient method for a simple perceptron [5]. NP learning estimates the gradient by examining the change in the scalar objective value when noise is added to the output of the network noise. If the objective value becomes smaller when noise is added to the network output, the weight vector changes in the direction of the noise. As a result, NP learning can be formulated as a reinforcement learning in which all the weight vectors are updated using a scalar reward, instead of a target vector as in the gradient method. Here, Werfel et al. calculated the learning curve of NP learning for linear perceptron by using the ensemble mean [4]. For more complex problems, i.e., linearly not separable problems, the multilayer architecture is required, and NP learning for multilayer network is an open problem in the machine learning research.

In the current paper, our opportunity is to explore the implement methods of NP learning in the multilayer network, and to show the learning behavior of the NP learning for the multilayer network. We also want to give insights for NP learning for multilayer networks. We implement NP learning in two ways, i.e., by adding perturbation noise to the hidden layer or by adding noise to the output layer. We formulate NP learning for soft-committee machine by using the statistical mechanical method [6], [7], [8], [9], and analyzed the behavior of the learning equations of adding noise to output unit or adding noise to hidden unit, based on the signal-to-noise analysis. The generalization error is given by using the order parameters $R_{kk'}$s and $Q_{k'l'}$ those are given by time course of the student weight vectors. A soft-committee machine, that is used as a multilayer network in this paper, has a simple network structure; however, it suffers from plateau phenomena and symmetry breaking. Plateau phenomena and symmetry breaking in two implementations are shown and compared in the generalization error. We analyze reasons of worse generalization error in two implementation and give suggestions to improve the generalization. We also compared NP learning with noisy learning. These our findings give insights for the NP learning for multilayer networks.

2. Previous Study

Previous study in NP learning, Williams et al. [2] proposed node-perturbation learning [3] based on the online stochastic gradient method. In this paper, the simple perceptron is used. Werfel et al. calculated the learning curve of NP learning for linear simple perceptron by using the ensemble mean [4]. Cho et al. [14] calculated the learning curve of NP learning without noiseless

[1] College of Industrial Technology, Nihon University, Narashino, Chiba 275–8575, Japan
[2] Graduate School of Informatics, Nagoya University, Nagoya, Aichi 464–8601, Japan
[3] Graduate School of Sciences, The University of Tokyo, Bunkyo, Tokyo 113–0033, Japan
[4] Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa, Chiba, 277–8561, Japan
[a)] hara.kazuyuki@nihon-u.ac.jp
[b)] katahira.kentaro@b.mbox.nagoya-u.ac.jp
[c)] okada@edu.k.u-tokyo.ac.jp

baseline by calculating ensemble mean.

We calculated the dynamics of NP learning applied for linear perceptrons [13] by using statistical mechanics method. We also calculated dynamics of NP learning applied for non-linear perceptrons [15] by using statistical mechanics method.

Prevous study in analysis of the soft-committee machine by using the statistical mechanics method, Saad [8] and Biehl [6], [7] independently calculated the dynamics of gradient descent learning for soft-committee machine.

In this paper, we apply NP learning for soft-committee machine. This is the first paper that apply the NP learning to the multilayer networks. In NP learning applied for non-linear perceptrons, the error is calculated for sum of the squared error of each output unit. As shown in Eq. (18), the error of soft-committee machine is calculated for the squared error of total of outputs. So, in soft-committee machine, we must consider cross-correlation of the outputs.

3. Formulation

Here, we formulate the teacher and student networks and derive a learning rule for applying the NP learning algorithm to a soft-committee machine. Supervised-learning and online-learning settings are assumed.

3.1 Model

First, we introduce the teacher-student formulation that is used in the statistical mechanical method. Then, we use their formulations to build the NP learning algorithm. The teacher network (teacher) generates the target of the student network (student) for a given input. By introducing a teacher network, we can directly measure the similarity of the student weight vector to the teacher weight vector.

Figure 1 shows the teacher and student, which are soft-committee machines with N inputs and one linear output. The teacher and student receive the same input vector $\boldsymbol{x}^{(m)}$ at the mth learning iteration. The teacher output $t^{(m)}$ is used as the scalar target for $\boldsymbol{x}^{(m)}$. Note that the iteration m is not shown in the figure. The teacher includes K hidden units, while the student includes M hidden units. The inner potential of hidden units of the teacher and student are the inner products of the input vector and the weight vector from the input to hidden layer. In the following part of the paper, the weight vector from the input to hidden layer is called the weight vector. The activation function of the hidden unit output is a non-linear function and the function is applied to the inner-potential of hidden unit. All weights from the hidden layer to the output layer for the teacher and student soft-committee machines are set to one [8]. The network output is determined by majority vote of hidden unit outputs.

Each element $x_j^{(m)}$, $j = 1 \sim N$ of the input vector $\boldsymbol{x}^{(m)}$ is drawn from a probability distribution $P(x_j)$ with zero mean and unit variance. The statistics of $\boldsymbol{x}^{(m)}$ in the thermodynamic limit, $N \to \infty$, are

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

Here, $\langle \cdots \rangle$ means the average of all elements, and $\|\cdot\|$ means the norm of a vector.

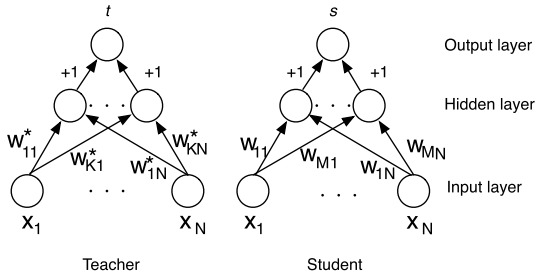


Fig. 1 Structure of Teacher and student networks.

The teacher is not to the object of the learning. Thus, the teacher weight vectors $\boldsymbol{w}_k^*$, $k = 1 \sim K$ are not updated during the learning process. The kth weight vector $\boldsymbol{w}_k^*$ is an N-dimensional vector, and each element w_{kj}^*, $j = 1 \sim N$ is drawn from a probability distribution $P(w_{kj}^*)$ with mean zero and variance $1/N$. The statistics of the jth element of the kth weight vector for the teacher w_{kj}^* in the thermodynamic limit, $N \to \infty$, are

$$\left\langle w_{kj}^* \right\rangle = 0, \ \left\langle (w_{kj}^*)^2 \right\rangle = \frac{1}{N}, \ \|\boldsymbol{w}_k^*\| = 1. \tag{2}$$

The inner potential of the kth hidden unit for $\boldsymbol{x}^{(m)}$ is written as

$$d_k^{(m)} = \sum_{j=1}^{N} w_{kj}^* x_j^{(m)} = \boldsymbol{w}_k^* \cdot \boldsymbol{x}^{(m)}, \tag{3}$$

The inner potential of the hidden unit d_k in the thermodynamic limit, $N \to \infty$, obeys a Gaussian distribution with zero mean and unit variance. The kth hidden unit output is denoted as $g(d_k^{(m)})$ where $g(\cdot)$ is a non-linear activation function. The output of teacher at the mth iteration $t^{(m)}$ is calculated as

$$t^{(m)} = \sum_{k=1}^{K} g(d_k^{(m)}) \tag{4}$$

The student consists of M hidden units. To ease the analysis, we assume that each element of the initial weight vector from the jth element of k'th weight vector $w_{k'j}^{(0)}$ is drawn from a probability distribution $P(w_{k'j})$ with mean zero and variance $1/N$. The statistics of the jth element of the k'th weight vector $w_{k'j}$ of the student in the thermodynamic limit, $N \to \infty$, are

$$\left\langle w_{k'j}^{(0)} \right\rangle = 0, \ \left\langle (w_{k'j}^{(0)})^2 \right\rangle = \frac{1}{N}, \ \|\boldsymbol{w}_{k'}^{(0)}\| = 1. \tag{5}$$

The inner potential of the k'th hidden unit for input $\boldsymbol{x}^{(m)}$ at the mth iteration is

$$y_{k'}^{(m)} = \sum_{j=1}^{N} w_{k'j}^{(m)} x_j^{(m)} = \boldsymbol{w}_{k'}^{(m)} \cdot \boldsymbol{x}^{(m)}. \tag{6}$$

The distribution of the inner potential $y_{k'}$ in the thermodynamic limit, $N \to \infty$, becomes a Gaussian with mean zero and variance $Q_{k'k'}^2$ Here, $Q_{k'k'}^2 = \boldsymbol{w}_{k'} \cdot \boldsymbol{w}_{k'}$. The output of the k'th hidden unit of the student is denoted as $g(y_{k'}^{(m)})$ where $g(\cdot)$ is a non-linear activation function. The output of the student at the mth iteration $s^{(m)}$ is calculated as

$$s^{(m)} = \sum_{k'=1}^{M} g(y_{k'}) \tag{7}$$

The weight vector $\boldsymbol{w}_{k'}$ is updated by using the stochastic gradient descent algorithm. Note that the weights from the hidden layer to the output of the student are fixed to +1 and are not objects of learning.

3.2 Node-perturbation Learning Applied for Soft-committee Machine

Here, we describe the node-perturbation learning (NP learning) algorithm that is applied for the soft-committee machine. First, we formulate the NP learning. The objective function is the squared error. The squared error at the mth learning iteration is defined as

$$E^{(m)} = \frac{1}{2}(t^{(m)} - s^{(m)})^2 = \frac{1}{2}\left(\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{M} g(y_{l'}^{(m)})\right)^2. \quad (8)$$

The weight vector of the student is updated in the direction of the noise if the squared error becomes smaller when noise is added to the network output, otherwise not updated. Accordingly, the learning equation is defined as

$$w_{k'j}^{(m+1)} = w_{k'j}^{(m)} - \frac{\eta}{N}(E_\xi^{(m)} - E^{(m)})e_{k'j} \quad (9)$$

Here, $e_{k'j}$ is defined as $e_{k'j} \equiv \partial \ln f_k / \partial w_{k'j}$ and is called the characteristic eligibility of $w_{k'j}$ and f_k is the probability mass function determining the value of student output. E_ξ is the squared error when the noise is added to the network output. NP learning can be accomplished in two ways: (1) by adding noise to the output layer or (2) by adding noise to the hidden layer.

3.2.1 Adding Noise to the Output Layer

The squared error when noise is added to the output layer (the output NP case) is defined as

$$\begin{aligned} E_\xi^{(m)} &= \frac{1}{2}\left(t^{(m)} - (s^{(m)} + \xi^{(m)})\right)^2 \\ &= \frac{1}{2}\left(\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{M} g(y_{l'}^{(m)}) - (\xi^{(m)})^2\right)^2. \end{aligned} \quad (10)$$

Here, the added noise ξ is drawn from the Gaussian distribution with mean zero and variance σ_ξ^2. $E_\xi^{(m)} - E^{(m)}$ is calculated as

$$\begin{aligned} &E_\xi^{(m)} - E^{(m)} \\ &= -\frac{1}{2}\left\{2\xi^{(m)}\left(\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{M} g(y_{l'}^{(m)})\right) - (\xi^{(m)})^2\right\}. \end{aligned} \quad (11)$$

In Eq. (11), $\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{M} g(y_{l'})^{(m)}$ is the gradient of the squared error. Although NP learning doesn't use the gradient explicitly, as shown in Eq. (9), the gradient information is implicitly included in Eq. (11). Equation (11) becomes negative when $E_\xi < E$, and the weight vector is updated in the direction of the noise ξ [2]. Independent noise is added to the output layer unit at every learning iteration. The iteration number m on the noise ξ is omitted. Accordingly, the learning equation of the output NP case is defined as

$$\begin{aligned} w_{k'j}^{(m+1)} &= w_{k'j}^{(m)} + \frac{\eta}{2N\sigma_\xi^2} \\ &\cdot\left\{2(\xi)^2\left(\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{M} g(y_{l'}^{(m)})\right) - (\xi)^3\right\} g'(y_{k'}^{(m)}) x_j^{(m)} \end{aligned} \quad (12)$$

In Eq. (12), the derivative at each hidden unit $g'(y_{k'})$ is independent from those of other hidden units, then the weight $w_{k'j}$ is updated independently.

3.2.2 Adding Noise to the Hidden Layer

As above, the noise $\xi_{k'}$ added to each hidden unit is drawn from a probability distribution with mean zero and variance σ_ξ^2. The squared error in this case is defined as

$$E_\xi^{(m)} = \frac{1}{2}\left(\sum_{l=1}^{K} g(d_l) - \sum_{l'=1}^{M}(g(y_{l'}) + \xi_{l'})\right)^2. \quad (13)$$

Here, $E_\xi^{(m)} - E^{(m)}$ is calculated as

$$\begin{aligned} E_\xi^{(m)} - E^{(m)} = &-\frac{1}{2}\left\{2\sum_{l'=1}^{M}\xi_{k'}\left(\sum_{l=1}^{K} g(d_l) - \sum_{l'=1}^{M} g(y_{l'})\right)\right. \\ &\left. -\left(\sum_{l'=1}^{M}\xi_{l'}\right)^2\right\} \end{aligned} \quad (14)$$

Similar to the above expression for the noise added to the output layer, $\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{M} g(y_{l'})^{(m)}$ is the gradient of the squared error. As well, Eq. (14) becomes negative when $E_\xi < E$, and the weight vector is updated in the direction of $\xi_{l'}$. Identical distribution is used for the noise. The iteration number m on the noise ξ is omitted. The learning equation for the case of adding noise to the hidden layer (hidden NP case) is defined as

$$\begin{aligned} w_{k'j}^{(m+1)} = w_{k'j}^{(m)} + \frac{\eta}{2N}&\left\{2\sum_{l'=1}^{M}\xi_{l'}\left(\sum_{l=1}^{K} g(d_l) - \sum_{l'=1}^{M} g(y_{l'})\right)\right. \\ &\left. -\left(\sum_{l'=1}^{M}\xi_{l'}\right)^2\right\} g'(y_{k'})\frac{\xi_{k'}}{\sigma_\xi^2} x_j \end{aligned} \quad (15)$$

We separate the noise on k'th hidden unit $\xi_{k'}$ and those come from other hidden units $\xi_{l'}$, then the learning equation rewritten as the next equation. $\xi_{k'}$ is considered as a signal and $\xi_{l'}$ is considered as a noise in the signal to noise analysis.

$$\begin{aligned} w_{k'j}^{(m+1)} &= w_{k'j}^{(m)} + \frac{\eta}{2N\sigma_\xi^2} \\ &\cdot\left\{2((\xi_{k'})^2 + \sum_{l'\neq k'}^{M}\xi_{l'}\xi_{k'})\left(\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{M} g(y_{l'}^{(m)})\right)\right. \\ &\left. -\left((\xi_{k'})^3 + \sum_{l'\neq k'}^{M}(\xi_{l'})^2 x_{k'} + \sum_{l'=1}^{M}\sum_{i'\neq l'}^{M}\xi_{l'}\xi_{i'}\xi_{k'}\right)\right\} g'(y_{k'}^{(m)}) x_j^{(m)} \end{aligned} \quad (16)$$

Here, the iteration number m on the noise ξ is omitted.

In Eq. (16), $\sum_{l'\neq k'}^{M}\xi_{l'}\xi_{k'}$ and $\sum_{l'\neq k'}^{M}(\xi_{l'})^2 x_{k'} + \sum_{l'}^{M}\sum_{i'\neq l'}^{M}\xi_{l'}\xi_{i'}\xi_{k'}$ are the cross-talk noise from other hidden units. Each hidden unit receives the sum of the noises added to each hidden unit. However, the mean value of the cross-talk noise is eliminated because $\xi_{k'}$ is independent of the other hidden-unit noises. The derivative of each hidden unit $g'(y_{k'})$ is independent from those of the other hidden units.

3.3 Generalization Error

The generalization error of applying NP learning to a soft-committee machine matches that of the soft-committee machine

itself. Therefore, it is given by the squared error averaged over all possible inputs, as follows:

$$\varepsilon_g = \int \mathrm{d}\boldsymbol{x} P(\boldsymbol{x}) E = \langle E \rangle . \tag{17}$$

Here, $P(\boldsymbol{x})$ is the probabilistic distribution of the input. $\langle \cdot \rangle$ denotes the average over the inputs. The generalization error of the soft-committee machine using stochastic gradient descent is given by Saad et al., and we follow their calculation [8]. Accordingly, the generalization error is

$$\begin{aligned}\varepsilon_g &= \frac{1}{2}\langle E \rangle = \frac{1}{2}\left\langle \left(\sum_{k=1}^{K} g(d_k) - \sum_{k'=1}^{M} g(y_k) \right)^2 \right\rangle \\ &= \frac{1}{2}\left\{ \sum_{k=1}^{K}\sum_{l=1}^{K} \langle g(d_k) g(d_l) \rangle + \sum_{k'=1}^{K}\sum_{l'=1}^{M} \langle g(y_{k'}) g(y_{l'}) \rangle \right. \\ &\quad \left. - 2\sum_{k=1}^{K}\sum_{k'=1}^{M} \langle g(d_k) g(y_{k'}) \rangle \right\}\end{aligned} \tag{18}$$

Here, $g(\cdot)$ is a sigmoidal function, i.e., $g(x) = \mathrm{erf}(\frac{x}{\sqrt{2}})$. Eq. (18) can thus be rewritten as

$$\begin{aligned}\varepsilon_g = \frac{1}{\pi}\left\{ \sum_{k=1}^{K}\sum_{l=1}^{K} \arcsin \frac{T_{kl}}{\sqrt{1+T_{kk}}\sqrt{1+T_{ll}}} \right. \\ + \sum_{k'=1}^{M}\sum_{l'=1}^{M} \arcsin \frac{Q_{k'l'}}{\sqrt{1+Q_{k'k'}}\sqrt{1+Q_{l'l'}}} \\ \left. - 2\sum_{k=1}^{K}\sum_{k'=1}^{M} \arcsin \frac{R_{kk'}}{\sqrt{1+T_{kk}}\sqrt{1+Q_{k'k'}}} \right\}.\end{aligned} \tag{19}$$

Here, T_{kl}, $Q_{k'l'}$, and $R_{kk'}$ are the order parameters defined by

$$T_{kl} = \langle d_k d_l \rangle = \boldsymbol{w}_k^* \cdot \boldsymbol{w}_l^* \tag{20}$$

$$Q_{k'l'} = \langle y_{k'} y_{l'} \rangle = \boldsymbol{w}_{k'} \cdot \boldsymbol{w}_{l'} \tag{21}$$

$$R_{kk'} = \langle d_k y_{k'} \rangle = \boldsymbol{w}_k^* \cdot \boldsymbol{w}_{k'} \tag{22}$$

Note that T_{kl} is a constant value and is the correlation between the weight vectors $\boldsymbol{w}_k^*$ of the kth weight vector and the lth weight vectors $\boldsymbol{w}_l^*$. In the limit $N \to \infty$, $T_{kl} = \delta_{kl}$, where δ_{kl} is the Kronecker delta. By substituting Eqs. (21) and (22) at each learning iteration m into Eq. (19), we can calculate the generalization error at m. In the following part of the paper, we call $R_{kk'}$ and $Q_{k'l'}$ as the overlap.

The overlaps $R_{kk'}$ and $Q_{k'l'}$ in learning are determined with the following procedure. $R_{kk'}$ is calculated at each learning iteration as the inner product of the k'th weight vector of the student and the kth teacher weight vector. $Q_{k'l'}$ is calculated as the inner product of the k' weight vector of the student at each learning iteration and the l'th student weight vector. For the output NP case, the weight vector is updated using Eq. (12), while for the hidden NP case, it is updated by Eq. (16). Equations (12) and (16) are recursion forms for updating the weight. The weights at each iteration are calculated in three steps: (1) initialize the weight vectors of the teacher and student by drawing from a probability distribution in accordance with Eqs. (2) and (5); (2) generate input by drawing from a probability distribution in accordance with Eq. (1); (3) update the weights by using Eq. (12) or (16). Steps (2) and (3) are repeated until the learning stopping conditions are satisfied.

From Eq. (8), the generalization error is zero when the teacher and student are identical. As such, $R_{kk'}$ and $Q_{k'l'}$ satisfy the following conditions in the thermodynamic limit, $N \to \infty$.

$$Q_{k'l'} = \delta_{k'l'} \tag{23}$$

$$R_{kk'} = \delta_{kk'} \tag{24}$$

4. Results

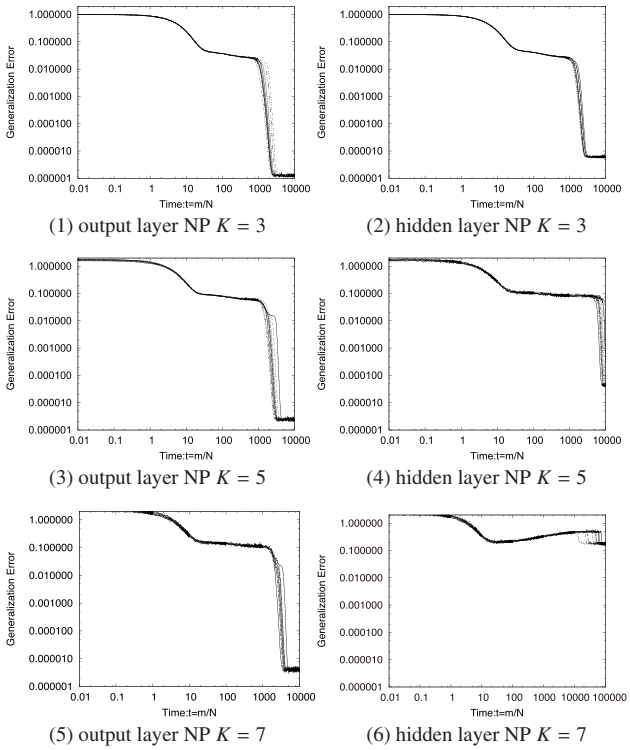
This section compares the dynamic properties of the case of adding noise to the output, the case of adding noise to the hidden layer, and the case of noisy learning.

The procedure described in Section 3.3 is used to determine $R_{kk'}$ and $Q_{k'l'}$ at each learning iteration, and the generalization error is calculated by substituting $R_{kk'}$ and $Q_{k'l'}$ into Eq. (19). The initial weight vectors of the teacher and student are set using the same procedure as in Section 3.1. In the following, the results of ten trials using different initial weight vectors are plotted on the same graph. Learning was stopped at $t = m/N = 10000$. Here, $N = 1000$, and m is the number of learning iterations.

4.1 Effect of Varying the Number of Hidden Units

In the output NP case, the noise added to the output unit propagates to the hidden units as common noise. Other hand, in the hidden NP case, noise is added to the hidden units independently, but from Eq. (13), the input potential of the output unit is the sum of those of the hidden unit outputs with noise ($\sum_{l'}(g(y_{l'}) + \xi_{l'})$), and this cross-talk noise propagates to each hidden unit when the weight vector is updated, as shown in Eq. (16). Consequently, the number of hidden units is set to 3, 5, or 7.

Figure 2 shows the results. We set the learning step size to $\eta = 0.1$ and drew the perturbation noise from a probability distri-

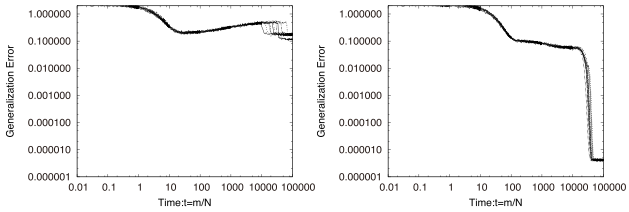


(1) output layer NP $K = 3$ (2) hidden layer NP $K = 3$

(3) output layer NP $K = 5$ (4) hidden layer NP $K = 5$

(5) output layer NP $K = 7$ (6) hidden layer NP $K = 7$

Fig. 2 Time course of the generalization error in the output NP and hidden NP cases. Number of hidden units $K = 3$, 5, or 7.



(1) $\eta = 0.1, \sigma_\xi^2 = 10^{-30}$ (2) $\eta = 0.01, \sigma_\xi = 10^{-5}$

Fig. 3 Evaluation changing two of the learning conditions: (1) effect of using a smaller perturbation noise and (2) effect of using a smaller learning step size. The original (baseline) conditions are $\sigma_\xi^2 = 10^{-5}$ and $\eta = 0.1$.

bution with mean zero and variance $\sigma_\xi^2 = 10^{-5}$. The teacher and student had the same architecture, so they had the same number of hidden units, i.e., $K = M$. Figure 2 (1), (3), and (5) show the results of the output NP case, and Fig. 2 (2), (4), and (6) show the results for the hidden NP case. In these figures, the horizontal axis is time $t = m/N$, where m is the number of learning iterations and N is the number of input dimensions, i.e., N = 1000. The vertical axis is the generalization error.

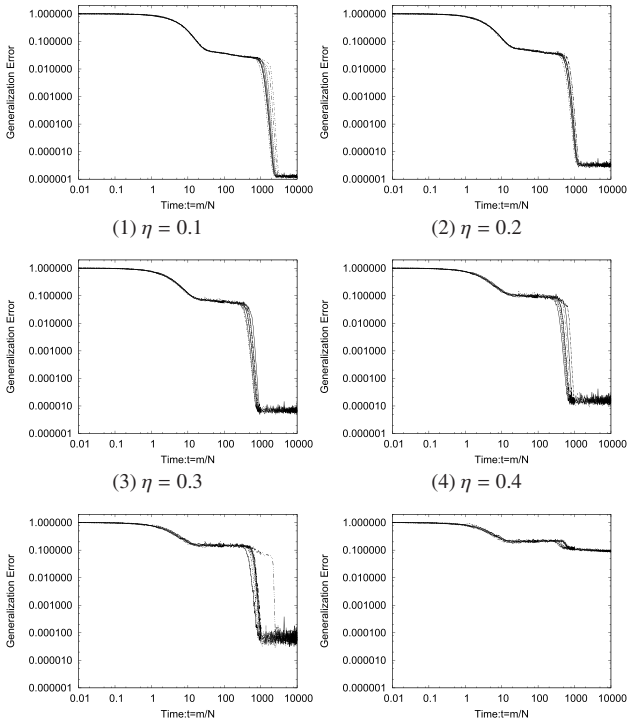
From Fig. 2 (1), (3), and (5), it is clear that the residual error becomes larger as the number of hidden units increases. However, the time it takes to escape from the plateau changes slightly with the number of hidden units. On the other hand, the escape from the plateau in Fig. 2 (2), (4), and (6) occurs later and at a larger number of hidden units. For $K = 7$, the learning time is very long (longer than the plot), and the generalization error does not monotonically decrease. These results show that in the hidden NP case, the cross-talk noise in each hidden unit is not eliminated when the number of hidden units is large, and this noise has a bad effect on the learning. Therefore, we investigated the effect of varying the learning conditions of the hidden NP case, i.e., by making the perturbation noise and learning step size smaller.

Figure 3 shows the results. The number of hidden units was $K = 7$, as in Fig. 2 (6). In (1), the learning step size is the same as in Fig. 2 (6), but the perturbation noise is $\sigma_\xi^2 = 10^{-30}$. It is apparent that the smaller perturbation noise didn't shorten the time it took to escape from the plateau. Thus, the magnitude of the variance of the perturbation noise has no effect on the plateau phenomenon. The noise makes the residual error smaller as its variance becomes smaller. Figure 3 (2) shows the case of perturbation noise having the same variance as in Fig. 2 (6), but a learning step size of $\eta = 0.01$. In this figure, the generalization error decreases. Overall, these results indicate that the generalization error converges within a sufficiently short learning time by using a smaller learning step size.

4.2 Effect of Varying the Learning Step Size

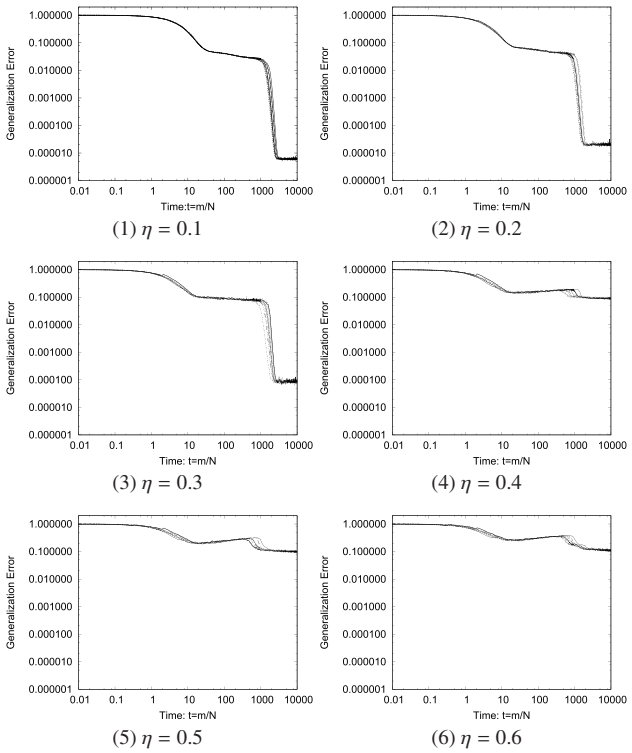
Next, we analyzed the effect of changing the learning step size. As described in Section 4.1, the number of hidden units in the teacher and student was set to $K = M = 3$, and the number of input dimensions was $N = 1000$. The variance of the perturbation noise for both cases was set to $\sigma_\xi^2 = 10^{-5}$. Moreover, the mean of the noise was zero in both cases. The learning step sizes were 0.1, 0.2, 0.3, 0.4, 0.5, and 0.6.

Figure 4 shows the generalization error for the output NP case,



(1) $\eta = 0.1$ (2) $\eta = 0.2$
(3) $\eta = 0.3$ (4) $\eta = 0.4$
(5) $\eta = 0.5$ (6) $\eta = 0.6$

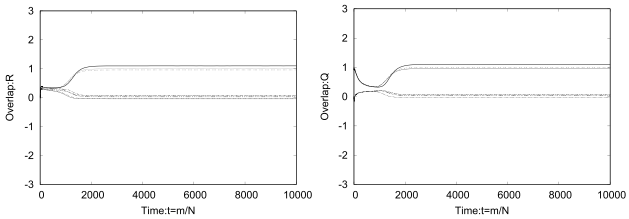
Fig. 4 Time course of the generalization error of the output NP case for different learning step sizes.



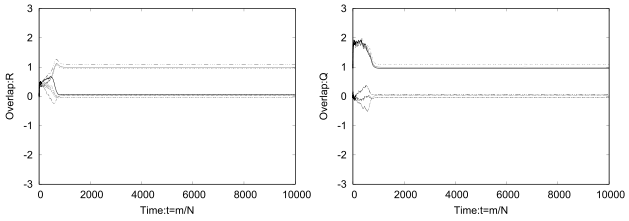
(1) $\eta = 0.1$ (2) $\eta = 0.2$
(3) $\eta = 0.3$ (4) $\eta = 0.4$
(5) $\eta = 0.5$ (6) $\eta = 0.6$

Fig. 5 Time course of the generalization error of the hidden NP case for different learning step sizes.

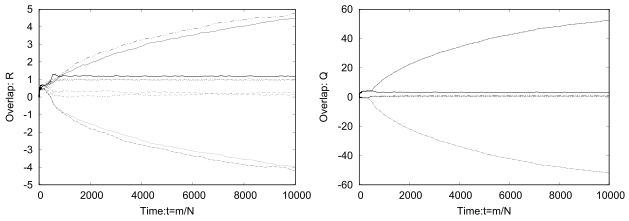
and **Fig. 5** shows it for the hidden NP case. The generalization error was calculated following the procedure in Section 3.3. In Fig. 4, the generalization error decreases until the learning step size η is less than 0.5. The time it takes to escape from the plateau for $0.3 \leq \eta \leq 0.5$ was independent of size of the learning step size. The residual error was much larger when $\eta = 0.6$. We will discuss the reason for the enlarged error later. It took longer to



(1) time course of $R_{kk'}$ (left) and $Q_{kl'}$ (right) when $\eta = 0.1$.

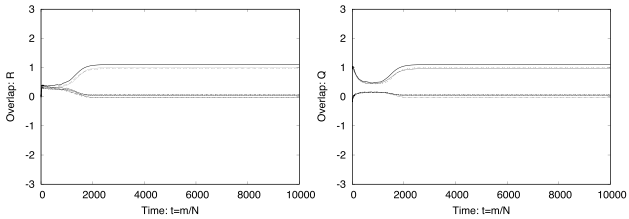


(2) time course of $R_{kk'}$ (left) and $Q_{kl'}$ (right) when $\eta = 0.5$.

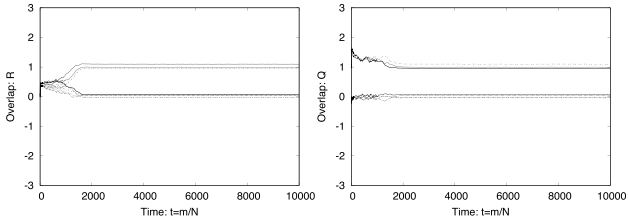


(3) time course of $R_{kk'}$ (left) and $Q_{kl'}$ (right) when $\eta = 0.6$.

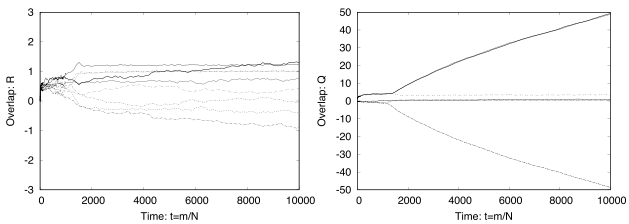
Fig. 6 Time course of overlap $R_{kk'}$ and $Q_{kl'}$ in the output NP case.



(1) time course of $R_{kk'}$ (left) and $Q_{kl'}$ (right) when $\eta = 0.1$.



(2) time course of $R_{kk'}$ (left) and $Q_{kl'}$ (right) when $\eta = 0.3$.



(3) time course of $R_{kk'}$ (left) and $Q_{kl'}$ (right) when $\eta = 0.4$.

Fig. 7 Time course of overlap $R_{kk'}$ and $Q_{kl'}$ in the hidden NP case.

escape from the plateau when $\eta < 0.3$ than when $0.3 \leq \eta \leq 0.5$.

As shown in Fig. 5, the generalization error in the hidden NP case converged for $\eta = 0.1, 0.2$, and 0.3. However, the residual error was large, and the generalization error did not decrease for $\eta > 0.4$. The residual error was larger than that of the output NP case for the same variance of perturbation noise and learning step size. Therefore, the hidden NP case can use only a smaller range of learning step sizes. Next, we investigated the reasons for the poor convergence in this case by examining the behavior of $R_{kk'}$ and $Q_{k'l'}$.

From Eq. (14), the generalization error was zero when the order parameters were $T_{kk} = 1$, $T_{kk'} = 0$, $R_{kk} = 1$, $R_{kk'} = 0$, $Q_{k'k'} = 1$, and $Q_{k'l'} = 0$. Thus, a large residual error may occur when the order parameters do not satisfy the conditions that achieve a generalization error of zero. For the output NP case, we analyzed settings of $\eta = 0.1$, $\eta = 0.5$ and $\eta = 0.6$. For the hidden NP case, we analyzed settings of $\eta = 0.1$, $\eta = 0.3$ and $\eta = 0.4$.

Figure 6 shows the time courses of $R_{kk'}$ and $Q_{k'l'}$ in the output NP case. In the figures, horizontal axis is time $t = m/N$, and vertical axis is $R_{kk'}$ or $Q_{k'l'}$. (1) shows the case of $\eta = 0.1$, (2) that of $\eta = 0.5$, and (3) that of $\eta = 0.6$. $R_{kk'}$ is the inner product of the kth weight vector of the teacher and the k'th weight vector of the student. Thus, if the teacher and student vectors are identical, $R_{kk'} = 1$. For $K = 3$, if the three teacher weight vectors and three student weight vectors are identical, then $R_{11} = 1$, $R_{22} = 1$, $R_{33} = 1$, and $R_{kk'} = 0$, where $k \neq k'$. $Q_{k'l'}$ is the inner product of the k'th weight vector of the student and the l'th weight vector of the student. When $k' = l'$, $Q_{k'k'}$ is the norm of the k'th weight vector. From Eq. (2), the norm of the teacher weight vector is 1, so $Q_{k'k'}$ is 1 if the generalization error is zero. Moreover, when $k' \neq l'$, $Q_{k'l'}$ is the correlation between k'th weight vector $\boldsymbol{w}_{k'}$ and the l'th weight vector $\boldsymbol{w}_{l'}$, and $Q_{k'l'} = 0$ if the generalization error converges to zero.

From Fig. 6 (1), the three R_{kk} converged to almost 1, and $R_{kk'}$ converged to almost zero at $t = 10000$. All three $Q_{k'k'}$ converged to $Q_{k'k'} = 1$, and others converged to $Q_{k'l'} = 0$. It is clear that the output NP case for $\eta = 0.1$ achieved a generalization error of zero. Figure 6 (2) ($\eta = 0.5$) show the behavior of $R_{kk'}$ and $Q_{k'l'}$ just before the residual error suddenly enlarged. From the figure, the behavior of the overlaps $R_{kk'}$ and $Q_{k'l'}$ are similar to that of Fig. 6 (1). From these results, It is clear that the output NP case for $\eta = 0.5$ also achieved a generalization error of zero. Figure 6 (3) shows different behavior from what is shown in Fig. 6 (1) and Fig. 6 (2). All three R_{kk} converged to almost 1, and two $R_{kk'}$ converged to around zero. However, the other $R_{kk'}$ did not converge to zero and remained large. In addition, two $Q_{k'k'}$ converged to 1, while one $Q_{k'k'}$ converged to 3. $Q_{k'l'}$ did not converge to zero. Therefore, when $\eta = 0.6$, some of the overlaps converged to too large a value, and this may have led to a larger residual error.

Figure 7 show the time courses of $R_{kk'}$ and $Q_{k'l'}$ in the hidden NP case. (1) depicts results for when $\eta = 0.1$, (2) for $\eta = 0.3$, and (3) for $\eta = 0.4$. The results in (1) and (2) are similar to those in Fig. 6 (1) and (2); thus, they show that the hidden NP case learned properly when $\eta = 0.1$ and $\eta = 0.3$. The results in (3) are similar to those of Fig. 6 (3). Therefore, the hidden NP case failed to learn when $\eta = 0.4$.

If the cross-talk noise in the hidden NP case is not eliminated, then the squared error becomes larger, and a larger squared error enlarges the norm of the weight vector. By comparing the $Q_{k'k'}$ of the output NP case (Fig. 6 (2)) and that of the hidden NP case (Fig. 7 (3)) given similar learning step sizes, we see that the norm of the weight vector is much longer than that of the output NP

case, and this means that the hidden NP case has a much larger squared error than the output NP case has.

4.3 Comparison with Noisy Learning

Perturbation noise is added to the output layer or hidden layer of student to get gradient information of the squared error. However, it is also useful for clarifying the difference in effect of adding noise in noisy learning and adding noise in NP learning. The learning equation of noisy learning is as follows:

$$w_{k'j}^{(m+1)} = w_{k'j}^{(m)} + \frac{\eta}{N}\left\{\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{M}\left(g(y_{l'}^{(m)}\right) - \xi\right\} \times g'(y_{k'})x_j^{(m)}, \tag{25}$$

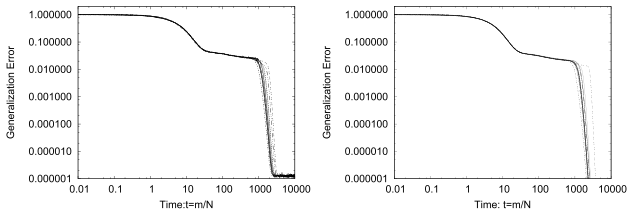
$$w_{k'j}^{(m+1)} = w_{k'j}^{(m)} + \frac{\eta}{N}\left\{\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{M}\left(g(y_{l'}^{(m)}) + \xi_{l'}\right)\right\} \times g'(y_{k'})x_j^{(m)}. \tag{26}$$

Here, Eq. (25) is noisy learning in which noise is added to the output layer, while Eq. (26) is noisy learning in which noise is added to the hidden layer. In Eqs. (25) and (26), $g(x) = \mathrm{erf}(\frac{x}{\sqrt{2}})$. **Figure 8** shows the results.

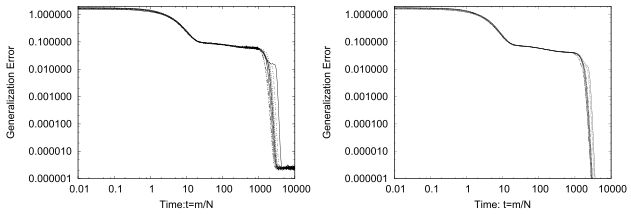
The learning step size was $\eta = 0.1$, and the added noise was drawn from a probability distribution of mean zero and variance $\sigma^2 = 10^{-5}$. The number of hidden units was $K = 3$ or $K = 5$. In the figures, the horizontal axis is time $t = m/N$, and the vertical axis is the generalization error. The generalization error was calculated using $R_{kk'}$ and $Q_{k'l'}$ at each learning iteration and Eq. (19). From Fig. 8 (1) and (2), the time it takes to escape from the plateau in the output NP case is almost the same as that of noisy learning when noise is added to the output layer. However, the residual error is small in noisy learning. Although it is not shown in Fig. 8 (1) and (2), we found that the residual error of noisy learning when noise is added to the output layer was 2×10^{-11}. The time it takes to escape from the plateau in both the output NP case and noisy learning when noise is added to the output layer did not change when the hidden units were increased from $K = 3$ to $K = 5$. These results indicate that the output NP case and noisy learning when noise is added to the output had similar learning performances, except for the residual error. From Fig. 8 (3), the time it takes to escape from the plateau in the hidden NP case and noisy learning when noise is added to the hidden layer were similar; however, from Fig. 8 (4), the hidden NP case took much longer to escape from the plateau. Although it is not shown in Fig. 8 (3) and (4), we found that the residual error of noisy learning when noise is added to the hidden layer was 8×10^{-11}. It is interesting that NP learning using implicit gradient information has similar performance to noisy learning using explicit gradient information.

5. Conclusion

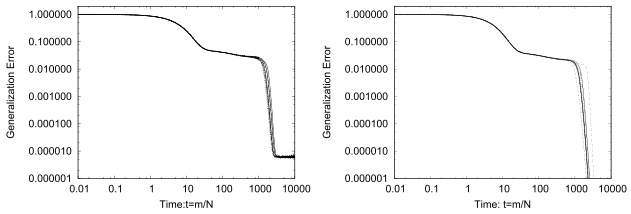
We proposed two implement methods of NP learning for multilayer networks; adding noise to either the output layer or hidden layer of a soft-committee machine. We analyze the learning equation of output NP case. It adding noise to output unit, so each hidden unit receive the same error. So, each hidden unit may updates



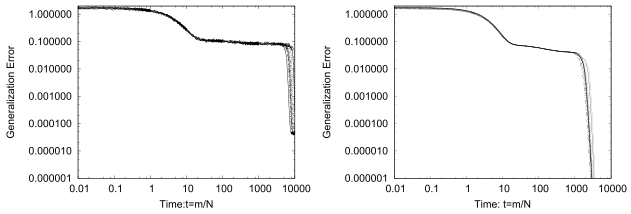
(1) output NP case (left) and noisy learning in which noise is added to the output layer (right). $K = 3$.



(2) output NP case (left) and noisy learning in which noise is added to the output layer (right). $K = 5$.



(3) hidden NP case (left) and noisy learning in which noise is added to the hidden layer (right). $K = 3$.



(4) hidden NP case (left) and noisy learning in which noise is added to the hidden layer (right). $K = 5$

Fig. 8 Generalization errors of NP learning and noisy learning.

the same way and learning is done improperly. However, by using characteristic eligibility of weight, the derivative of each hidden unit is independent, and output NP case can learn properly. This analysis was supported by the computer analysis. We also analyzed the learning equations of two implementations by using signal-to-noise analysis. The results show that adding noise to output is free from cross-talk noise from other hidden units, and adding noise to hidden units is affected by cross-talk noise. This affect tend to be larger when the number of hidden units becomes larger. So, adding noise to hidden units have some limitation in the number of hidden units.

We compared two implementations by the time it takes to escape from the plateau phenomena. As the results of signal-to-noise analysis of the two learning equations, output NP case has an advantage to hidden NP case. In output NP case, the time it tales to escape from the plateau phenomena is not changed for number of hidden units, and the size of learning step size. However, in hidden NP case, the time it takes to escape from plateau delayed for larger number of hidden units, and larger size

of learning step size. So, output NP case is selected as implementation of NP learning for soft-committee machine.

When the number of hidden units is large or the learning step size is large, worth generalization error is obtained for both implementations. The reason of worth generalization error is analyzed in terms of the overlaps $R_{kk'}$ and $Q_{k'l'}$. From the results, the reason is due to the cross-talk noise, and the length of student vectors becomes larger by the cross-talk noise and this becomes cause larger generalization error. However, it is possible to improve the generalization by using smaller learning step size.

Moreover, we compared the proposed NP learnings with noisy learning. The results showed that performance of output NP case was similar to that of the noisy learning. Therefore, it is found that output NP case does't use explicit gradient information, however, it achieved the similar performance as the noisy learning that using explicit gradient information. In the future, we will analyze the application of NP learning to a two layer network.

References

[1] Widrow, B. and Lehr, M.A.: 30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation, *Proc. IEEE*, Vol.78, No.9, pp.1415–1442 (1990).

[2] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Machine Learning*, Vol.8, pp.229–256 (1992).

[3] Fiete, I.R., Fee, M.S. and Seung, H.S.: Model of Birdsong Learning Based on Gradient Estimation by Dynamic Perturbation of Neural Conductances, *Journal of Neurophysiology*, Vol.98, pp.2038–2057 (2007).

[4] Werfel, J., Xie, X. and Sueng, H.S.: Learning curves for stochastic gradient descent in linear feedforward networks, *Neural Computation*, Vol.17, pp.2699–2718 (2005).

[5] Saad, D. (Ed.): *On-line learning in neural networks*, Cambridge: Cambridge University Press (1999).

[6] Biehl, M. and Riegler, P.: On-Line Learning with a Perceptron, *Europhysics Letters*, Vol.28, No.7, pp.525–530 (1994).

[7] Biehl, A. and Schwarze, H.: Learning by on-line gradient descent, *J. Phys. A: MAth. Gen.*, Vol.28, 643 (1995).

[8] Saad, D. and Solla, S.A.: On-line learning in soft committee machines, *Physical Review E*, Vol.52, No.4 (1995).

[9] Nishimori, H.: *Statistical physics of spin glass and information processing: An introduction*, Oxford: Oxford University Press (2001).

[10] Engel, A. and den Broeck, C.V.: Statistical Mechanics of Learning, Cambridge University Press, Cambridge, UK, 1st edition (2001).

[11] Krogh, A.: Learning with noise in a linear perceptron, *Journal of Physics A: Mathematical and General*, Vol.25, No.5, pp.1119–1133 (1992).

[12] Krogh, A. and Hertz, J.A.: Generalization on a linear perceptron in the presence of noise, *Journal of Physics A: Mathematical and General*, Vol.25, No.5, pp.1135–1147 (1992).

[13] Hara, K., Katahira, K., Okanoya, K. and Okada, M.: Statistical mechanics of on-line node-perturbation learning, *Information Processing Society of Japan Trans. Mathematical Modeling and Its Applications*, Vol.4, No.1, pp.72–81 (2011).

[14] Cho, T., Katahira, K., Okanoya, K. and Okada, M.: Node perturbation learning without noiseless baseline, *Neural Networks*, Vol.24, pp.267–272 (2011).

[15] Hara, K., Katahira, K., Okanoya, K. and Okada, M.: Statistical mechanics of Node-perturbation Learning for nonlinear perceptron, *Journal of Physical Society of Japan*, Vol.82, 054001 (2013).

[16] Moody, J.E.: The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems, *Proc. Advances in Neural Information Processing Systems*, Vol.4, pp.847–854 (1991).

[17] Bishop, C.M.: Training with noise is equivalent to Tikhonov regularization, *Neural Computation*, Vol.7, No.1, pp.108–116 (1995).

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 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. He is a senior member of the IPSJ and a member of the JPS, IEEE and IEICE.

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. From 2004 to 2005, he worked at Yamaha Corporation. From 2009 to 2012, he was an researcher of Japan Science Technology Agency, ERATO, OKANOYA Emotional Information Project. Currently, he is an associate professor at Graduate School of Informatics, Nagoya University. His research interests include computational and statistical modeling of behavior and learning of psychology.

Masato Okada received B.Sc. degree in physics from Osaka City University in 1985, M.Sc. degree in physics and Ph.D degrees in science from Osaka University, Osaka, Japan, in 1987, and 1997, respectively. From 1987 to 1989, he worked at Mitsubishi Electric Corporation, and from 1991 to 1996, he was a Research Associate at Osaka University. He was a Researcher on the Kawato Dynamic Brain Project until 2001. He was a Deputy Head of the Laboratory for Mathematical Neuroscience, RIKEN Brain Science Institute, Saitama, Japan, and a PRESTO Researcher on intelligent cooperation and control at the Japan Science and Technology Agency until 2004. Currently, he is a Professor in the Graduate School of Frontier Science, The University of Tokyo. His research interests include the computational aspects of neural networks and statistical mechanics for information processing.