Node-perturbation Learning applied for Soft-committee machine

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

Abstract: Node-perturbation learning is a 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. 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 simulations to show the validity of the proposed 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 (NP learning) based on the stochastic gradient method. 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 by using the ensemble mean [4].

Statistical mechanics has been used to study online learning [9], [10], [11], [12], mainly the simple perceptron. Statistical mechanics gives a compact description of the dynamics of learning that uses a large input dimension N and especially provides an accurate model of the mean behavior for realistic values of N[5], [6], [7], [8].

In a previous study, we analyzed the dynamics of NP learning applied for linear perceptrons [13]. In particular, we used statistical mechanics to determine the generalization error of NP learning and arrived at two findings. The first is that NP learning is the same as noisy learning that changes the amount of noise related to the error. The second is that the generalization error increases as a result of cross-talk noise generated by the other output noise. Moreover, we analyzed the dynamics of NP learning in non-linear perceptrons [14]. In that paper, we showed that NP learning performs better when using nonlinear perceptrons than when using linear ones.

In the current paper, we apply NP learning to a multi-layer neural network called a soft-committee machine and analyze the dynamics of the learning behavior. 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. A soft-committee machine has a simple network structure; however, it suffers from plateau phenomena and symmetry breaking. Our findings give insights for networks with more complicated layers.

2. 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.

2.1 Model

First, we formulate the teacher and student networks. 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

[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

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 [6]. The hidden outputs are determined by majority vote.

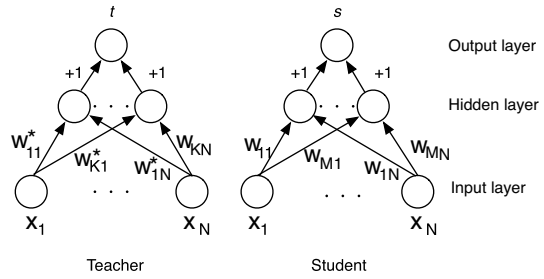


Fig. 1 Structure of Teacher and student networks.

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.

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 $\boldsymbol{w}_{kj}^*$ 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 $\boldsymbol{w}_{k'j}$ 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.

2.2 Node-perturbation Learning 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

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

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. Accordingly, the learning equation is defined as

$$w_{k'j}^{(m+1)} = w_{k'j}^{(m)} - \frac{\eta}{N}(E_\xi - E) g'(y_{k'}) \frac{\xi}{\sigma_\xi^2} x_j \tag{9}$$

Here, 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.

2.2.1 Adding noise to the output layer

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

$$E_\xi^{(m)} = \frac{1}{2}\left(t^{(m)} - (s^{(m)} + \xi^{(m)})\right)^2 = \frac{1}{2}\left(\sum_{k=1}^{K} g(d_k^{(m)}) - \sum_{k'=1}^{M} g(y_{k'}^{(m)}) - (\xi^{(m)})^2\right)^2 . \quad (10)$$

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

$$E_\xi^{(m)} - E^{(m)} = -\frac{1}{2}\left\{2\xi^{(m)}\left(\sum_{k=1}^{K} g(d_k^{(m)}) - \sum_{k'=1}^{M} g(y_{k'}^{(m)})\right) - (\xi^{(m)})^2\right\}. \quad (11)$$

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). Eq. (11) becomes negative when $E_\xi < E$, and the weight vector is updated in the direction of the noise ξ[2]. Identical noise is added to the output layer unit at every learning iteration. Accordingly, the learning equation of the output NP case is defined as

$$w_{k'j}^{(m+1)} = w_{k'j}^{(m)} + \frac{\eta}{2N}\left\{2\xi^{(m)}\left(\sum_{k=1}^{K} g(d_k^{(m)}) - \sum_{k'-1}^{M} g(y_{lk}^{(m)})\right) - (\xi^{(m)})^2\right\} g'(y_{k'}^{(m)}) \frac{\xi^{(m)}}{\sigma_\xi^2} x_j^{(m)} \quad (12)$$

IIn Eq. (12), the same noise ξ is added to the output of every hidden unit; however, the independence of each hidden unit is kept by the independence of the derivative at each hidden unit $g'(y_{k'})$

2.2.2 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

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

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

$$E_\xi - E = -\frac{1}{2}\left\{2\sum_{k'=1}^{M} \xi_{k'}\left(\sum_{k=1}^{K} g(d_k) - \sum_{k'=1}^{M} g(y_{k'})\right) - \left(\sum_{k'=1}^{M} \xi_{k'}\right)^2\right\} \quad (14)$$

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 $\xi_{l'}$. The learning equation for the case of adding noise to the hidden layer (hidden NP case) is defined as

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

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.

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

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.

2.3 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:

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

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 D. Saad et al., and we follow their calculation [6]. The number of hidden units in the teacher is denoted as K and that of the student is denoted as M. Accordingly, the generalization error is

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

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

$$\varepsilon_g = \frac{1}{\pi}\left\{\sum_{k=1}^{K}\sum_{l=1}^{K}\sin^{-1}\frac{T_{kl}}{\sqrt{1+T_{kk}}\sqrt{1+T_{ll}}} + \sum_{k'=1}^{M}\sum_{l'=1}^{M}\sin^{-1}\frac{Q_{k'l'}}{\sqrt{1+Q_{k'k'}}\sqrt{1+Q_{l'l'}}} - 2\sum_{k=1}^{K}\sum_{k'=1}^{M}\sin^{-1}\frac{R_{kk'}}{\sqrt{1+T_{kk}}\sqrt{1+Q_{k'k'}}}\right\}. \quad (19)$$

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

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

$$Q_{k'l'} = \langle y_{k'} y_{l'} \rangle = \boldsymbol{w}_{k'} \cdot \boldsymbol{w}_{l'}^T \tag{21}$$

$$R_{kk'} = \langle d_k y_{k'} \rangle = \boldsymbol{w}_k^* \cdot (\boldsymbol{w}_{k'})^T \tag{22}$$

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). Equation (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 Eq. (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, $N \to \infty$.

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

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

3. Results

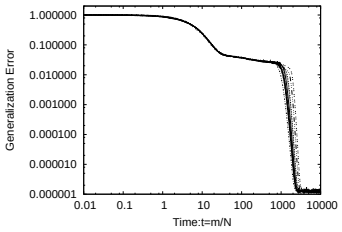
The hidden NP is the same as output NP except for the crosstalk noise, so we only analyzed the dynamic properties of the output NP (subsection 2.2.1) and results are compared with those of noisy learning.

In this section, the same procedures as subsection 2.3 are used to get the order parameters $R_{kk'}$ 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 values of the weight vectors of teacher and student are set by the same procedures as in subsection 2.1. In the following, the results of 10 trials that using different initial input-to-hidden weight vectors are plotted on the same graph.Learning is stopped at $t = m/N = 10000$. Here, $N = 1000$ and m is the iteration number of learning.

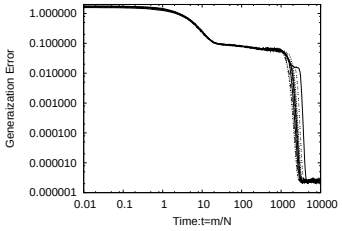
3.1 Effect of varying the number of hidden units

In this subsection, we analyze effect of changing the number of hidden units in the generalization error.

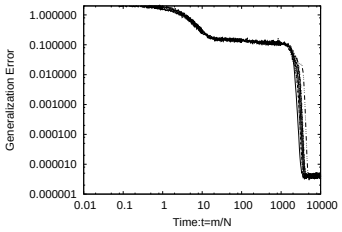
In the output NP case, the noise added to the output unit propagates to the hidden units as common noise. Consequently, the number of hidden units is set to 3, 5, or 7.



(1) output layer NP $K = 3$



(2) output layer NP $K = 5$



(3) output layer NP $K = 7$

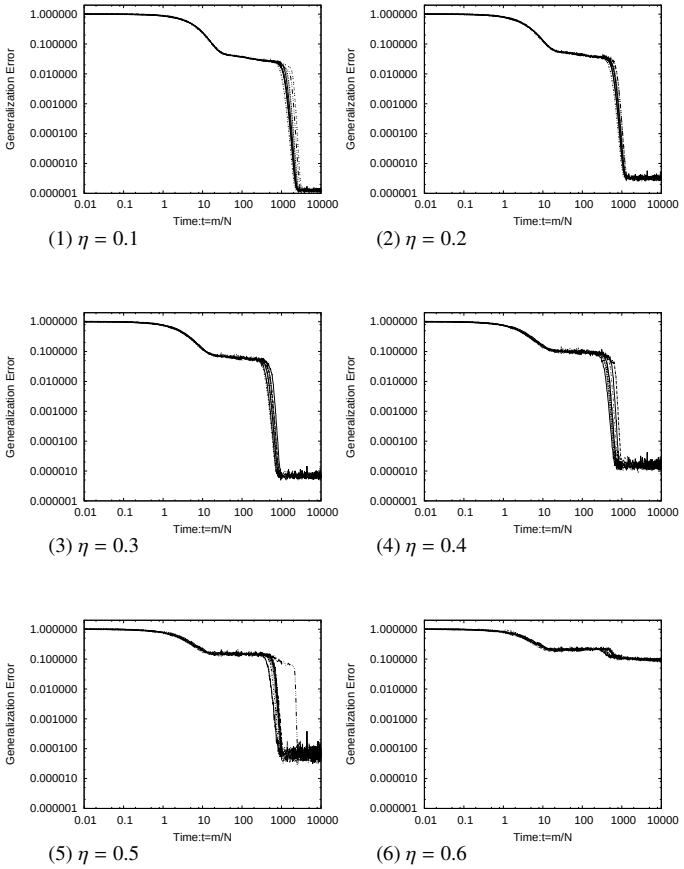
Fig. 2 Time course of the generalization error in the output NP case. Number of hidden units $K = 3$, 5, or 7.

Figure 2 shows the results. We set the learning step size to $\eta = 0.1$ and drew the perturbation noise from a probability distribution 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), (2), and (3) show the results of the output 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), (2), and (3), 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.

3.2 Effect of varying the learning step size

Next, we analyzed the effect of changing the learning step size. As described in subsection 3.1, 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 3 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. 3 Time course of the generalization error of the output NP case for different learning step sizes.

The generalization error was calculated following the procedure in subsection 2.3. In Fig. 3, 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 $0.3 < \eta < 0.5$ 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 escape from the plateau when $\eta < 0.3$ than when $0.3 < \eta < 0.5$.

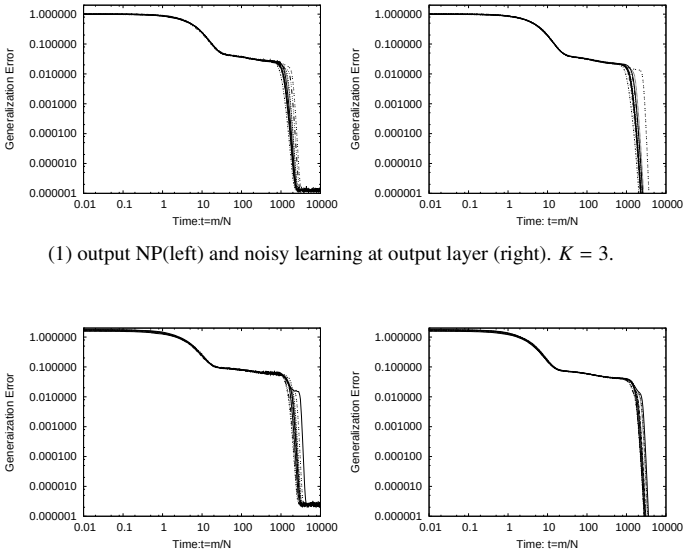
3.3 Comparison with noisy learning

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:

$$w_{k'j}^{(m+1)} = w_{k'j}^{(m)} + \frac{\eta}{N}\left\{\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{M}\left(g(y_{l'}^{(m)}\right) - \xi\right\} \times \sqrt{\frac{2}{\pi}}\exp\left(-\frac{y_{k'}^2}{2}\right)x_j^{(m)}. \quad (25)$$

Here, Eq. (25) is noisy learning in which noise is added to the output layer. Figure 4 shows the results. Figure 4 show the learning results.

The learning step size was $\eta = 0.1$, and the added noise was



(1) output NP(left) and noisy learning at output layer (right). $K = 3$.

(2) output NP(left) and noisy learning at output layer (right). $K = 5$.

Fig. 4 Generalization errors of NP learning and noisy learning.

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. 4(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. 4(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. It is interesting that NP learning using implicit gradient information has similar performance to noisy learning using explicit gradient information.

4. Conclution

We proposed adding noise to either the output layer or hidden layer of a soft-committee machine. We analyzed the dynamic behavior of these NP learnings in terms of the overlaps $R_{kk'}$ and $Q_{k'l'}$. The results indicated that proposed NP learnings were learnable, and they could avoid plateaus. We compared the cases of adding noise to output and that of adding noise to the hidden layer while varying the number of hidden units and learning step size. The results showed that the case of adding noise to the output can learn a wider numerical range of hidden units and step sizes. This difference between it and the case of adding noise to the hidden layer comes from cross-talk noise in the hidden NP case; the cross-talk noise increases the squared error. This makes

the norm of the weight vector longer and the generalization error larger. Moreover, we compared the proposed NP learnings with noisy learning. The results showed that adding noise to the output was similar in effect to noisy learning. In the future, we will analyze the application of NP learning to a two layer network.

References

[1] B. Widrow and M. A. Lehr: 30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation, Proc. IEEE, Vol.78, No.9, pp.1415-1442 (1990).
[2] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning", Machine Learning, **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] J. Werfel, X. Xie and H. S. Sueng, "Learning curves for stochastic gradient descent in linear feedforward networks", Neural Computation **17**, pp. 2699-2718 (2005).
[5] D. Saad editor: *On-line learning in neural networks*. Cambridge: Cambridge University Press, (1999).
[6] D. Saad, S. A. Solla: On-line learning in soft committee machines, Physical Review E, vol. 52, no. 4, (1995).
[7] Biehl, M. and Riegler, P.: On-Line Learning with a Perceptron, Europhysics Letters, Vol.28, No.7, pp.525-530 (1994).
[8] A. Biehl and H. Schwarze: "Learning by on-line gradient descent", J. Phys. A: MAth. Gen. , **28**, 643 (1995).
[9] H. ¯ Nishimori: *Statistical physics of spin glass and information processing: An introduction*. Oxford: Oxford University Press, (2001).
[10] A. Engel and C.V den Broeck : Statistical Mechanics of Learning, Cambridge University Press, Cambridge, UK, 1st edition (2001).
[11] A. Krogh : Learning with noise in a linear perceptron, Journal of Physics A: Mathematical and General, Vol.25, No.5, pp.1119-1133, (1992).
[12] A. Krogh and J. A. Hertz: 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] K. Hara, K. Katahira, K. Okanoya and M. Okada: "Statistical mechanics of on-line node-perturbation learning," Information Processing Society of Japan Trans. on Mathematical Modeling and Its Applications: **4**, 1, 72–81, (2011).
[14] K. Hara, K. Katahira, K. Okanoya, M. Okada: "Statistical mechanics of Node-perturbation Learning for nonlinear perceptron", Journal of Physical Society of Japan, **82** 054001 (2013).
[15] J. E. Moody: "The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems", Proceeding of Advances in Neural Information Processing Systems 4, pp. 847–854 (1991).
[16] C. M. Bishop: "Training with noise is equivalent to Tikhonov regularization", Neural Computation, **7**, no. 1, pp. 108-116 (1995).