Analysis of Dropout and its Application to Group Dropout

Kazuyuki Hara[1,a)] Daisuke Saitoh[2] Satoshi Suzuki[3] Takumi Kondou[2] Hayaru Shouno[3]

Abstract: Deep learning is a state-of-the-art learning method that is used in fields such as visual object recognition and speech recognition. This learning uses a high number of layers and a huge number of units and connections, so overfitting is a serious problem. Dropout is a kind of regularizer that neglects some inputs and hidden units in the learning process with a probability p; then, the neglected inputs and hidden units are combined with the learned network to express the final output. Warger et al. pointed out that dropout is an adaptive L2 regularizer, so we compared the learning behavior of dropout with that of SGD with the L2 regularizer. Moreover, we found that the process of combining the neglected hidden units with the learned network can be regarded as ensemble learning, so we analyzed dropout learning from this point of view. We compared dropout and ensemble learning from three viewpoints and found that dropout can be regarded as ensemble learning that divides the student network into two groups. On the basis of this insight, we explored novel dropout that divides the student network into more than two groups to enhance the benefit of ensemble learning.

1. Introduction

Deep learning [1],[2] is attracting much attention in visual object recognition, speech recognition, object detection, and many other fields. It provides automatic feature extraction and can achieve outstanding performance [3].

Deep learning uses a very deep layered network and a huge amount of data, so overfitting is a serious problem. To avoid overfitting, dropout [3] is used for regularization. Dropout consists of two processes. During learning, some hidden units are neglected with a probability p, and this process reduces the network size; therefore, overfitting is avoidable. During testing, learned hidden units and hidden units that have not been learned are summed up and multiplied by p to calculate the network output. Hinton said that this procedure seems like a type of ensemble learning. On the other hand, Warger et al. pointed out that dropout is an adaptive L2 regularizer.

Ensemble learning improves the performance of a single network using many networks. Bagging and the Ada-boost algorithm are well known [4]. We theoretically analyzed ensemble learning using linear or non-linear perceptrons[5],[6].

In this paper, we first present our analysis of dropout as a regularizer. Then, we present our analysis of dropout regarded as ensemble learning. On-line learning [7],[8] is used to learn a network. We compared the residual error of dropout and stochastic gradient descent (SGD) with L2 regularization to analyze the regularization performance of dropout. Next, we compared the learnability of dropout and that of ensemble learning using the same network structure. The results revealed that dropout can be regarded as ensemble learning, except when different sets of hidden units are used in dropout. After that, we propose a novel dropout called group dropout. The proposed method divides the hidden units in the student network into several groups at once, and then each group learns from the teacher independently. After the learning, group outputs are averaged to calculate the student output. The effect of ensemble learning is more than that of dropout. Finally, we show the validity of the proposed method using computer simulations.

[1] College of Industrial Technology, Nihon University, 1-2-1 Izumicho, Narashino, Chiba 275–8575, Japan
[2] Graduate School of Industrial Technology, Nihon University
[3] Graduate School of Informatics and Engineering,The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182–8585, Japan
[a)] hara.kazuyuki@nihon-u.ac.jp

2. Model

In this paper, we use a teacher-student formulation and assume the existence of a teacher that produces the desired output for learning data. By introducing the teacher, we can directly measure the similarity of the student weight vector compared to that of the teacher. First, we formulate a teacher network (referred to as the "teacher") and a student network (referred to as the "student"), then we introduce the gradient descent algorithm.

The teacher and student are a soft-committee machine with N input units, hidden units, and an output, as shown in Fig. 1. The teacher consists of K hidden units, and the student consists of K' hidden units. Each hidden unit is a perceptron. The kth hidden weight vector of the teacher is $\boldsymbol{B}_k = (B_{k1}, \ldots, B_{kN})$, and the k'th hidden weight vector of the student is $\boldsymbol{J}_{k'}^{(m)} = (J_{k'1}^{(m)}, \ldots, J_{k'N}^{(m)})$, where m denotes the number of learning iterations. In the soft-committee machine, all hidden-to-output weights are fixed to +1 [8]. This network calculates the majority vote of the hidden outputs.

We assume that both the teacher and the student receive N-dimensional input $\boldsymbol{\xi}^{(m)} = (\xi_1^m, \ldots, \xi_N^{(m)})$ and that the teacher outputs $t^{(m)}$ and the student outputs $s^{(m)}$ are

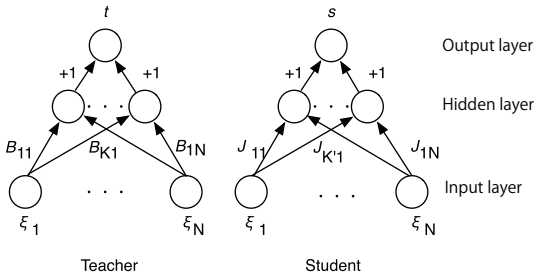

Fig. 1 Network structures of teacher and student.

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

$$s^{(m)} = \sum_{k'=1}^{K'} s_{k'}^{(m)} = \sum_{k'=1}^{K'} g(y_{k'}^{(m)}). \tag{2}$$

Here, $g(\cdot)$ is the output function of a hidden unit, $d_k^{(m)}$ is the inner potential of the kth hidden unit of the teacher, and $y_{k'}^{(m)}$ is the inner potential of the k'th hidden unit of the student calculated as

$$d_k^{(m)} = \sum_{i=1}^{N} B_{ki}\xi_i^{(m)}, \tag{3}$$

$$y_{k'}^{(m)} = \sum_{i=1}^{N} J_{k'i}^{(m)}\xi_i^{(m)}. \tag{4}$$

We assume that the ith elements $\xi_i^{(m)}$ of the independently drawn input $\boldsymbol{\xi}^{(m)}$ are uncorrelated random variables with zero mean and unit variance; that is, the ith element of the input is drawn from a probability distribution $\mathrm{P}(\xi_i)$. The thermodynamic limit of $N \to \infty$ is also assumed. The statistics of the inputs $\boldsymbol{\xi}^{(m)}$ at the thermodynamic limit of $N \to \infty$ are

$$\left\langle \xi_i^{(m)} \right\rangle = 0, \left\langle (\xi_i^{(m)})^2 \right\rangle \equiv \sigma_\xi^2 = 1, \left\langle \|\boldsymbol{\xi}^{(m)}\| \right\rangle = \sqrt{N}, \tag{5}$$

where $\langle\cdot\rangle$ denotes a mean, and $\|\cdot\|$ denotes the norm of a vector.

For each element B_{ki}, $k = 1 \sim K$ is drawn from a probability distribution with zero mean and $1/N$ variance. With the assumption of the thermodynamic limit, the statistics of the teacher weight vector are

$$\langle B_{ki} \rangle = 0, \left\langle (B_{ki})^2 \right\rangle \equiv \sigma_B^2 = \frac{1}{N}, \langle \|\boldsymbol{B_k}\| \rangle = 1.$$

This means that any combination of $\boldsymbol{B}_l \cdot \boldsymbol{B}_{l'} = 0$. The distribution of inner potential $d^{(m)}$ follows a Gaussian distribution with zero mean and unit variance at the thermodynamic limit.

For the sake of analysis, we assume that each element of $J_{k'i}^{(0)}$, which is the initial value of the student vector $\boldsymbol{J}_{k'}^{(0)}$, is drawn from a probability distribution with zero mean and $1/N$ variance. At the thermodynamic limit, the statistics of the k'th hidden weight vector of the student are

$$\left\langle J_{k'i}^{(0)} \right\rangle = 0, \left\langle (J_{k'i}^{(0)})^2 \right\rangle \equiv \sigma_J^2 = \frac{1}{N}, \left\langle \|\boldsymbol{J}_{k'}^{(0)}\| \right\rangle = 1.$$

This means that any combination of $\boldsymbol{J}_l^{(0)} \cdot \boldsymbol{J}_{l'}^{(0)} = 0$. The output function of the hidden units of the student $g(\cdot)$ is the same as that of the teacher. The statistics of the student weight vector at the mth iteration are

$$\left\langle J_{k'i}^{(m)} \right\rangle = 0, \left\langle (J_{k'i}^{(m)})^2 \right\rangle = \frac{(Q_{k'k'}^{(m)})^2}{N}, \left\langle \|\boldsymbol{J}_{k'}^{(m)}\| \right\rangle = Q_{k'k'}^{(m)}.$$

Here,

$$(Q_{k'k'}^{(m)})^2 = \boldsymbol{J}_{k'}^{(m)} \cdot \boldsymbol{J}_{k'}^{(m)}.$$

The distribution of the inner potential $y_{k'}^{(m)}$ follows a Gaussian distribution with zero mean and $(Q_{k'k'}^{(m)})^2$ variance in the thermodynamic limit.

Next, we introduce the stochastic gradient descent (SGD) algorithm for the soft-committee machine. For the possible inputs $\{\boldsymbol{\xi}\}$, we want to train the student to produce the desired outputs $t = s$. The generalization error is defined as the squared error ε averaged over possible inputs:

$$\begin{aligned} \varepsilon_g^{(m)} &= \left\langle \varepsilon^{(m)} \right\rangle = \frac{1}{2}\left\langle (t^{(m)} - s^{(m)})^2 \right\rangle \\ &= \frac{1}{2}\left\langle \left(\sum_{k=1}^{K} g(d_k^{(m)}) - \sum_{k'=1}^{K'} g(y_{k'}^{(m)}) \right)^2 \right\rangle, \end{aligned} \tag{6}$$

At each learning step m, a new uncorrelated input, $\boldsymbol{\xi}^{(m)}$, is presented, and the current hidden weight vector of the student $\boldsymbol{J}_{k'}^{(m)}$ is updated using

$$\begin{aligned} \boldsymbol{J}_{k'}^{(m+1)} =& \boldsymbol{J}_{k'}^{(m)} + \frac{\eta}{N}\left(\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{K'} g(y_{l'}^{(m)}) \right) \\ & \times g'(y_{k'}^{(m)})\boldsymbol{\xi}^{(m)}, \end{aligned} \tag{7}$$

where η is the learning step size and $g'(x)$ is the derivative of the output function of the hidden unit $g(x)$.

On-line learning uses a new input at once; therefore, overfitting does not occur. To evaluate dropout, the learning must exhibit overfitting. To exhibit the overfitting in on-line learning, pre-selected whole inputs are frequently used in an on-line manner. From our experience, when the input dimension is N, then overfitting occurs for less than pre-selected $10 \times N$ learning input data. This assumption is held in this paper.

3. Analysis of Dropout

Dropout is used in deep learning to prevent overfitting[3]. A small amount of data compared with the size of a network may cause overfitting [10]. In the state of overfitting, learning error, which is error from learning data, and error from test data, which is individual from learning data, become different.

The learning equation of dropout for the soft-committee machine can be written as follows.

$$\begin{aligned} \boldsymbol{J}_{k'}^{(m+1)} =& \boldsymbol{J}_{k'}^{(m)} + \frac{\eta}{N}\left(\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l' \notin D^{(m)}}^{(1-p)K'} g(y_{l'}^{(m)}) \right) \\ & \times g'(y_{k'}^{(m)})\boldsymbol{\xi}^{(m)}, \end{aligned} \tag{8}$$

Here, $D^{(m)}$ includes a number of hidden units that are randomly

selected with respect to the probability p from all the hidden units at the mth iteration. Subscript k of the student weight vector $\boldsymbol{J}$ is included in $D^(m)$. Note that the second term in the bracket of R.H.S. of Eq. (8) is a soft-committee machine composed of not selected hidden units. Then, the hidden units in $D^{(m)}$ are not subject to learning, the size of the student decreases, and a shrunken student may avoid overfitting. This effect is the dropout opportunity. After the learning, the student's output $s^{(m)}$ is calculated using the sum of learned hidden outputs and hidden outputs that have not been learned multiplied by p.

$$s^{(m)} = p * \left\{ \sum_{l' \notin D^{(m)}}^{(1-p)K'} g(y_{l'}^{(m)}) + \sum_{l' \in D^{(m)}}^{pK'} g(y_{l'}^{(m-1)}) \right\} \qquad (9)$$

This equation is regarded as the ensemble of a learned soft-committee machine (the first term of R.H.S.) and that of a not learned soft-committee machine (the second term of R.H.S.) when the probability is $p = 0.5$. However, in deep learning, a set of hidden units in $D^{(m)}$ is changed in every iteration where the same set of hidden units are used in the ensemble learning. Therefore, dropout is regarded as ensemble learning using a different set of hidden units in every iteration. Therefore, we refer to dropout as "random dropout" in this paper.

Figure 2 shows the results of the SGD without regularization and those of dropout. The soft committee machine was used for both the teacher and student. $\mathrm{erf}(x/\sqrt{2})$ was used as the output function $g(x)$. We generated $10 \times N$ learning data and N testing data. The input dimension was $N = 1000$. The teacher had two hidden units, and the student had 100 hidden units. The input and its target were generated as those of Fig.5. The learning step size η was set to 0.01.

Figure 2(a) shows the learning curve of the SGD without regularization. In this setting, overfitting will occur. Figure 2(b) shows the learning curve of the SGD with dropout. The learning error was less than the test error; however, the difference between the learning error and the test error was not as substantial as that of the SGD. Therefore, these results show that dropout prevents overfitting.

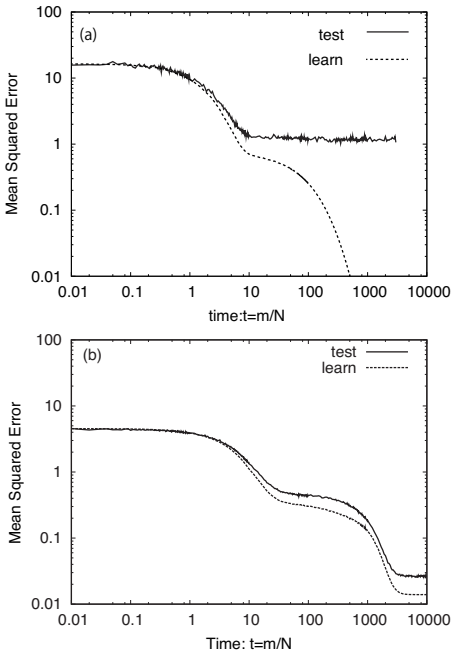

Fig. 2 Effect of dropout. (a) is the learning curve of SGD, and (b) is that of dropout learning.

3.1 Comparison between dropout and SGD with L2 regularization

As Warger et al. pointed out, dropout is an adaptive L2 regularizer[9]. Thus, in this subsection, we present a comparison of dropout and SGD with L2 regularization (refer as SGD with L2).

The next learning equation shows the SGD with L2.

$$\boldsymbol{J}_{k'}^{(m+1)} = \boldsymbol{J}_{k'}^{(m)} + \frac{\eta}{N} \left(\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l'=1}^{K'} g(y_{l'}^{(m)}) \right) \times g'(y_{k'}^{(m)}) \boldsymbol{\xi}^{(m)} - \alpha \|\boldsymbol{J}_{k'}^{(m)}\|^2. \qquad (10)$$

Here, α is a coefficient of the L2 penalty.

In Fig. 3, we show the learning results of the SGD with L2. We used soft-committee machines that include 100 hidden units. For dropout, we set $p = 0.5$. For SGD with L2, we selected $\alpha = 10^{-6}$. The learning step size was set to $\eta = 0.01$. Input data were generated by using eq. (5), and the target was generated by using the teacher. We used $10 \times N$ inputs for learning and N inputs for testing. Training data were frequently used. Results were obtained using an average of 10 trials. The conditions were the same as those of Fig. 7.

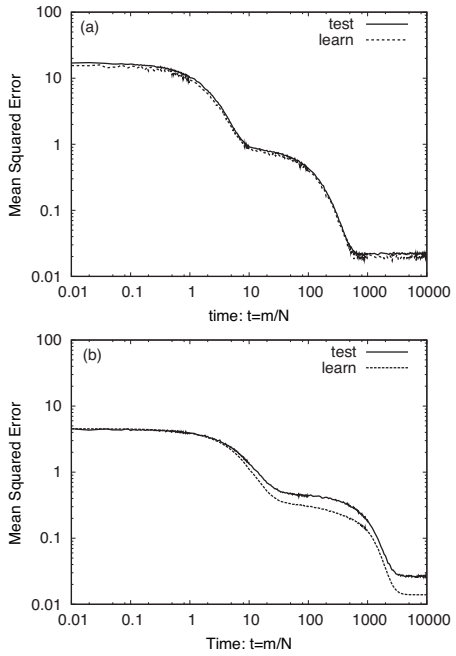

Fig. 3 Learning curve of SGD with L2.

A comparison between Fig. 3 and Fig. 7(b) reveals that the residual error of dropout learning was almost the same as that of the SGD with L2 regularization. Therefore, the regularization effort of dropout learning is the same as the L2. Note that for the SGD with L2, we must choose α in trials; however, dropout learning has no tuning parameter.

Figure 4 shows the time course of the average and variance of the squared norm of the student weight vector $\|\boldsymbol{J}_k\|^2$ that are used

in Fig. 3. The horizontal axes are continuous time $t = m/N$, and the vertical axes are the squared norm of the student weight. The solid lines are the averages of the squared norm of the student weights (refer to "L2 ave" and "Dropout ave"), and the broken lines are those of variances (refer to "L2 vari" and "Dropout vari"). In these figures, the squared norm of the student weight decreases as the learning proceeds for both SGD with L2 and dropout. Therefore, regularization is effective for both methods. The average of the squared norm of L2 regularization is smaller than dropout, so regularization is more effective on SGD with L2 than dropout. However, the variance of the squared norm of dropout is higher than that of SGD with L2. This means that the diversity of hidden units when using dropout is maintained. This may be an advantage for ensemble learning.

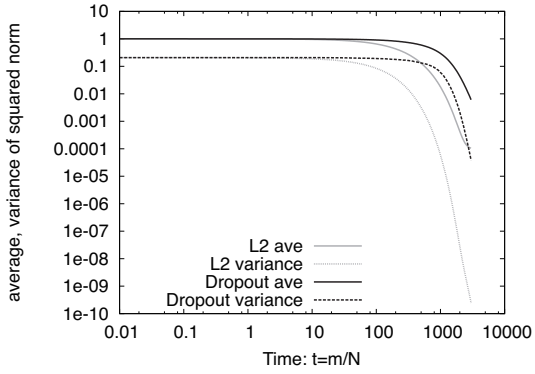

Fig. 4 Squared norm of SGD with L2 and that of dropout.

3.2 Ensemble learning

Ensemble learning is performed by using many learners (referred to as "students") to achieve better performance [5]. In ensemble learning, each student learns from the teacher independently, and each student output $s_{k'_{en}}$ is averaged to calculate the ensemble output s_{en}. We assume that the teacher and the students are the soft-committee machines. Thus, the ensemble output $_{en}$ is calculated by

$$s_{en} = \sum_{k'_{en}=1}^{K_{en}} C_{k'_{en}} s_{k'_{en}} = \sum_{k'_{en}=1}^{K_{en}} C_{k'_{en}} \sum_{k'=1}^{K'} g(y_{k'}). \tag{11}$$

Here, K' is the number of hidden units in the student, $C_{k'_{en}}$ is a weight for averaging, and K_{en} is the number of students to be averaged. The learning equation of ensemble learning is the same as eq. (7).

There are three cases of setting the number of hidden units in the student: (1) $K' < K$, (2) $K' = K$, and (3) $K' > K$. The case of $K' < K$ is unlearnable and insufficient because the degree of complexity of the student is less than that of the teacher. The case of $K' = K$ is learnable because the degree of complexity of the student is the same as that of the teacher. The case of $K' > K$ is learnable and redundant because the degree of complexity of the student is higher than that of the teacher[10]. Therefore, if we set $K' = K$, the network performance will be the best of them.

Figure 5 shows computer simulation results. The student has the same architecture as the teacher, and they include 2 hidden units, that is $K = K' = 2$. The output function $g(x)$ is the error function $\mathrm{erf}(x/\sqrt{2}) = \int_{-x}^{x} dt \exp(-t^2/2)/\sqrt{2\pi}$. We generated $10 \times N$ learning inputs $\boldsymbol{\xi}^{(m)}$ where $N = 10,000$, and they were frequently used. Each of the elements $\xi_i^{(m)}$ of the independently drawn input $\boldsymbol{\xi}^{(m)}$ are uncorrelated random variables with zero mean and unit variance, as shown in eq. (5). We also generated N testing data of cross validation. The target for an input $\boldsymbol{\xi}$ is the output of the teacher. In the figure, the horizontal axis is time $t = m/N$. Here, m is the iteration number, and N is the dimension of input units. The vertical axis is the mean squared error (MSE) for N input data. MSE was calculated for N independent inputs. In the figure, "Single" is the results of using a single student. "m2" is the results of using an ensemble of two students, "m3" is that of an ensemble of three students, and "m4" is that of an ensemble of four students. As shown, the performance of the ensemble improves when a higher number of students is used. Therefore, the ensemble of four students outperformed the other two cases.

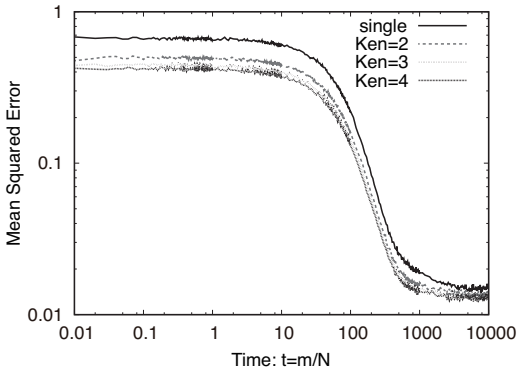

Fig. 5 Effect of ensemble learning.

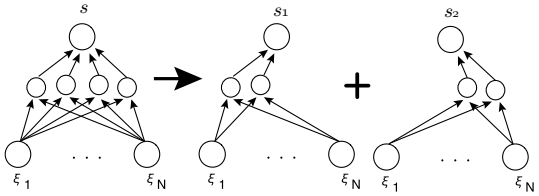

Fig. 6 Network divided into two networks to apply ensemble learning.

Next, we modified the ensemble learning. We divided the student (with K' hidden units) into K_{en} networks (See Fig. 6. Here, $K' = 4$ and $K_{en} = 2$). These sub-networks learned the teacher independently, and then we calculated the ensemble output s_{en} by averaging the outputs of sub-networks $s_{k'_{en}l'}$ as:

$$s_{en} = \frac{1}{K_{en}} \sum_{k'_{en}=1}^{K_{en}} s_{k'_{en}} = \frac{1}{K_{en}} \sum_{k'_{en}=1}^{K_{en}} \sum_{l'=1}^{M} g(y_{k'_{en}l'}). \tag{12}$$

Here, $s_{k'_{en}}$ is the output of a sub-network with M hidden units, and $g(y_{k'_{en}l'})$ is the l'th hidden output in the k'_{en}th sub-network. Eq. (12) corresponds to Eq. (11) when $C_{k'_{en}} = \frac{1}{K_{en}}$ and $K' = M$.

The next section present our comparison of dropout and ensemble learning to clarify the effect of the random selection of hidden units.

3.3 Comparison between random dropout and ensemble learning

We compared dropout and ensemble learning from three viewpoints: (1) selecting the hidden units in a group randomly or using the same hidden units, (2) dividing the student into two or more groups that contain a part of hidden units in the student, and (3) averaging the outputs of learned networks and those of unlearned networks or averaging only the output of learned networks. Dropout involves selecting the hidden units in a group randomly, dividing the student into two groups, and averaging the output of learned hidden units and that of unlearned networks. Ensemble learning involves using the same hidden units in a group throughout the learning, dividing the students into more than two groups, and averaging the output of learned networks. Thus, this subsection concentrates on the effect of selecting the hidden units in a group to be compared randomly.

In comparison, we used two soft-committee machines with 50 hidden units for ensemble learning. For dropout, we used 100 hidden units and set $p = 0.5$; then, dropout selected 50 hidden units in $D^{(m)}$ with 50 unselected hidden units remaining. Therefore, dropout and ensemble learning had the same architectures. The number of input units was $N = 1000$, and the learning step size was set to $\eta = 0.01$. The output function $g(x)$ is a sigmoid-like function $\mathrm{erf}(x/\sqrt{2})$. Input data were generated by using eq. (5), and the target was generated by using the teacher. We used $10 \times N$ inputs for learning and N inputs for testing.

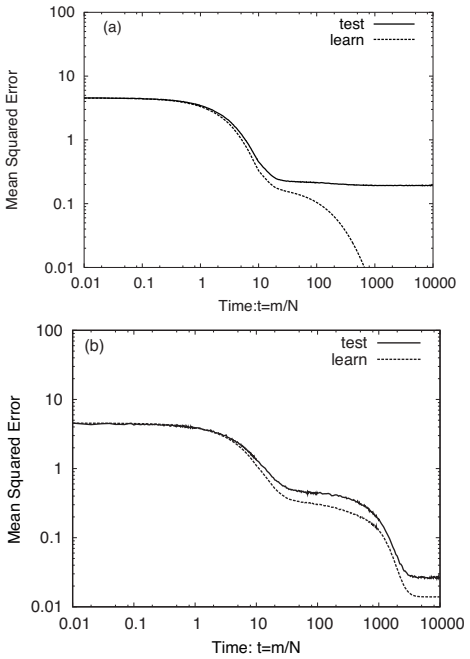

Fig. 7 Results of comparison between dropout and ensemble learning. (a) is ensemble learning of two networks, and (b) is dropout with respect to $p = 0.5$.

Figure 7 shows the results. The results were obtained by taking the average of the results of ten trials. 10^4 independent data were used in the learning, and 10^3 independent data were used to calculate the MSE. The horizontal axes are time $t = m/N$, and the vertical axes are the MSE calculated for N data. In Fig. 7(a), "Single" shows the soft-committee machines with 50 hidden units, and "Ensemble" shows the results given using ensemble learning. Test errors are used in these figures. In Fig. 7(b), "Test" shows the MSE given using the test data, and "Learning" shows the MSE given using the learning data. Dropout was used. From Fig. 7(a), the ensemble learning achieved an MSE less than that of the single network. However, from Fig. 7(b), dropout achieved an MSE less than that of ensemble learning. Therefore, dropout outperforms ensemble learning because it uses randomly selected hidden units and averages the outputs of learned networks and those of unlearned networks.

4. Group dropout

In dropout, two groups of learned hidden units and those of not-learned hidden units are used. This means that dropout uses only two students for ensemble. In ensemble learning, using many students will achieve better performance. Therefore, the effect of the ensemble in dropout is not enough. Therefore, we propose "group dropout," which involves dividing the student into more than two groups composed of hidden units of the student and applying ensemble learning. In group dropout, the number of hidden units in a group is the same as that in the teacher network.

In the proposed method, we selected K hidden units in a group at random from a pool of K' hidden units before the learning was started. Therefore, we had $K_{gd} = K'/K$ groups. (See Fig. 6. Here, $K' = 4$ and $K = 2$). These groups learned from the teacher independently, and then we calculated the ensemble output s_{en} by averaging the group outputs s_{gd} as:

$$\boldsymbol{J}_{k'}^{(m+1)} = \boldsymbol{J}_{k'}^{(m)} + \frac{\eta}{N}\left(\sum_{l=1}^{K} g(d_l^{(m)}) - \sum_{l' \notin D}^{K} g(y_{l'}^{(m)})\right) \times g'(y_{k'}^{(m)})\boldsymbol{\xi}^{(m)}, \tag{13}$$

$$s_{gd} = \frac{1}{K_{gd}} \sum_{k'_{gd}=1}^{K_{gd}} s_{k'_{gd}} = \frac{1}{K_{gd}} \sum_{k'_{gd}=1}^{K_{gd}} \sum_{k'=1}^{K} g(y_{k'_{gd}k'}). \tag{14}$$

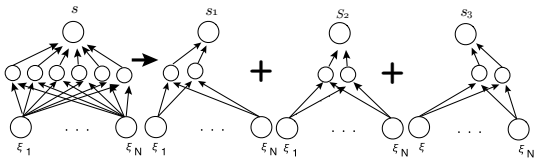

Fig. 8 Student divided into three groups to learn by ensemble learning. Teacher was composed of two hidden units.

Here, $s_{k'_{gr}}$ is the output of a group with K hidden units, and $y_{k'_{gd}k'}$ is the k'th hidden output in the k'_{gd}th group. We set the number of hidden units in a group to K, and this makes the learning of the group network possible.

Figure 9 shows the results. The results were obtained by taking the average of the results of 10 trials. We generated $10 \times N$ learning data and N testing data with respect to eq. (5). The input dimension was $N = 1000$. Learning data were frequently used. In these figures, the horizontal axes show the learning time $t = m/N$. The vertical axes show the MSE. The simulation conditions are the same as those of Fig. 7. The line labeled "Half Dropout" shows the results of the ensemble of two groups of students, and

it uses the same hidden units in a group during the learning. "Half Dropout" is the same as "Random Dropout" except for using the same hidden units in a group and the ensemble of learned soft-committee machines.

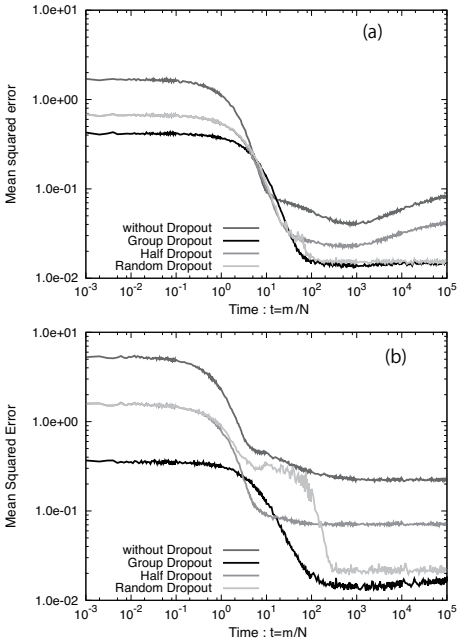

Fig. 9 Effects of proposed method. (a) shows when $K' = 8$, and (b) shows when $K' = 30$. $K = 2$ for both. The results are shown for the average of ten trials.

As can be seen from Fig. 9, the residual error of "Without Dropout" stays high and can be considered as overfitting the data. The residual error of "Half Dropout" is in the middle of "Without Dropout" and "Random Dropout" or "Group Dropout", so dividing the student into two groups and applying ensemble learning does not perform well. The residual error of "Random Dropout" and "Group Dropout" converge to a low value, so these are considered as not overfitting the data. Therefore, both "Random Dropout" and "Group Dropout" can work as a regularizer. As can be seen from these figures, when the number of hidden units is low (Fig. 9(a)), the MSE of the group dropout and that of the random dropout are identical. However, when the number of hidden units is high (Fig. 9(b)), the MSE of the group dropout outperforms that of the random dropout.

The group dropout differs from the random dropout for three reasons. First, dropout divided the student into two groups; however, the proposed method divided the student into more than two groups. Second, the dropout randomly selected the hidden units to be neglected at each learning step; however, the proposed method used the same hidden units in the group. Third, the dropout is the ensemble learning of learned hidden units and unlearned hidden units; however, the group dropout is the ensemble of only learned hidden units.

5. Conclusion

This paper presented our analysis of the dropout regarded as ensemble learning. We showed that the dropout can be regarded as ensemble learning except for when using a different set of hidden units in every learning iteration. This analysis clarified that using a different set of hidden units outperforms ensemble learning. We next proposed using group dropout, which divides the student into several groups with some hidden units that are identical to those of the teacher, and described the performance of ensemble learning. The proposed method outperforms the dropout when the number of hidden units in a group is higher than that of the teacher. Our future work is to clarify the effect of averaging the outputs of learned networks and that of unlearned networks and to explore group dropout using different hidden units in a group.

Acknowledgments

The authors thank Professor Masato Okada and Assistant Professor Hideitsu Hino for their insight and advice. The authors also thank Professor Yutaka Sakai for his advice on comparing regularization performance between dropout and the regularization method.

References

[1] LeCun, Y., Bengio, Y., and Hinton, G., "Deep learning", Nature, vol. 521, pp. 436–444 (2015).
[2] Hinton, G. E., Osindero, S., and Teh, Y. W., "A fast learning algorithm for deep belief nets", Neural Computation, **18**, pp. 1527–1554 (2006).
[3] Krizhevsky, A., I. Sutskever, I., and Hinton, G. E., "ImageNet Classification with Deep Convolutional Neural Networks", Advances in Neural Information Processing Systems 25, (2012).
[4] Freund, Y., and Schapire, R. E., "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting", Journal of Computer and System Science vol. 55, pp. 119–139, (1997).
[5] Hara, K., and Okada, M., "Ensemble Learning of Linear Perceptrons: On-Line Learning Theory", Journal of the Physical Society of Japan, vol. 74, no. 11, pp. 2966–2972 (2005).
[6] Miyoshi, S., Hara, K., and Okada, M., "Analysis of ensemble learning using simple perceptron based on online learning theory", Physical Review E, American Physical Society, (2005).
[7] Biehl, M., and Schwarze, H., "Learning by on-line gradient descent", Journal of Physics A: Mathematical and General Physics, **28**, 643–656 (1995).
[8] Saad, D., and Solla, S. A., "On-line learning in soft-committee machines", Physical Review E, **52**, pp. 4225–4243 (1995).
[9] Wager, S., Wang, S., and Liang, P., "Dropout Training as Adaptive Regularization", Advances in Neural Information Processing Systems 26 (2013).
[10] Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer (2006).