Learning Process of Affordable Neural Network for Backpropagation Algorithm

Yoko Uwate, *Member, IEEE* and Yoshifumi Nishio, *Member, IEEE*

***Abstract*— We have recently proposed a novel neural network structure called an "Affordable Neural Network" (AfNN), in which affordable neurons of the hidden layer are considered as the elements responsible for the robustness property as is observed in human brain function. We have confirmed that the AfNN gains good performance both of the generalization ability and the learning ability. Furthermore, the AfNN has durability, because the AfNN still performs well even if some of neurons in the hidden layer are damaged after learning process. In this study, we study the characteristics of weights of the AfNN during the learning process to make clear the reason of that the AfNNs can perform well for learning and generalization abilities and operate as usually against damaging neurons.**

I. Introduction

The mechanisms of many kinds of higher brain functions become clear with the advances in neuroscience technology. Therefore, the studies of bio-inspired artificial neural networks have been extensively reported to realize such intelligence systems applying for the future engineering applications. However, the performance of artificial neural networks still lags behind that of biological networks in many respects. In order to fill the gap between biological neural systems and artificial neural systems, it is important to apply these high functional mechanisms of the human brain to novel artificial neural networks.

In our previous work, we have proposed a new network structure of the feedforward neural network with affordable neurons in the hidden layer for Backpropagation (BP) learning [1]. We named this network "Affordable Neural Network (AfNN)." The AfNN was inspired by cell assembly [2]-[5]. which is one of explanations of multiple information processing in the brain and an information is represented by a firing space pattern of a group of plural neurons. In the AfNN, we prepare some extra neurons more than requires in the hidden layer. When the AfNN executes operating, all of the neurons in the hidden layer are not used at every updating. Namely, the AfNN is able to operate with high function by using different neuron patterns in the hidden layer. By computer simulations, the AfNN has been confirmed to gain better performance for the BP learning on both learning ability and generalization ability.

On the other hand, am important issue is the understanding how the mammalian brain is affected in its performance by the omnipresent death of their elements in the field of Neuroscience. It is also important to investigate the performance of the artificial neural network when some neurons are damaged. Several research groups have reported the studies of fault tolerance in the artificial neural networks [6]-[8]. In Ref. [6], they have proposed the learning algorithm of the multi-layer perceptron to create robustness. The units in the hidden layer are randomly disabled for some pattern presentations during a standard BP training phase. This learning algorithm seems similar as the learning process of the AfNN. Actually, there are completely difference learning processes. Because, in this learning algorithm of Ref. [6], the disable neurons in the hidden layer are selected "for each pattern." The network learns by using specific neurons in the hidden layer for each input pattern. While, in the case of the AfNN, the affordable neurons are selected "at every updating time." The AfNN learns by using all neurons in the hidden layer with different pattern for every input patterns. Therefore, the learning process of the AfNN is more complex and it has a possibility to create the high functions such as durability and flexibility. In order to confirm that the AfNN can generate a kind of fault tolerance, we also have investigated the durability of the AfNN when some of the neurons in the hidden layer are damaged after the learning process [9]. We have confirmed that the AfNN can operates with keeping its efficiency against damaging neurons.

However, the reason of what property of learning process of the AfNN affect good performance for the learning ability and durability has not yet understood in detail. In this study, we study the characteristics of the weights between the hidden and the output layer of the AfNN during the learning process. First, we investigate an amount of weight change of neurons in the hidden layer during the learning process for the learning ability. Second, we discuss the characteristics of the weight vectors by applying Cosine similarity for the durability. By computer simulations, we observe that the weight vectors of the AfNN and the conventional network have different characteristics. From these results, we conclude that affordable neurons are an important element for the learning ability and the durability against damaging neurons.

II. Affordable Neural Network (AfNN)

We have proposed a network with affordable neurons in the hidden layer of the three-layered feedforward neural network (one input layer, one hidden layer and one output layer) for the BP learning. The output characteristics of the neurons are implemented by a sigmoid function. We introduced the affordable neurons to reflect important properties of the brain. During the BP learning, all of neurons in the

Yoko Uwate is with the Dept. of Electrical and Electronic Engineering, Tokushima University, 2-1 Minami-Josanjima, Tokushima, Japan (email: uwate@ee.tokushima-u.ac.jp).

Yoshifumi Nishio is with the Dept. of Electrical and Electronic Engineering, Tokushima University, 2-1 Minami-Josanjima, Tokushima, Japan (email: nishio@ee.tokushima-u.ac.jp).

978-1-4244-8126-2/10/$26.00

hidden layer are not used at every updating. Namely, some of the neurons are selected for the learning and the rest of the neurons are deactivated. The affordable neurons are selected by random at every updating time. The network model of the AfNN is shown in Fig. 1.

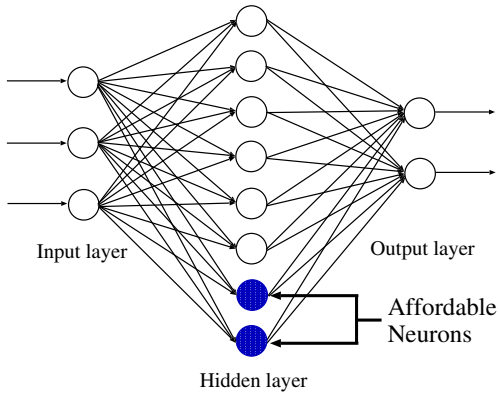


Fig. 1. Affordable neural network.

The standard BP learning algorithm was introduced in [10]. The BP is the most common learning algorithm for the feedforward neural networks. In this study, we use the batch BP learning algorithm. The batch BP learning algorithm is expressed by a formula similar to the standard BP learning algorithm. The difference lies in the timing of the weight. The update of the standard BP is performed after each single input data, while for the bath BP the update is performed after all input data has been processed. The total error E of the network is defined as

$$E = \sum_{p=1}^{P} E_p = \sum_{p=1}^{P} \left\{ \frac{1}{2} \sum_{i=1}^{N} (t_{pi} - o_{pi})^2 \right\}, \qquad (1)$$

where P is the number of the input data, N is the number of the neurons in the output layer, t_{pi} denotes the value of the desired target data for the pth input data, and o_{pi} denotes the value of the output data for the pth input data. The goal of the learning is to set the weights between the neurons so as to minimize the total error E. In order to minimize E, the weights are adjusted according to:

$$\begin{aligned} w_{i,j}^{k-1,k}(m+1) &= w_{i,j}^{k-1,k}(m) + \sum_{p=1}^{P} \Delta_p w_{i,j}^{k-1,k}(m), \\ \Delta_p w_{i,j}^{k-1,k}(m) &= -\eta \frac{\partial E_p}{\partial w_{i,j}^{k-1,k}}, \end{aligned} \qquad (2)$$

were $w_{i,j}^{k-1,k}$ is the weight between the ith neuron of the layer $k-1$ and the jth neuron of the layer k. m is the learning time, and η is a proportionality factor known as the learning rate. In this study, to the third line of Eq.(2) an inertia term was added, which leads to

$$\Delta_p w_{i,j}^{k-1,k}(m) = -\eta \frac{\partial E_p}{\partial w_{i,j}^{k-1,k}} + \zeta \Delta_p w_{i,j}^{k-1,k}(m-1), \qquad (3)$$

where ζ denotes the inertia rate. The inertia term is introduced for efficient learning convergence.

III. Learning Ability of AfNN

First, we investigate the reason of that the AfNN gains the good learning ability by calculating the amount of weight change between the hidden and the output layers during the learning process.

A. Simulation of Learning Ability for $y = x^2$

We consider the task to learn the function $y(x) = x^2$ as a simple learning example. The sampling range of the input data is $[-1.0,\ 1.0]$ and the step size of the input data is set to be 0.01. Our BP learning used fixed learning and inertia rates of $\eta = 0.1$ and $\zeta = 0.01$, respectively; the initial values of the weights were chosen uniformly random from the interval $[-1,\ 1]$ and the learning time is $M = 20000$.

In order to measure the learning ability, we define "Average Error E_{ave}" by the following equation:

$$E_{ave} = \frac{1}{P} \sum_{p=1}^{P} \left\{ \frac{1}{2} (t_p - o_p)^2 \right\} \qquad (4)$$

where P denotes the number of input data ($P = 200$).

In this study, we focus on simulations using hidden layers consisting of 8 neurons with the number of the affordable neurons ranging between 1 and 4. To give an example: If the number of neurons in the hidden layer is 8 and the number of the affordable neurons is set to 2, only 6 neurons operate at all times. Such a network will be denoted as "AfNN $(8-2)$." Below we will compare the performance of an AfNN of type $(n_h - n_{aff})$ with a conventional neural network $(n_h - 0)$ without the affordable neurons.

Figure 2 shows the simulation result of the error curves when the number of the affordable neurons is changed from 1 to 4. When the number of the affordable neurons is set to 1 to 3, the AfNNs gain the better performance than the conventional neural network. While, the case of the number of the affordable neurons is set to 4, the AfNN does not converge to small error. We can see that the AfNNs have good learning ability and the appropriate number of the affordable neuron is exist for given problems.

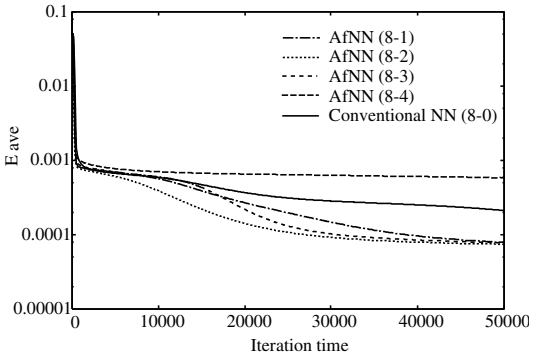


Fig. 2. Typical simulation result of the error curves by using several network types.

B. Amount of Weight Change of AfNN for $y = x^2$

In order to make clear the reason that the AfNN gains better learning ability than the conventional neural network, the characteristics of an amount of weight change during the learning process is investigated. In this study, we focus on the weights between the hidden and the output layers of the AfNN. The equation of the amount of weight change is defined as follows,

$$C_{i,j}^{k-1,k} = \frac{1}{M} \sum_{m=1}^{M} | \sum_{p=1}^{P} \Delta_p w_{i,j}^{k-1,k}(m) | \quad (k = 3). \tag{5}$$

Figure 3 shows the simulation result of the amount of the weight change by using several types of the AfNNs and the conventional neural network. In this simulation result, in order to understand the characteristics of the weight change easily the obtained weight changes are sorted in ascending order after the computer simulation. The results in Fig. 3 are the average of simulations for 100 times by using the different initial conditions of the weight vectors. Form this figure, we can see that the amount of weight change of the AfNNs and the conventional neural network have similar curves when the neuron number is smaller than 4. By increasing the neuron number, their curves become difference and the amount of weight change of the AfNNs show larger value than the conventional neural network.

Table I is summarized the sum of the amount of weight change of the neurons in the hidden layer by using several types of AfNNs. To comparison between the amount of weight change and the network performance, the E_{ave} of each AfNN is also described in Tab. I. The sum of the amount of weight change of the AfNNs are larger than the conventional neural network. The best learning ability (E_{ave}) is obtained by using the AfNN (8-2).

We assume that in the case of the AfNNs the selected affordable neurons does not operate at every updating time, then other operating neurons update their weights a lot. We consider that this characteristics of the learning process is one of good reason of the AfNN.

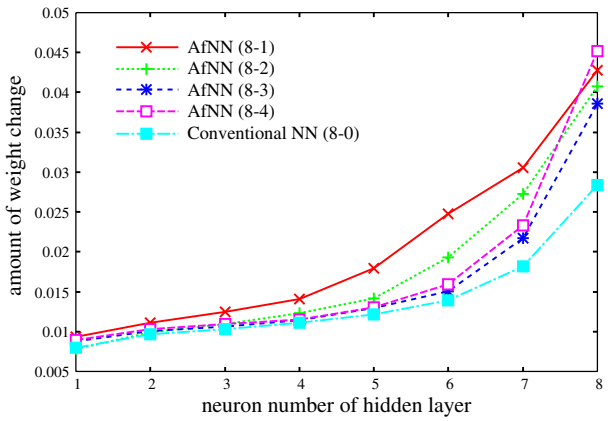


Fig. 3. The amount of weight change by using several types of the AfNNs.

TABLE I

TOTAL AMOUNT OF WEIGHT CHANGE AND E_{ave} (AVERAGE: 100).

Network type	$\sum_{h=1}^{8} C_{i,j}$	Eave
AfNN (8-1)	0.16305	0.624e-3
AfNN (8-2)	**0.14237**	**0.562e-3**
AfNN (8-3)	0.12916	0.586e-3
AfNN (8-4)	0.13918	0.679e-3
Conv. NN (8-0)	0.11161	0.626e-3

IV. DURABILITY OF AFNNS

A. Damaging Neurons

1) Zero output: In order to investigate influence of damaging neurons, we consider that the connections of the damaged neurons to the output layer are temporally cut off (see. Fig. 4). Namely, the damaged neurons do not operate. In this situation, we investigate the performance of the network when the learned data are inputted to the network.

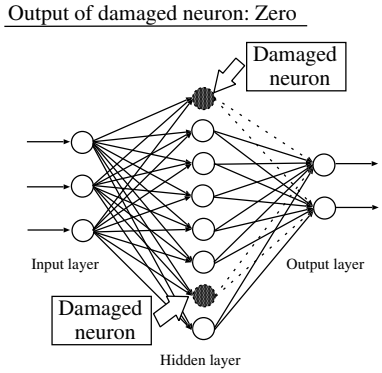


Fig. 4. Conceptual network model for damaging neurons (output: zero).

2) Random output: Next, the connections to output layer of the damaged neurons produce random values from 0.0 to 1.0. The diagram of this network model is shown in Fig. 5. This situation is corresponding to that some neurons are operate randomly without any oder or control.

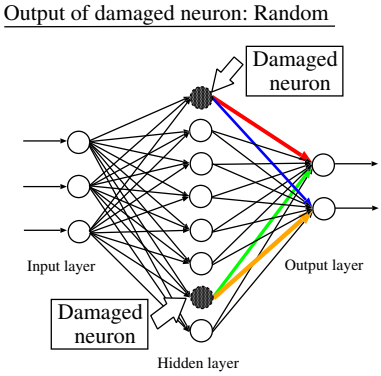


Fig. 5. Conceptual network model for damaging neurons (output: random).

B. Simulation Results of Durability for $y = x^2$

We investigate how after learning damaged neurons affect the total error between the output and the desired target. The learning example is $y = x^2$ and the same parameters of the AfNN are used as above section. The comparison between the AfNN with the conventional network yields the results displayed in Figs. 6, 7. These figures show the averaged results for 8 neurons in the hidden layer, for 100 computer simulations using different random initial condition of the network. When increasing the number of the damaged neurons, E_{ave} of both the AfNN and the conventional BP network worsen. From the comparison of the performance of the two networks, the AfNNs achieve a clearly superior performance to the conventional neural network. From these results, we can see that AfNNs provide an important element of reliability in cases where neurons are affected by damage.

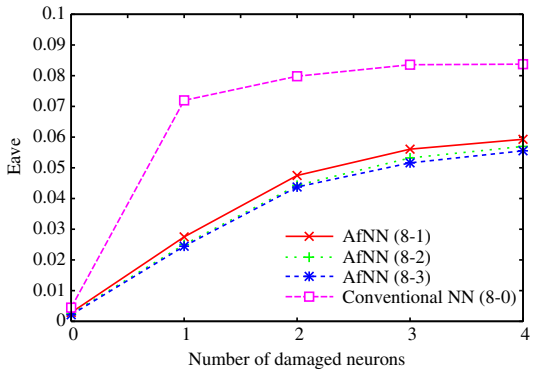


Fig. 6. E_{ave} dependence on the number of damaged neurons (Zero output).

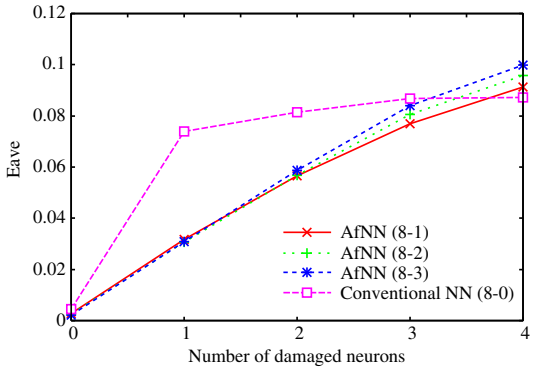


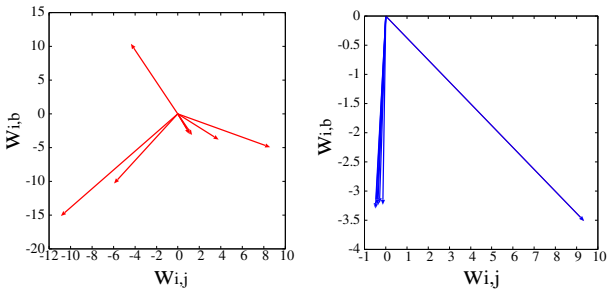
Fig. 7. E_{ave} dependence on the number of damaged neurons (Random output).

C. Characteristics of Weight Vectors for $y = x^2$

In order to make clear the reason that the AfNN gains the better performance than the conventional NN against for the damaging neurons, we focus on the characteristics of the weight vectors between the hidden an the output layers after learning process. We use the three-layered feedforward neural network and there is one neuron in the output layer for learning $y = x^2$, then the weight vector of ith neuron in the hidden layer is composed of two scalar.

$$\mathbf{w}_{i,j}^{2,3} = \{w_{i,1}^{2,3}, w_{i,b}^{2,3}\}, \tag{6}$$

where, $w_{i,1}^{2,3}$ means the weight between the ith neuron of the hidden layer (2nd layer) and the 1st neuron of the output layer (3rd layer), and $w_{i,b}^{2,3}$ denotes the basis of ith neuron in the hidden layer. Figure 8 shows one typical example of the weight vectors of the AfNN and the conventional NN. In the case of the AfNN (Fig. 8 (a)), each vector spreads in the vector space. While, in the case of the conventional NN (Fig. 8 (b)), obtained weight vectors shows almost same values.



(a) AfNN (8-4). (b) Conventional NN (8-0).

Fig. 8. Weight vectors after learning process (Hidden layer: 8). horizontal axis: the weights between the hidden and the output layers. vertical axis: the bias vectors of the neurons in the hidden layer.

We compare the characteristics of weight vectors between the hidden layer and the output layer by using Cosine similarity defined as following equation (e.g. ath and bth neurons in the hidden layer).

$$W_{s(a,b)} = cos(\theta) = \frac{\mathbf{w_{a,j}} \cdot \mathbf{w_{b,j}}}{\|\mathbf{w_{a,j}}\| \|\mathbf{w_{b,j}}\|} \tag{7}$$

The resulting similarity ranges from -1 meaning exactly opposite, to 1 meaning exactly same, with 0 indicating independence.

The 100 averaged results of similarities of the AfNN (8-4) and the conventional network (8-0) are described as following matrices.

Weight vector similarity of the AfNN (8-4):

$$W_s = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} \mathbf{1.00} & 0.24 & 0.18 & 0.12 & 0.26 & 0.24 & 0.15 & 0.21 \\ 0.24 & \mathbf{1.00} & 0.25 & 0.26 & 0.40 & 0.34 & 0.28 & 0.40 \\ 0.18 & 0.25 & \mathbf{1.00} & 0.20 & 0.35 & 0.32 & 0.23 & 0.33 \\ 0.12 & 0.26 & 0.20 & \mathbf{1.00} & 0.31 & 0.20 & 0.07 & 0.18 \\ 0.26 & 0.40 & 0.35 & 0.31 & \mathbf{1.00} & 0.48 & 0.34 & 0.36 \\ 0.24 & 0.34 & 0.32 & 0.20 & 0.48 & \mathbf{1.00} & 0.24 & 0.29 \\ 0.15 & 0.28 & 0.23 & 0.07 & 0.34 & 0.24 & \mathbf{1.00} & 0.26 \\ 0.21 & 0.40 & 0.33 & 0.18 & 0.36 & 0.29 & 0.26 & \mathbf{1.00} \end{pmatrix} \end{array}$$

Norm of similarity matrix: **3.526**

Weight vector similarity of the conventional NN (8-0):

$$W_s = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} \mathbf{1.00} & 0.43 & 0.38 & 0.43 & \mathbf{0.51} & 0.40 & 0.38 & 0.45 \\ 0.43 & \mathbf{1.00} & 0.40 & \mathbf{0.51} & 0.49 & 0.42 & 0.42 & 0.46 \\ 0.38 & 0.40 & \mathbf{1.00} & 0.49 & \mathbf{0.59} & \mathbf{0.55} & 0.47 & 0.44 \\ 0.43 & 0.51 & 0.49 & \mathbf{1.00} & \mathbf{0.62} & 0.42 & 0.34 & 0.39 \\ \mathbf{0.51} & 0.49 & \mathbf{0.59} & \mathbf{0.62} & \mathbf{1.00} & \mathbf{0.56} & 0.44 & 0.45 \\ 0.40 & 0.42 & \mathbf{0.55} & 0.42 & \mathbf{0.56} & \mathbf{1.00} & \mathbf{0.50} & 0.40 \\ 0.38 & 0.42 & 0.47 & 0.34 & 0.44 & \mathbf{0.50} & \mathbf{1.00} & 0.40 \\ 0.45 & 0.46 & 0.44 & 0.39 & 0.45 & 0.40 & 0.40 & \mathbf{1.00} \end{pmatrix} \end{array}$$

Norm of similarity matrix: **4.447**

In these matrices, the element shows the similarity W_s in all combinations and the index number is corresponding to the neuron number in the hidden layer. We can see that the similarity of the conventional NN has large value (e.g. larger than 0.5). Namely, each weight vector of the conventional network has similar value. In order to evaluate these matrices, we calculate the norm and the value of norm are described under the matrices. The norm value of the AfNN is smaller than the conventional network.

From these results, in the case of the AfNN, each weight vector becomes difference charactor. Because the network leans by using different neuron pattern at every updating time. We consider that this is one of the reason of the AfNN to obtain good performance against the damaging neurons.

V. Simulated Results for Pattern Recognition

As more demanding example for the BP learning, we consider a pattern recognition task, where 4 alphabetical letters (**B**, **U**, **C**, **S**) are fed into the neural network for recognition (see Fig. 9).

In this case, the number of neurons in the input layer is 35 The number of neurons in output layer is fixed to 4, and we choose 16 hidden layer neurons. For recognition, a set of 25 patterns shifted 1 bit from each original pattern was prepared, leading a set of 100 patterns to be recognized. The learning rate and the inertia rate were set to $\eta = 1.0$ and $\zeta = 0.8$, respectively, and the initial weights were uniformly randomly chosen from the interval $[-1, 1]$. The learning time was set to $m = 10000$ steps. By using 100 different network initial conditions, we obtained reliable averages characterizing the network performance in terms of the recognition rate R defined as

$$R = \frac{Number\ of\ success}{Total\ number\ of\ input\ patterns} \times 100[\%]. \quad (8)$$

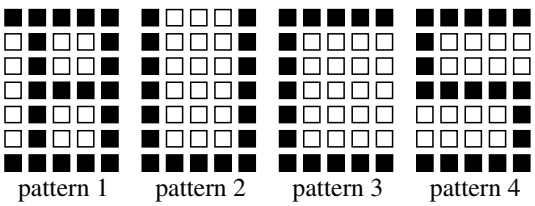


Fig. 9. Pattern recognition.

A. Performance of AfNN against Damaging Neurons

Figures 10, 11 display the recognition rate for the AfNNs $(16-2)$, $(16-10)$, $(16-12)$, $(16-14)$ and conventional BP networks $(16-0)$, $(4-0)$. From these figures, we infer that for this task, the recognition rates of the AfNNs and the conventional networks are similar, if only one neuron is damaged. Upon increasing the number of damaged neurons, the difference between the recognition rates, however, increases, leading to a noticeable higher recognition rate of the AfNNs $(16-10)$, $(16-12)$, where the difference of recognition rate between the both networks is larger than 10 percent. From these results, we can see that the number of affordable neurons has important role for the network performance.

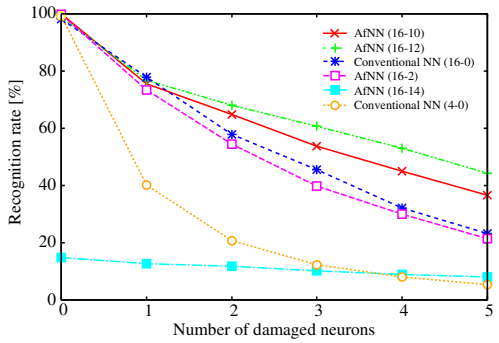


Fig. 10. Recognition rate against damaging neurons (Zero output).

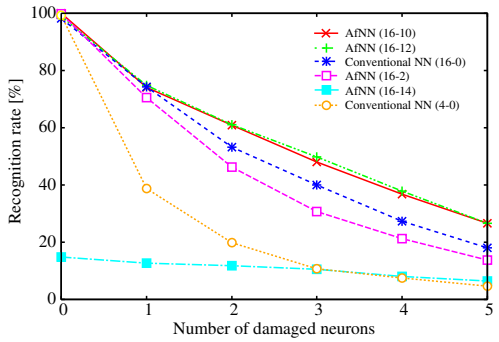


Fig. 11. Recognition rate against damaging neurons (Random output).

B. Characteristics of Weight Vectors for Pattern Recognition

For learning this pattern recognition, 4 neurons are prepared in the output layer. The weight vector of ith neuron in the hidden layer is composed of five scalar as following equation.

$$\mathbf{w}_{i,j}^{2,3} = \{w_{i,1}^{2,3}, w_{i,2}^{2,3}, w_{i,3}^{2,3}, w_{i,4}^{2,3}, w_{i,b}^{2,3}\}, \quad (9)$$

where, $w_{i,1}^{2,3}$, $w_{i,2}^{2,3}$, $w_{i,3}^{2,3}$ and $w_{i,4}^{2,3}$ mean the weight between the ith neuron of the hidden layer (2nd layer) and the 1st, 2nd, 3rd and 4th neurons of the output layer (3rd layer), respectively, and $w_{i,b}^{2,3}$ denotes the basis of ith neuron in the hidden layer.

$W_s =$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1.000	0.105	0.256	0.092	0.136	0.101	−0.031	0.017	0.007	0.010	−0.024	0.036	0.015	−0.055	−0.035	−0.051
2	0.105	1.000	0.071	0.111	0.044	0.092	−0.008	0.029	0.033	−0.014	−0.043	0.011	−0.040	0.065	−0.021	−0.070
3	0.256	0.071	1.000	0.134	0.112	0.231	0.022	−0.002	−0.022	0.032	−0.024	0.019	−0.013	−0.031	−0.008	−0.004
4	0.092	0.111	0.134	1.000	0.076	0.146	−0.059	−0.025	−0.047	0.017	−0.057	0.011	0.054	0.044	−0.011	−0.012
5	0.136	0.044	0.112	0.076	1.000	0.115	0.004	0.015	−0.008	−0.007	0.076	−0.067	−0.024	−0.032	0.003	−0.010
6	0.101	0.092	0.231	0.146	0.115	1.000	−0.028	−0.022	0.002	−0.015	0.033	0.068	0.054	0.070	−0.024	−0.099
7	−0.031	−0.008	0.022	−0.059	0.004	−0.028	1.000	−0.072	0.039	0.087	−0.050	0.060	0.042	−0.038	0.094	0.067
8	0.017	0.029	−0.002	−0.025	0.015	−0.022	−0.072	1.000	0.015	0.004	−0.030	0.025	−0.023	0.023	−0.005	0.009
9	0.007	0.033	−0.022	−0.047	−0.008	0.002	0.039	0.015	1.000	−0.007	−0.093	0.037	−0.030	0.049	−0.046	0.006
10	0.010	−0.014	0.032	0.017	−0.007	−0.015	0.087	0.004	−0.007	1.000	−0.051	−0.067	−0.063	0.015	0.002	0.023
11	−0.024	−0.043	−0.024	−0.057	0.076	0.033	−0.050	−0.030	−0.093	−0.051	1.000	−0.029	−0.025	0.030	0.001	0.028
12	0.036	0.011	0.019	0.011	−0.067	0.068	0.060	0.025	0.037	−0.067	−0.029	1.000	0.015	−0.020	0.076	0.031
13	0.015	−0.040	−0.013	0.054	−0.024	0.054	0.042	−0.023	−0.030	−0.063	−0.025	0.015	1.000	0.035	−0.032	−0.030
14	−0.055	0.065	−0.031	0.044	−0.032	0.070	−0.038	0.023	0.049	0.015	0.030	−0.020	0.035	1.000	−0.065	−0.026
15	−0.035	−0.021	−0.008	−0.011	0.003	−0.024	0.094	−0.005	−0.046	0.002	0.001	0.076	−0.032	−0.065	1.000	−0.065
16	−0.051	−0.070	−0.004	−0.012	−0.010	−0.099	0.067	0.009	0.006	0.023	0.028	0.031	−0.030	−0.026	−0.065	1.000

(a) Matrix of similarity for the AfNN (16-10).

$W_s =$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1.000	0.185	0.292	0.244	0.246	0.192	0.244	0.203	0.201	0.229	0.219	0.186	0.211	0.229	0.246	0.230
2	0.185	1.000	0.179	0.180	0.146	0.188	0.167	0.260	0.266	0.217	0.204	0.229	0.141	0.243	0.249	0.157
3	0.292	0.179	1.000	0.241	0.256	0.280	0.204	0.265	0.283	0.326	0.187	0.232	0.296	0.227	0.254	0.245
4	0.244	0.180	0.241	1.000	0.156	0.174	0.223	0.116	0.219	0.263	0.217	0.223	0.219	0.197	0.229	0.166
5	0.246	0.146	0.256	0.156	1.000	0.215	0.178	0.205	0.235	0.189	0.290	0.182	0.187	0.128	0.227	0.189
6	0.192	0.188	0.280	0.174	0.215	1.000	0.221	0.230	0.241	0.266	0.187	0.269	0.258	0.202	0.264	0.224
7	0.244	0.167	0.204	0.223	0.178	0.221	1.000	0.118	0.203	0.226	0.228	0.223	0.276	0.254	0.202	0.245
8	0.203	0.260	0.265	0.116	0.205	0.230	0.118	1.000	0.147	0.210	0.180	0.217	0.190	0.220	0.229	0.189
9	0.201	0.266	0.283	0.219	0.235	0.241	0.203	0.147	1.000	0.231	0.245	0.170	0.206	0.231	0.236	0.209
10	0.229	0.217	0.326	0.263	0.189	0.266	0.226	0.210	0.231	1.000	0.221	0.226	0.191	0.205	0.183	0.162
11	0.219	0.204	0.187	0.217	0.290	0.187	0.228	0.180	0.245	0.221	1.000	0.233	0.182	0.275	0.194	0.234
12	0.186	0.229	0.232	0.223	0.182	0.269	0.223	0.217	0.170	0.226	0.233	1.000	0.216	0.253	0.240	0.236
13	0.211	0.141	0.296	0.219	0.187	0.258	0.276	0.190	0.206	0.191	0.182	0.216	1.000	0.186	0.188	0.225
14	0.229	0.243	0.227	0.197	0.128	0.202	0.254	0.220	0.231	0.205	0.275	0.253	0.186	1.000	0.174	0.201
15	0.246	0.249	0.254	0.229	0.227	0.264	0.202	0.229	0.236	0.183	0.194	0.240	0.188	0.174	1.000	0.202
16	0.230	0.157	0.245	0.166	0.189	0.224	0.245	0.189	0.209	0.162	0.234	0.236	0.225	0.201	0.202	1.000

(b) Matrix of similarity for the conventional NN (16-0).

Fig. 12. Matrices of weight vector similarity for pattern recognition (hidden:16).

The 100 average results of similarity of the AfNN (16-10) and the conventional network (16-0) are described in the matrices of Fig. 12. The many similarity value of the AfNN are smaller than the conventional network.

Table II is summarized the norm of weight vector similarity which is obtained from 100 average simulations by calculating similarity of each weight vector between the hidden and the output layers after learning process. We can see that norm value of the AfNNs are smaller than the conventional NN. This results show the same as previous task which is function approximation problem ($y = x^2$).

TABLE II

NORM OF SIMILARITY MATRIX.

Network type	Norm
AfNN (16-2)	5.151
AfNN (16-4)	4.823
AfNN (16-6)	4.505
AfNN (16-8)	4.273
AfNN (16-10)	4.108
Conventional NN (16-0)	5.253

Finally, we investigate the relationship between norm of similarity matrix and several recognition rate. The simulation result by changing the number of affordable neuron is shown in Fig. 13. The recognition rate before damage is almost same, however, norm value decreases by increasing the number of the affordable neuron. The recognition rates after 2 neurons are damaged (zero output and random output) gain good performance when the number of the affordable neurons ranges from 10 to 12.

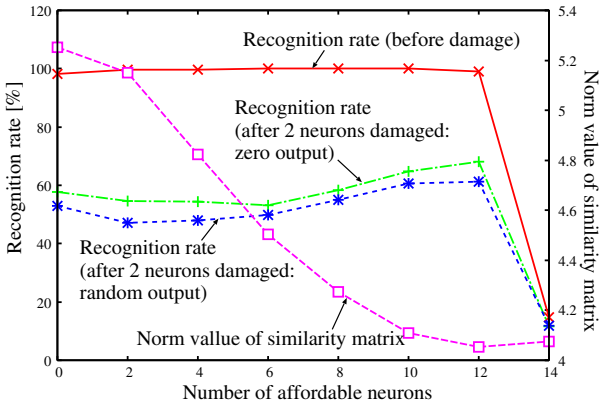


Fig. 13. Relationship between norm of similarity matrix and several recognition rates.

VI. Conclusions

In this study, we have investigated the characteristics of the weight vectors between the hidden and the output layer after the learning process by using the amount of weight change and the weight vector similarity. We confirmed that the property of the weights of the AfNN and the conventional network have different characteristics. We conclude that such weight vector property obtained from learning process is one of the reason that the AfNN performs well for learning ability and durability against damaging neurons.

For the future works, we apply the theoretical or systematic approaches to analyze the characteristics of the AfNNs and use more practical task as the learning example.

References

[1] Y. Uwate and Y. Nishio, "Performance of Affordable Neural Network for Back Propagation Learning," *IEICE Transactions on Fundamentals*, vol. E89-A, no. 9, pp. 473-478, Nov. 2005.

[2] Y. Sakurai, "Dependence of Functional Synaptic Connections of Hippocampal and Neocortical Neurons on Types of Memory," *Neuroscience Letters*, vol. 158, pp. 181-184, 1993.

[3] E. Vaadia, I. Haalman, M. Abeles, H. Bergman, Y. Prut, H. Slovin and A. Aertsen, "Dynamics of Neural Interactions in Monkey Cortex in Relation to Behavioral Events," *Nature*, vol. 373, pp. 515-518, Feb. 1995.

[4] Y. Sakurai, "Hippocampal and Neocortical Cell Assemblies Encode Memory Processes for Different Types of Stimuli in the Rat," *Journal of Neuroscience*, vol. 16, pp. 2809-2819, 1996.

[5] Y. Sakurai, "How Do Cell Assemblies Encode Information in the Brain?," *Neuroscience and Biobehavioral Reviews*, vol. 23, pp.785-796. 1999.

[6] C. H. Sequin and R. Clay, "Fault Tolerance in Artificial Neural Networks," *Proc. IEEE-INNS'90*, pp. 703-708, June. 1990.

[7] W. Hsieh and B. Sher, "Fault Tolerant Capability of Multi-Layer Perceptron Neural Network," *EUROMICRO'94*, pp. 644-650, 1994.

[8] V. Piuri, "Analysis of Fault Tolerance in Artificial Neural Networks," *Journal of Parallel and Distributed Computing*, vol. 61, pp. 18-48, Jan. 2001.

[9] Y. Uwate, Y. Nishio and R. Stoop, "Durability of Affordable Neural Networks against Damaging Neurons," *IEICE Transactions on Fundamentals*, vol. E92-A, no. 2, pp. 585-593, Feb. 2009.

[10] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing*, vol. 1, MIT Press, MA, pp. 318-362, 1986.