# 論文 / 著書情報
# Article / Book Information

| | |
|---|---|
| 題目(和文) | マルチタスク学習過程における階層型ニューラルネットワークの構造解析と形成 |
| Title(English) | Structure Analysis and Creation of Hierarchical Neural Network in Multi-Task Learning Process |
| 著者(和文) | 柴田淳司 |
| Author(English) | Atsushi Shibata |
| 出典(和文) | 学位:博士（学術）,<br>学位授与機関:東京工業大学,<br>報告番号:甲第9902号,<br>授与年月日:2015年3月26日,<br>学位の種別:課程博士,<br>審査員:廣田　薫,柴田　崇徳,室伏　俊明,長谷川　修,董　芳艶 |
| Citation(English) | Degree:,<br>Conferring organization: Tokyo Institute of Technology,<br>Report number:甲第9902号,<br>Conferred date:2015/3/26,<br>Degree Type:Course doctor,<br>Examiner:,,,, |
| 学位種別(和文) | 博士論文 |
| Type(English) | Doctoral Thesis |

# Structure Analysis and Creation of Hierarchical Neural Network in Multi-Task Learning Process

# (マルチタスク学習過程における階層型ニューラルネットワークの構造解析と形成)

Atsushi Shibata

Supervisor: Prof. Kaoru Hirota

Doctoral Thesis

Department of Computational Intelligence and Systems Science
Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology

# Abstract

Hierarchical neural network is analyzed by force directed model, and is created by task-related pruning method, in multi-task learning process.

The structure analysis monitors the learning process by checking the task-wise learning progress, and the creation removes neurons/connections by keeping the content of learned multi-task.

An experiment using MNIST dataset shows that neuron-clusters represent the structure state by 15% higher distribution of a learning task's cluster than that of other learned tasks' clusters, and an experiment using UCI dataset confirms that neurons are deleted without disturbing the learning and 80% of neurons are belonged to the corresponding task.

The presented methods provide foundations of real time learning and unsupervised learning in a robot behavior learning and its status monitoring in real time.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Neural network, one of learning algorithm in the machine learning area, is used as classification, regression, and clustering method [1, 2, 3]. In particular, large-scale neural network are becoming major with its establishing technique of deep learning, and it realizes high accuracy in a research area of image classification area [4, 5]. In response to this result, a study to speed up the learning of the large-scale neural network is conducted flourishingly, e.g. development a computer chip dedicated the large-scale neural network by company [6, 7], and improvement the learning algorithm [8, 9].

Let us aim to adapt these large-scale neural network researches into a more general issue, likes human's object recognition. Human unconsciously takes out important information from the enormous amount of information. For example, when human recognize what is in a picture, he/she watches that it shows red, rounded, and some kind of fruits, or might think it is photo. Finally, although, he/she recognizes it is an apple, ignoring miscellaneous information. In this way, it is necessary for general issue to sort the essential information from a task which also looks like a single task.

There are some similar situation studies, multi-task learning [10, 11] and domain adaptation [12]. Frame problem [13] is also close from another view point. In these research areas, there are common that multiple tasks are existing, and information, s. t.,

similarity and tasks' relationship, is used to help the next task learning. Some neural network researches [11, 14] approach it by 2 steps, one is feature extraction from input data, and the other is learning with extracted data from step 1. These approaches also resolve the multiple task problem, but neural network's learning procedure and structure are constrained by clearly division. In this reason, it is hard to learn with one neural network with comprehensive learning process.

In this thesis, acquired functions in neural network are confirmed by information from inside s. t., number of neurons, weight of connection, and structure, without extra information other than right and wrong.

In the chapter 2, an algorithm of the neural network which is used in this thesis is introduced, and thesis concept is explained which is came from a relationship between neural network structure and its function acquired in learning process is discussed via comparison by biological neural network.

In the chapter 3, hierarchical force-directed graph drawing is proposed for neural network structure analysis by creating neuron cluster in 2D Euclidian spaces, in which the placement of neurons are determined using weights of connections between neurons. In detail, attractive force is acting on neurons locating on two adjacent layers, and repulsive force is acting on neurons inside each layer. Applied force between neurons is used to update lactations of the neurons during iteration, and constructed distribution of neurons are defined as the neuron cluster. The averaged variance of the neurons in clusters which reacts to the input data is designed as criteria for adjusting the number of neurons in each layer.

By applying the proposal, those who do not have enough know-how for designing the

structure of a neural network are able to check the clusters of visualized neurons and use the variance of that as the criteria for determining the neural network structure. Given a specific task, the trial-and-error process for determining its network structure also benefits from lower computational cost.

In the chapter 4, we propose a pruning method and its visualization method. The proposed pruning decomposes neural network structures that correspond to each task, and the proposed visualization shows its relationship via the arrangement of neurons in 2D space. Concretely, rewards and lifetimes are defined for pruning purposes. The generated reward is only given to the neurons that are related to the generation of reward. The lifetime of neurons is updated using the given reward so that neurons and connections that hinder the generation of rewards are removed. The concept of neuron space is also defined and neurons are allocated in the neuron space. Concretely speaking, neurons converge or spread according to their mutual relevance so that structures of neural networks related to specific tasks are displayed in 2D neuron space. Using the pruning method of rewards and lifetimes, structures that are locally suited from a composite task to individual decomposed tasks are likely to be formed. Also, by viewing the placement of neurons in neuron space after learning, the network structure of a composite task is understood intuitively.

The dissertation of this thesis is shown in Fig. 1.1.

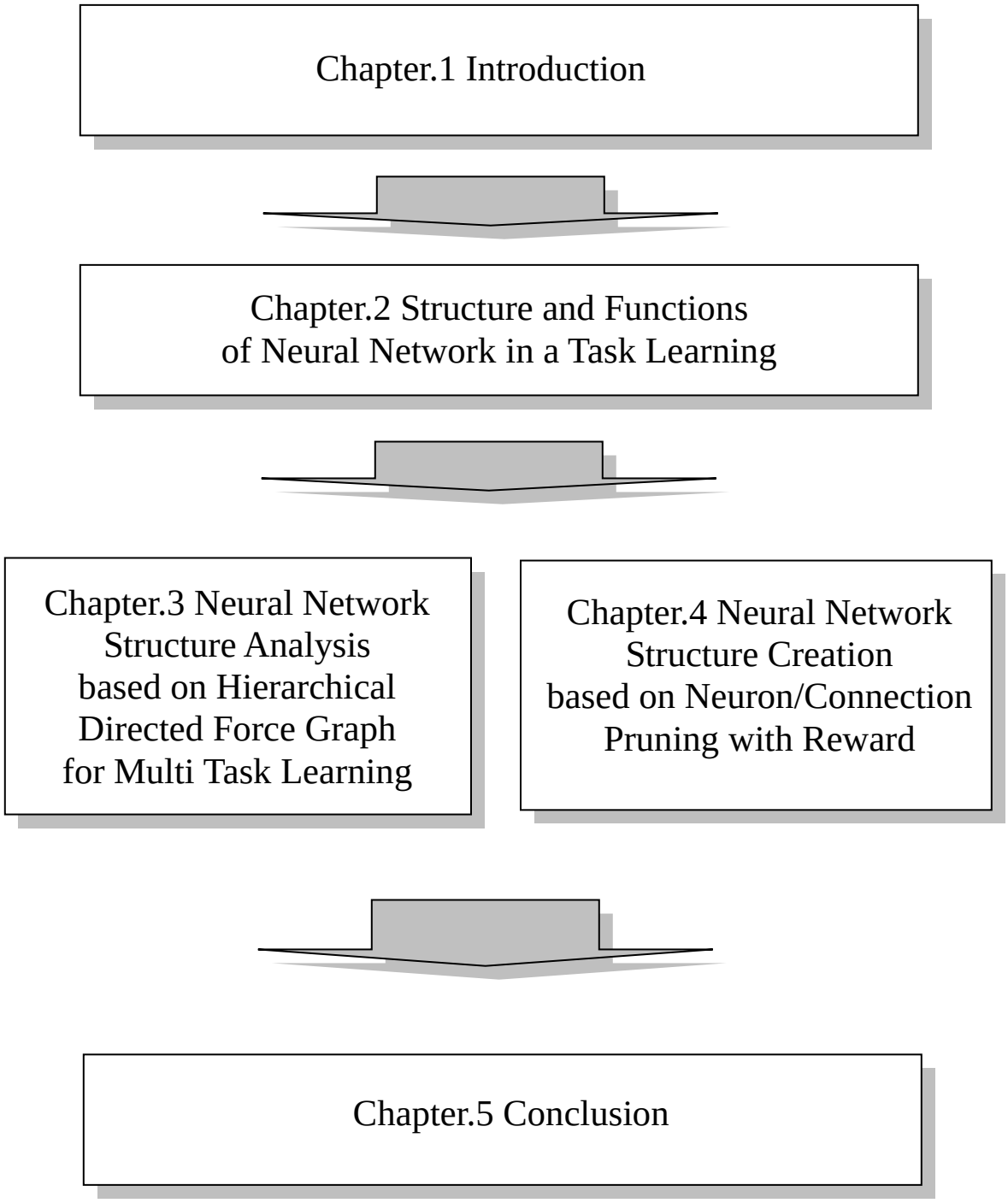



Fig. 1-1 A roadmap of dissertation

# Chapter 2

# Structure and Functions

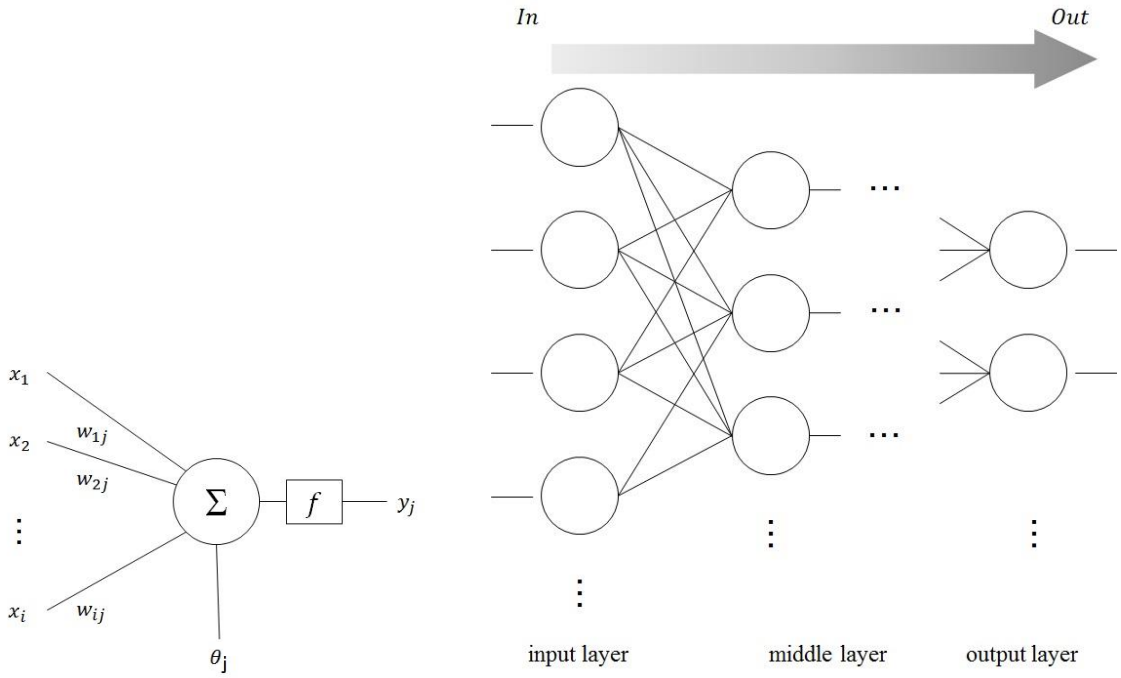
# of Neural Network in a Task Learning

In this chapter, an algorithm of the neural network which is used in this thesis is introduced, and a relationship between neural network structure and its function acquired in learning process is discussed via comparison by biological neural network

## 2.1 Behavior and learning of hierarchical neural network

Neural network is a mathematical model inspired by biological neuron's network. There are innumerable kinds of neural network defined by neuron behavior and network structure [1, 2, 15]. In this thesis, hierarchical neural network is used ad model and backpropagation algorithm is used as learning method, which are one of general neural network styles.

Hierarchical neural network, used in this thesis, is shown in fig 2-1. Each neuron behaves based on a simple rule, which a neuron, numbered $j$, fires if the summation of input signals $x_i$ exceeds the threshold $\theta_j$, and it is expressed by equation




(a) Behavior of neuron *j* (b) Overall graph of hierarchical neural network

**Fig. 2-1** One neuron and hierarchical neural network structure

$$u_j = \sum_{i=1}^{n} x_i w_{ij} \quad , \tag{1.1}$$

$$y_j = f\left(u_j + \theta_j\right) \ . \tag{1.2}$$

Input signal $x_i$ weighted by $w_{ij}$ to neuron $j$ is summed to the internal value $u_i$. This $u_i$ is biased by a previously established threshold $\theta_j$, and output to next neuron through an activation function $f$. A sigmoid function is used as $f$ in this thesis. $w_{ij}$, $\theta_j$, $f$,

and structure constituted by them are neural network parameters and adjusted by a learning algorithm expect structure.

Backpropagation is used as learning algorithm for adjusting $w_{ij}$ and $\theta_j$. In the output layer, $w_{ij}$ is adjusted by

$$w_{ij} \coloneqq w_{ij} + LR \cdot e_j \cdot x_i \quad , \tag{1.3}$$

$$e_j = y_j \cdot (1 - y_j) \cdot (y_j^* - y_j) \quad , \tag{1.4}$$

$LR$ is learning rate of neural network, and $e_j$ is a error value calculated by difference between the desired output $y_j^*$ and the actual output $y_j$. In the middle layer, $w_{ij}$ is adjusted by

$$e_j = y_j \cdot (1 - y_j) \cdot (y_j^* - y_j) \quad , \tag{1.5}$$

$$w_{ij} \coloneqq w_{ij} + (1 - m) \cdot LR \cdot e_j \cdot x_i + m \cdot \left( w_{ij} - w_{ij}' \right) \quad . \tag{1.6}$$

The $m$ is a momentum factor for avoiding local solution by affecting ex weight value $w_{ij}'$.

Stacked denoising autoencoder [16, 17], one of deep learning model, is used when numbers of layers/neurons are big. Learned stacked neural network behaves as same as hierarchical neural network, however, it require pre-training before learning term, shown Fig. 2-2.

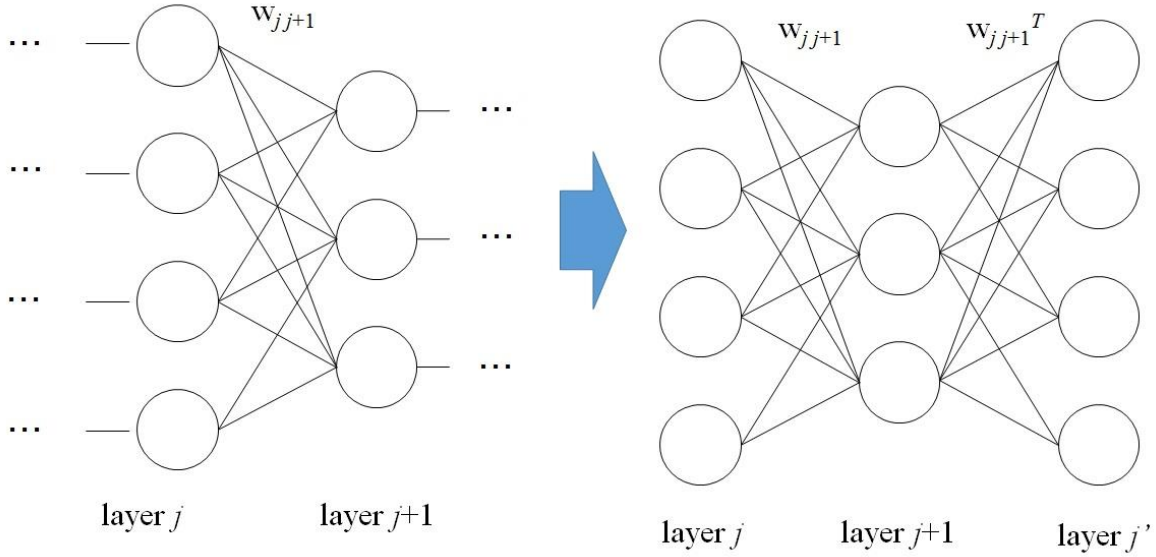



**Fig. 2-2** Two layers of stacked denoising autoencoder are picked up in pre-training term.

In the pre-training term of stacked denoising autoencoder, two layers are picked up in order from input layer side. One of picked up layers in side of input is copied like a mirror, and then weights are learned with the backpropagation method using input signals as desired output signals of neurons in copied layer. In example of **Fig.2-2**, weights and thresholds of neurons in layer *j* are copied as layer *j*'. Input signals is set to layer *j*, and then propagates to layer *j*' via layer *j*+1. After that, weight and thresholds are adjusted by backpropagation with the input signal as the desired output signal. Through this process, the signal is restored after simplified by less neurons in layer *j*+1, which means features in input signal are extracted by this flow.

## 2.2 Knowledge from artificial neural network and biological neural network with deep architecture

During the process of deep learning, it is confirmed that feature extraction is automatically carried out by the neural network [18].

A living neuron has its own functionality which is different from those of other neurons. For example, the neurons reacting to specific stripe patterns are confirmed in the visual area of cat brain. The neurons, which provide high level functionalities, are observed in groups on the brain surface (e.g., motor area and language area in human brain). These grouped neurons on brain surface are called the Brodmann areas [19]. The reason for formation of neuron groups by their functionalities appears to be a result of evolution, during which the neural pathways are optimized for adding new functionalities.

In terms of artificial neural network, analysis of its structure by finding shortest connection path for the signals suggests the possibility of clarifying the functionalities of the neural network. It also suggests that the functionalities of neural networks be acquired and learned locally by this clarification.

This thesis aims to decomposes a neural network to multi functions from a task by 2 approach, one is came from analysis of weight and structure, the other is output signal trace and structure creation.

# Chapter 3

# Neural Network Structure Analysis based on Hierarchical Directed Force Graph for Multi-Task Learning

## 3.1 Introduction

For multitask learning by neural networks with deep layers, encouraging classification accuracy is studied [4, 20], where lower layers close to the input layer of a neural network are formed via unsupervised learning for the purpose of feature extraction, and then the entire neural network is learned via a combination of those partially structured lower layers with higher layers. Some parameters, such as iteration times of unsupervised learning and number of neurons in each layer, have to be determined by two methods without any guidance from clear criteria. One is an empirical decision that chooses the most plausible number between the upper and lower layers. Another is a pruning method that deletes neurons connected to weak weights. The former requires an experiment and trial-and-error. The latter can determine a suitable size, which is lower than the initial size, through pruning,

but it cannot identify the role of deleted neurons in multitask. Moreover, the high learning cost of deep neural networks and time required for trial-and-error result in significant computational costs for determining a suitable neural network structure. It is necessary to visualize the neurons in a task-related pattern as guiding criteria for adjusting network size.

Therefore, a hierarchical force-directed graph drawing is proposed for neural network structure analysis by creating neuron clusters in 2D Euclidian spaces, in which the placement of neurons is determined using connection weights between neurons. Explained in detail, an attractive force acts on neurons located on two adjacent layers, and a repulsive force acts on neurons in each layer. The force applied between neurons is used to update neuron lactations during iteration, and constructed neuron distributions are defined as the neuron cluster. The variance of neurons in clusters that react to input data is designed as the criteria for adjusting the number of neurons in each layer.

By applying our proposal, those with insufficient experience designing neural network structures can check the clusters of visualized neurons and use their variance as the criteria for determining the neural network structure. Given a specific task, the trial-and-error process for determining its network structure also benefits from a lower computational cost.

The proposal is evaluated by experiments with neural networks learned on the Mixed National Institute of Standards and Technology (MNIST) database that contains 70,000 handwritten digits of ten categories. In the first experiment, the average variance of the neurons in clusters that react to the input data is calculated in each layer to show its relationship with the status of the learning process. In the second experiment, the training data of one specific task (i.e., digit “1”) is cut in half, and the variance of neurons in

clusters that react to this task is calculated in the middle layer, which is adjacent to the input layer of the network, and the value is compared with the average variance of neurons in the cluster that corresponds to all tasks in the same layer.

The visualization method for showing neuron clusters in each layer and the evaluation criteria for network size are proposed in Section 2. Evaluation experiments that use handwritten digits to show the relationship between evaluation criteria with each task are covered in Section 3.

## 3.2 Neuron cluster expressed by hierarchical force-directed graph

It is described how to analysis neural network with force-directed graph in 3.2.1, and neurons' reaction expressed by the proposal is also explained in 3.2.2.

### 3.2.1 Visualizing neurons into clusters by hierarchical force-directed graph drawing

To clarify the correspondence between neural network processing and its structure, the relationship between neurons in each layer is expressed in the 2D Euclidian space by assigning their location using a force-directed graph drawing [4], and the proposed visualization in Fig. 3-1 is utilized to generate a clustering algorithm for a hierarchical structured network.

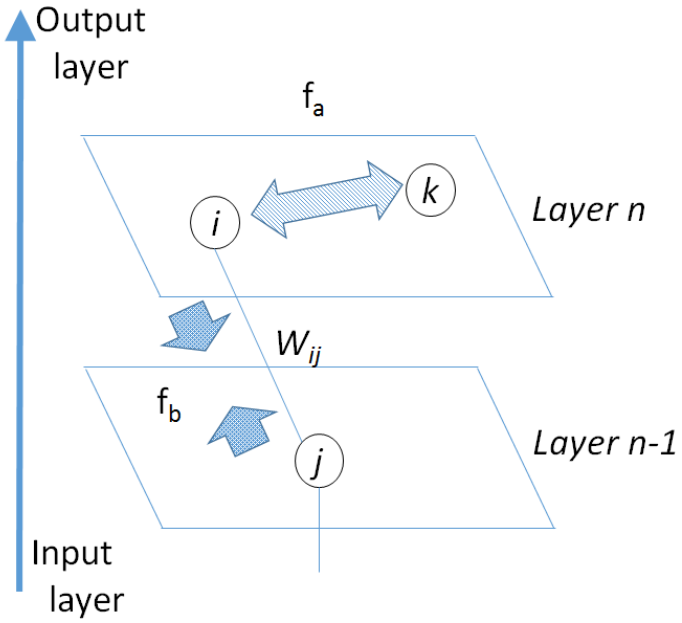



Fig. 3-1 Neuron cluster in 2D Euclidean representation of each layer in neural network

As a preprocessing step, the weights in the neural network are normalized. Then, the following two steps are iterated until the neuron movement velocity is reduced to a certain value.

(1) Forces related to neural network weights are calculated and applied to neurons.

(2) Neuron velocity and location are updated using applied forces.

First, for equitable comparison of neurons with different ratio between their weights and biases, the connection weight $W_{ij}$ between neuron $i$ and neuron $j$ is normalized using bias $b_i$ of neuron $i$ as

$$W_{ij} \coloneqq \frac{W_{ij}}{1+I(b_i)}, \tag{3.1}$$

$$I(b_i) = \begin{cases} b_i & if \quad b_i > 0 \\ 0 & otherwise \end{cases}. \tag{3.2}$$

This normalization process is based on the idea that neuron firing is discriminated by whether its value is higher than the bias. If $b_i$ is less than or equal to zero, neuron $i$ fires regardless of the state of neuron $j$. However, if $b_i$ is a positive value, the ratio of $W_{ij}$ to $b_i\neg$ is important in deciding the state of neuron $i$.

Second, an attractive force is applied on neuron pairs in adjacent layers, and a repulsive force is applied on neuron pairs in the same layer. For two neurons on adjacent layers with weight $W$ and distance $d$, their connection is regarded as a stretched spring. Attractive force $F_a$ is calculated based on Hooke's law as

$$F_a = \begin{cases} -Wd & if \quad d < d_1 \\ -Wd_1 & otherwise \end{cases}, \tag{3.3}$$

where $d_1$ is a limitation that prevents excessive increase in the velocity and divergence of neuron positions, and $d_1$ is set to one from experience. The repulsive force $F_b$ applied on neuron pairs on the same layer is calculated in reference to the cubic function approximation of the van der Waal force as

$$F_b = \begin{cases} \left(\frac{5}{4}d^3 - \frac{19}{8}d^2 + \frac{9}{8}\right) & if \quad d < 1 \\ 0 & otherwise \end{cases}, \tag{3.4}$$

Force $F_b$ is designed as a local repulsive force that works in a radius range of one. If the distance between two neurons is over the range, the value of $F_b$ is set to zero in order to avoid a negative value.

Finally, the velocity and position of all neurons are updated using (3.3) and (3.4) by

$$V := \theta\left[V + dt\left(F_a + F_b\right)\right], \tag{3.5}$$

$$x := x + dtV\ , \tag{3.6}$$

where $\theta$ is a decay coefficient, and $dt$ is a short period of time.

Neuron distributions are suitably constructed as clusters by iterating (3.3) to (3.6) until velocity change decreases to a termination threshold. In this paper, $\theta$, $dt$, and the termination threshold are set to 0.7, 0.1, and 0.01, respectively.

### 3.2.2 Neural Network Structure Evaluation from its Cluster

The size of a neural network and its learning process are estimated using neuron clusters in the 2D Euclidian space obtained in Section 3.2.1.

Neural network processing is supposed to be performed by firing the neurons that react to an input signal, and these neurons pass their information to the neurons in the upper layer via positive weights. At the same time, the information via negative weight is used to inhibit other neurons. Then, the weights converge to specific values during the learning progress.

In the neural network that learned a task, the neurons that react to the task receive stronger information from a neuron in the lower layer, and these neurons are located in the vicinity and create clusters. For the same reason, the neurons not related to any tasks are connected by weak weights and are removed from the clusters. Therefore, the neural network size in each layer is estimated by the variance of the neurons in the clusters that react to the tasks. Furthermore, the neurons located sparsely around the clusters are marked as unnecessary in the estimation process.

## 3.3 Neural network estimation experiments from cluster reaction

The proposal is evaluated by the clusters in a visualized neural network that already learned multitask, where multitask means the serial data, and each task is managed individually. To evaluate the proposed estimation methods for learning the progress and size of neural networks, we confirm that the neurons located sparsely around clusters are not necessary because these neurons do not react to any tasks, and the variance of neurons in a cluster that reacts to a task is reduced in accordance with the decrease in learning error. In addition, the relationship between each visualized cluster and its corresponding task is also confirmed by an experiment in which training data on one specific task is cut in half, and the variance of neurons in the cluster that reacts to that task is studied.

The MNIST database [21] commonly used in multitask learning is an image database of handwritten digits, as shown in Fig. 2. In the database, each example has a 784-pixel (28 × 28) binary image and number labels (from zero to nine). The database has a training set of 60,000 examples, and a test set of 10,000 examples. MINST is a subset of a larger set available from NIST. Each digit has been normalized in size and centered in an image with fixed size. In this paper, one task is characterized by identification of the number label to each input image, and 10,000 examples are retrieved from the training data and used as validation data to evaluate the variance of neurons in the 2D Euclidean space, and calculate the learning error that represents learning progress.

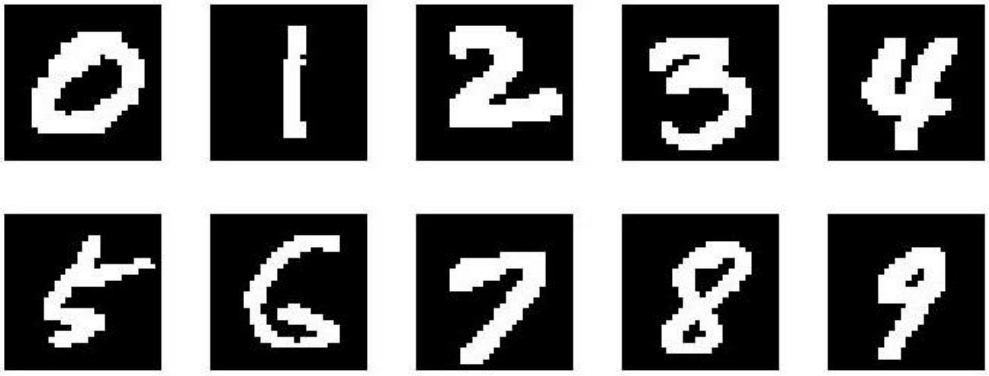


**Fig. 3-2** 10 number image examples in MNIST dataset

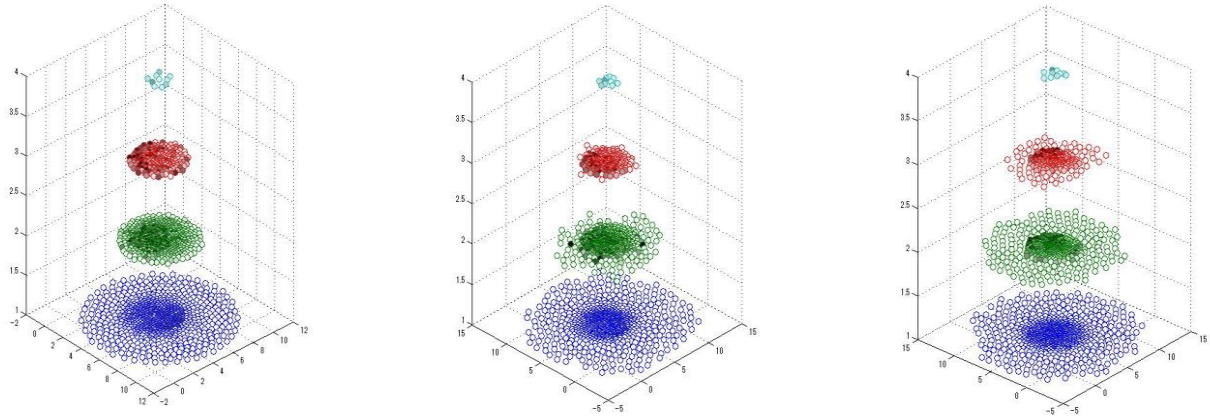
A denoising autoencoder [17] is used as the neural network model for multi-task learning. In this paper, learning rate, learning decay, momentum, and mini-batch size are set to 1, 0.98, 0.5, and 50, respectively. The number of neurons is fixed to 784 in the input layer and ten in the output layer.

### 3.3.1 Neural Network Structure Estimation from its Cluster

The neural network structure and its learning progress are evaluated through experiments confirmed by the reaction of the neuron cluster expressed in the 2D Euclidean space, where the four-layered neural network is used to verify correspondence between clusters in upper and lower middle layers.

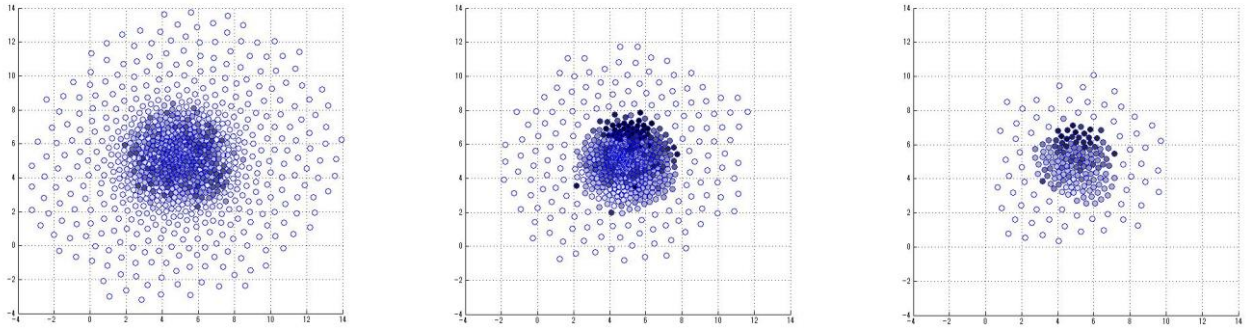
Before evaluating the proposal, we show how a neural network is visualized during the training set learning in Fig. 3-3. In Fig. 3-3, three neural networks consist of the same size 784-500-200-10, and Figs. 3-3(a), 3-3(b), and 3-3(c) are in the different epochs of one, five, and 50 trial times where the neural network is learning the training set. The neurons are

divided into four layers arranged in order from the input layer side, and the clusters are visualized circles filled in monochrome according to the firing frequency that reacts to the validation set. (In other words, neurons that do not react to any tasks are painted in white. Conversely, neurons that react independently of the task are painted in black.)



(a) (b) (c)

**Fig. 3-3** Euclidean positions of neurons, (a)epoch=1 (b)epoch=5 (c)epoch=50



(a) (b) (c)

**Fig. 3-4** Clusters on two middle layers
(a) the input layer, (b) the lower middle layer which is adjacent to output layer, (c) the upper middle layer which is adjacent to input layer

Through the learning progress shown in Figs. 3-3(a) to 3-3(c), specific neurons fire more frequently, which is observed by that the neurons in center is darker than other neurons in the surrounding. In addition, the weight between inactive neurons not related to any tasks becomes lower, which is observed by the white-colored neurons located sparsely around the center, and their arrangement becomes uneven given that their weight is not sufficient to work as an attraction force (too weak). From these results, the trend of changes in the neural network state can be represented in the 2D Euclidean space.

In the first experiment, visualization of the neural network is evaluated by visualizing the neural network according to the neuron reaction to all tasks (i.e., the digits from “0” to “9”). Furthermore, the relationship between the error rate and variance of the neurons on each layer (strongly related with suitable neural network size) is shown with a learning progress.

To discuss more details of the relationship between the visualized neural network and its size in each layer, three layers (one input layer and two middle layers) of the visualized neural network from Fig. 3-3(c) are shown in Fig. 3-4 for clarity. Figs. 3-4(a), 3-4(b), and 3-4(c) are the input layer, lower middle layer, and upper middle layer, respectively. In the two middle layers, neurons with similar color depth are arranged collectively, and their cluster approximates a slightly distorted round shape. On the other hand, neurons in the input layer have uniform color in the center, and they are not constructed as a clear cluster, but as a nearly round form. The reason for this is that the neurons that correspond to the center of the input image react equally to any tasks and are connected evenly to the neurons in the lower middle layer. In addition, the neurons that correspond to the corners of the

input image are removed from the clusters.

To summarize, we confirm by the color gradation in clusters that neurons with similar firing frequency on tasks are arranged in a group as a cluster, and that neural network size is thus discriminated to be smaller in the given case.

Furthermore, to show the relationship between neuron clusters and the learning process, the average variance of the neurons in clusters that react to all tasks is calculated in the two middle layers for epochs one to 12, and the trend of the variance obtained is compared with that of the learning error calculated by supervised data based on the standard cross-validation method shown in Fig. 3-5. In Fig. 3-5, the variances of both middle layers are gradually reduced after an initial rise on their curves, and the learning error also decreases gradually with some local oscillation. However, the variance tendency decrease in the upper middle layer is more remarkable than that of the lower middle layer.

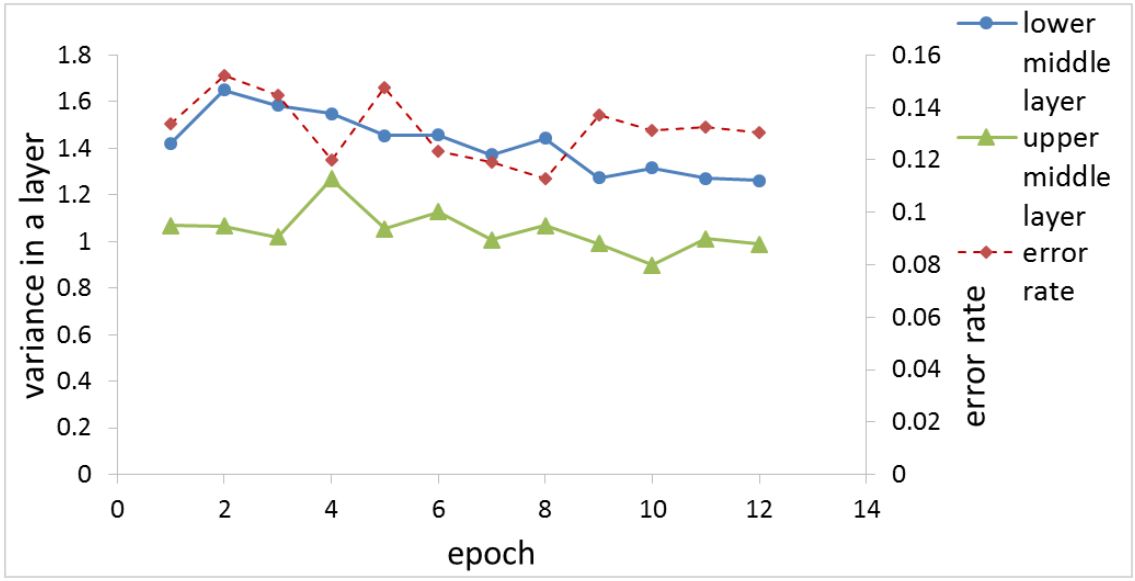
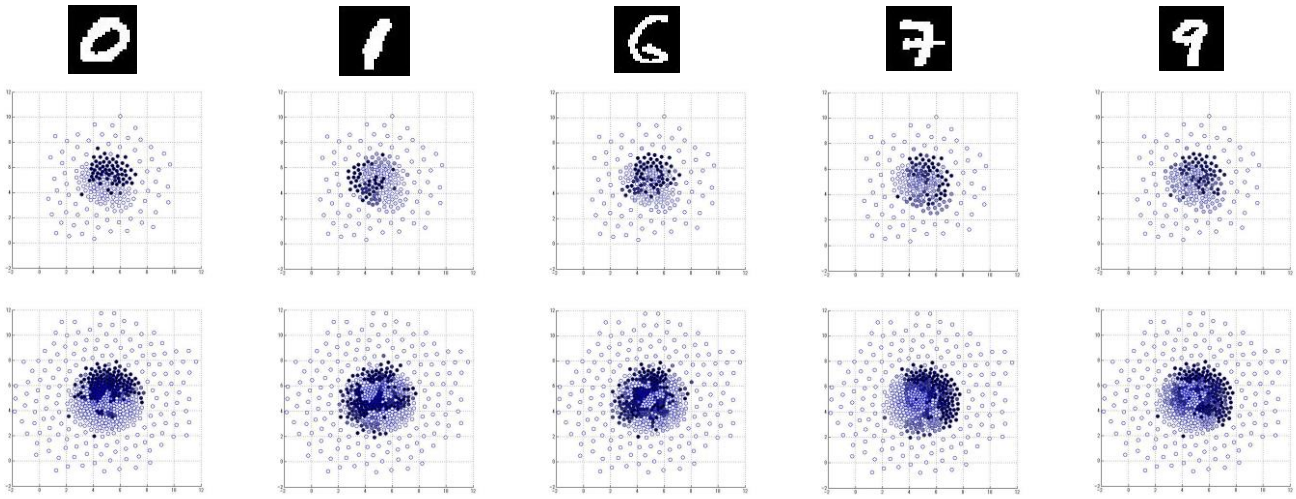



**Fig. 3-5** Epoch-variances of neurons in each layer compared with the error rate based on cross-validation

This is attributed to the characteristic of the back propagation learning method, in which weight changes are sufficiently weak in layers farther from the output layer.

To summarize, the overall trend of the learning progress can be presented indirectly by the variances of neurons in each layer without help of supervised data, but only an approximate change in the learning progress can be obtained using variance as criteria.

In the second experiment, reaction of neurons to each task is evaluated by neuron clusters, where *task0* to *task9* correspond to identifications of the digit numbers from "0" to "9," respectively. Furthermore, neuron clusters that react to one specific task and their neuron variance trend are shown with reduced training examples by comparison of (a) and (e) in Fig. 3-6.



(a) (b) (c) (d) (e)

**Fig. 3-6** Neuron clusters in one neural network which are colored according to its reaction to each specific 5 task,
5 images in the first row are examples of input images of each number task,
5 neuron clusters in the second row are expressed reacting neurons to each task in an upper middle layer, and 5 neuron clusters in the third row are expressed reacting neurons to each task in a lower middle layer.
(a) task0, (b) task1, (c) task6, (d) task7, (e) task9

**Table.3-1** The error rate in each task in Fig.3-6

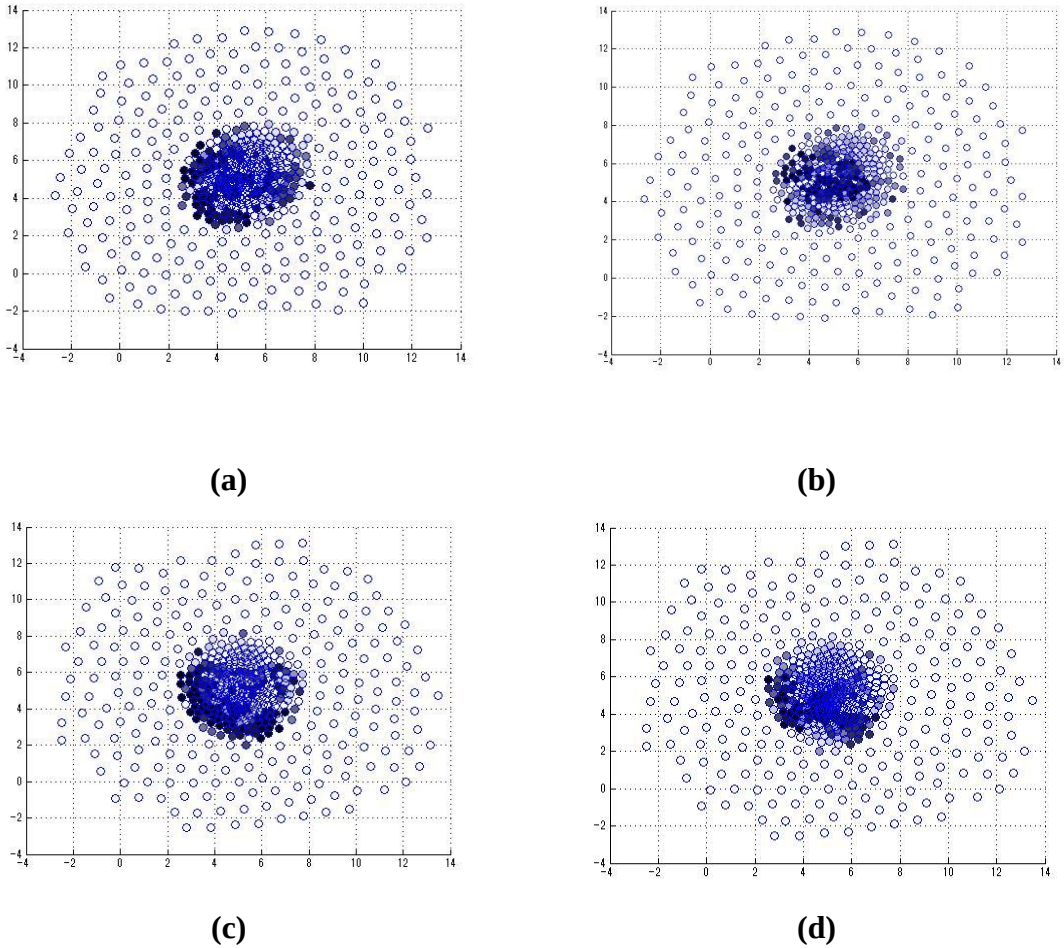
| *task0* | *task1* | *task6* | *task7* | *task9* |
|---|---|---|---|---|
| 0.0245 | 0.0317 | 0.0428 | 0.1031 | 0.0763 |

To observe the relationship between neuron clusters and tasks, the neurons that react to specific tasks in the two middle layers of Fig. 3-3(c) are shown in Fig. 3-6, where Figs. 3-6(a), 3-6(b), 3-6(c), 3-6(d), and 3-6(e) express the neurons that react to *task0*, *task1*, *task6*, *task7*, and *task9*, respectively; a higher reacting frequency is indicated with lower color depth. In addition, the learning error that corresponds to each task in Fig. 3-6 is also listed in Table 1. By checking the mutual distances of reacting neurons in the lower layer in Fig. 6, the neurons that react to a specific task are arranged into the same cluster. In particular, the neurons that react to *taks0*, which share the lowest learning error in Table 3-1, are clearly divided by neuron color depth in Fig. 3-6(a). On the other hand, the neurons that react to *task7*, which are of the highest learning error in Table 3-1, form a dispersive cluster in the 2D Euclidean space, especially in the upper middle layer.

The two neuron clusters that react to *task7* and *task9*, for which both handwritten digits are similar in their visual perception, share an overlapping part intermediately in the lower middle layer. On the other hand, the two neuron clusters that react to *task0* and *task1*, for which both handwritten digits are different in their visual perception, do not have a significant overlapping part. Thus, the *task6* cluster of reacting neurons in the lower layer is similar to a combined cluster by *task0* and *task1*, and it is consistent with the shape of the handwritten digits. However, in the upper layer, three clusters are not overlapped, which means that clusters in the upper layer are more specific to the corresponding tasks than in

the lower layer. We also confirm that the neurons located sparsely around the center are unrelated to any tasks, and thus such neurons are possible candidates for deletion. We conclude that neuron clusters configured by learned neurons are constructed from lower to upper layers that correspond to the identification process of handwritten digits, and that neuron clusters configured by neurons under learning are detected when the variance of the neurons in the cluster becomes large, e.g., “7” in Fig. 3-6.

In order to check how neuron clusters change in accordance with the size of the training data for related tasks, the neuron cluster in the lower middle layers of two neural networks trained on the entire training set and lacking training set, respectively, are shown in Fig. 3-7. Figs. 3-7(a) and 3-7(b) show the neuron clusters that react to *task0* and *task1* in the same neural network trained on the entire training set (50,000 images of handwritten digits) in 12 epochs. Figs. 3-7(c) and 3-7(d) show the neuron clusters that react to *task0* and *task1* in the neural network trained on a reduced training set (47,500 images of handwritten digits with images from digit “1” cut in half, namely, 2,500 images of “1”) of 12 epochs. By comparing Figs. 3-7(a) and 3-7(c), we can see that the neuron clusters show a similar cluster. However, in the comparison of Figs. 3-7(b) and 3-7(d), the neurons that react to task1 have a more dispersive cluster in Fig. 3-7(d) than in Fig. 3-7(b). This is because the neurons that correspond to *task1* have not learned with sufficient data, given the data shortage, in comparison to *task0*.



**(a)** **(b)**

**(c)** **(d)**

**Fig. 3-7** Neurons on 2st middle layer response to a particular task.
(a) response to *task0*, learning with whole data,
(b) response to *task1*, learning with whole data,
(c) response to *task0*, learning with lacked *task1* data,
(d) response to *task1*, learning with lacked *task1* data,

To show the change of the neuron clusters that react to all tasks compared with those that react to one specific task using the reduced training data set, the variance of the neurons in the clusters is calculated from a neural network that learned on a training set (47,500 images for ten digits) where the images for task1, namely digit "1," is cut in half (2,500 images), and it is repeated from epochs one to 12, as shown in Fig.3-8.

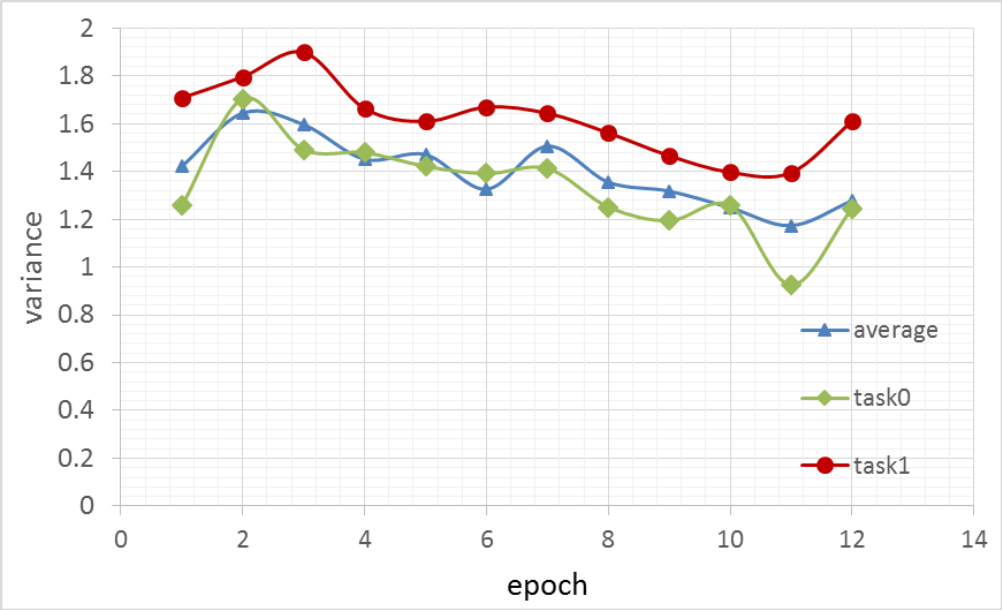



**Fig.3-8** The 3 variances of neurons in each cluster, react to fig.3-6(a), (b), and 10 tasks, in each epoch.

In Fig. 8, the average variance of neurons in the cluster that reacts to task0 shares almost the same trend with that of neurons that react to all tasks. However, the variance of neurons in the cluster that reacts to task1 is always greater than the average variance of neurons by approximately 15%. To summarize, a relationship between tasks learned by neural network is inferred by the variance of neurons in the clusters that react to specific tasks in the 2D Euclidean space.

### 3.3.2 Discussion about calculation cost and relationship with network structure

In the experiments, a period longer than 46,000 sec is required for the learning process of the neural network learning with a predefined size (784-500-250-10) under an environment where i7 CPU (3.5GHz), 8GB RAM, and MATLAB are used. Furthermore, the calculation time for visualization is longer than 282 sec in the same environment. Therefore, there is a significant computational cost when finding a more suitable network size through trial-and-error. On the other hand, it is possible to reduce the computational cost using our proposal, which detects unnecessary neurons for learning during the learning process.

Incidentally, one of the difficulties for neural networks learning multitasking is caused by task processing conducted in a black box. When performing a task using a neural network, the task is processed in cooperation with a neuron simple calculation by the summation and activation functions. Therefore, the relationship between neuron and task is unclear, and it is difficult to identify the neuron role in the network process. If neurons are associated by their processing and tasks, they can realize more efficient learning. For example, a neuron is assigned to learn a specific task, and those neurons that have completed their learning are excluded from the ongoing learning process. Furthermore, our proposal aims to be applied so that the learning process is accomplished in less time by identifying the tasks that are not advancing, and learning intensively on such tasks.

## 3.4 Conclusions

In the identification task for handwritten digits using the MNIST database, we confirmed through the characteristics of neurons in the 2D Euclidean space that the neural network size in each layer is estimated as reasonable based on whether its clusters are sparse or dense, and that the learning progress for each layer is evaluated through variance reduction in the neuron clusters that react to tasks. Furthermore, we also verified that the neuron cluster represents the characteristics of each learned task by the neuron clusters that react to that task, and the variance of the neurons in the clusters that react to a specific task is 15% larger than the average variance when the training data for that task is cut in half.

From these results, we can state that in neural network learning experiments on general character identification, the proposal clarifies the network size, learning progress, and relationship among tasks. Thus, the proposal shows the criteria that make neural network structure design easier for those with insufficient experience designing the neural network structures, and the trial-and-error process for determining network structure also benefits from lower computational cost.

The proposal aims to be developed in order to support the design of neural networks that consider multitasking of different categories by managing neuron clusters related with each task. As a long-term objective, the proposal aims to be applied to multitasking identification in the real world, such as the visual and situation cognition tasks of a robot.

# Chapter 4

# Neural Network Structure Creation based on Neuron/Connection Pruning with Reward

## 4.1 Introduction

The classification of surrounding states and acquisition of corresponding motions for robots that carry out composite tasks are the main issues when a flexible response to a rapidly changing environment is considered. Among the studies of neural networks in robotics applications for classification and motion learning purposes are methods like the preparation of multiple learners for different states [22] and the learning of motions from automatically classified input [23]. With the increase in new states and acquired motions, the ballooning of data size becomes a salient issue. The optimization of network structures is studied for alleviating computation cost. Different pruning methods by evaluating the averaged output of neurons [24, 25], the sensitivity of teacher signals [26], or normalized terms [27] have been well studied for the structural optimization of neural networks. All of these methods of pruning are designed mainly for single tasks, whereas tasks in the real

world, in contrast, are usually composite combination of single tasks with unknown relationships. When these pruning methods are introduced to networks dealing with composite tasks, there is the possibility that learned network structures may be damaged. To solve this problem, the portion of network structures related to individual tasks has to be modified by removing unwanted neurons and connections.

We propose a pruning method and its visualization method. The proposed pruning decomposes neural network structures that correspond to each task, and the proposed visualization shows its relationship via the arrangement of neurons in 2D space. Concretely, rewards and lifetimes are defined for pruning purposes. The generated reward is only given to the neurons that are related to the generation of reward. The lifetime of neurons is updated using the given reward so that neurons and connections that hinder the generation of rewards are removed. The concept of neuron space is also defined and neurons are allocated in the neuron space. Concretely speaking, neurons converge or spread according to their mutual relevance so that structures of neural networks related to specific tasks are displayed in 2D neuron space. Using the pruning method of rewards and lifetimes, structures that are locally suited from a composite task to individual decomposed tasks are likely to be formed. Also, by viewing the placement of neurons in neuron space after learning, the network structure of a composite task is understood intuitively.

To verify that the proposed pruning method does not hinder the process of learning, a comparison of convergence between the Back Propagation (BP) method and the BP method with the proposed pruning method is carried out. Comparison experiments are also done between the proposed pruning method and the Dynamic Node Decaying Method (DNDM)

[24, 25] to determine the accuracy of the proposal in the learning process of neural networks. The UCI dataset [28] is used as the benchmark for these experiments. To verify the visualization of composite tasks by the proposal, two tasks are selected from the UCI dataset and combined to form a composite task. The composite task is learned using the BP method with proposed pruning method. The composite task is evaluated using averaged task ratios of single tasks in each cluster formed by k-means. Based on the BP method with the proposed pruning method and the visualization method of neurons, an experiments on the learning of walking motion using a six-legged robot that is installed in the environment of PhysXTM [29] is also carried out to confirm the applicability of the proposal in the area of autonomic robot learning.

In section 2 of this paper, the visualization of the structure of neural networks and pruning based learning methods using rewards and lifetimes are introduced. Experiments on UCI datasets and the learning of a simulated six-legged robot are done in 3. Conclusions are presented in 4.

## 4.2 Structure creation based on pruning to non-related neurons to tasks

Two proposed methods, visualization and pruning for network decomposition, are explained in this section. A flowchart of the system used in this experiment is depicted in Fig.1. The proposed methods can be applied among existing learning methods.

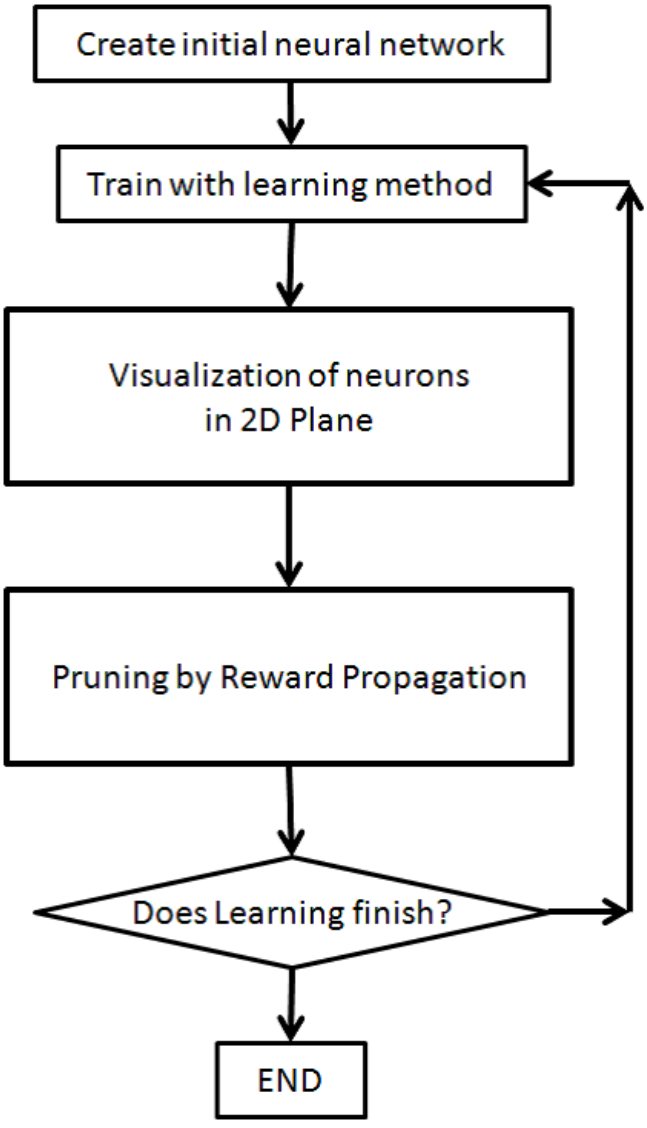



**Fig.4-1** Flowchart for neural networks with two proposed methods.

### 4.2.1 Arrangement of neurons

2D Euclidean space in which neurons are allocated is defined as neuron space. To visualize network structures, each neuron is given position vector $\mathbf{x}$, and arranged in neuron space. For human visual perception, $\mathbf{x}$ has two elements, $\mathbf{x}^1$ and $\mathbf{x}^2$.

Initial location $\mathbf{x}_{i(0)}$ of neuron *i* is given randomly. And location $\mathbf{x}_{i(t)}$ at time *t* is updated by adding $\Delta\mathbf{x}_{i(t)}$ as

$$\mathbf{x}_{i(t+1)} = \mathbf{x}_{i(t)} + \Delta\mathbf{x}_{i(t)}, \tag{4.1}$$

$$\Delta\mathbf{x}_{i(t)} = \delta \sum_{m \in M} \left( \gamma \frac{e_{r_{im}}}{|r_{im}(t)|} - |w_{im}| r_{im}(t) \right), \tag{4.2}$$

$$r_{im}(t) = \mathbf{x}_{i(t)} - \mathbf{x}_{m(t)}, \tag{4.3}$$

where *M* denotes the total number of neurons in the neuron space, $r_{im}$ is a vector from neuron *i* to neuron *m* in the neuron space, $e_{r_{im}}$ represents a unit vector in the same direction as $r_{im}$, and $|w_{im}|$ is the absolute value of weight between *i* and *m*. Note that if there is no connection between two neurons, 0 is assigned as the value of $|w_{im}|$. The quantity of $\Delta\mathbf{x}_{i(t)}$ is altered by adjusting two parameters, that is, attraction ratio $\gamma$ and magnification ratio of movement $\delta$. When two neurons overlap, one is moved randomly to a neighboring location.

By applying this idea, neurons with strong connections are made more adjacent and in

contrast, neurons with no connection are kept far away from each other. Neurons with a commonness in processing information therefore are concentrated in one place, while neurons with different processing information are placed away from each other. In addition, the distribution range of neurons is adjusted by $\gamma$ and $\delta$. When a new neuron is added to neuron space, connections between this new neuron and existing ones are determined by distance in neuron space.

### 4.2.2 Pruning by reward propagation and neuron/connection lifetime

For both neurons and connections, *life*(*t*) quantity is defined as time *t*. The lifetime of a firing neuron and the connection on its output side is updated as

$$life(t+1) = I(\alpha \cdot life(t) + \beta \cdot reward(t)), \tag{4.4}$$

$$I(x) = \begin{cases} 1 & (x \geq 1) \\ x & (1 \geq x > \theta) \\ 0 & (\theta \geq x) \end{cases}, \tag{4.5}$$

where parameter $\alpha(\in [0,1])$ is the attenuation rate of a lifetime, parameter $\beta(>0)$ is the recovering rate of lifetime due to a reward, $reward(t)(\geq 0)$denotes the reward of a neuron propagating through connection at time *t*, and parameter $\theta(\in [0,1])$ is the threshold for a lifetime. If lifetime *life* is below threshold $\theta$, then related neurons and connections are

deleted.

In cases where there is a large deviation in training data, it is necessary to adjust parameters $\alpha$, $\beta$, and $\theta$. By setting $\alpha$ to a value of 0.9, neurons and connections gradually attenuate if the reward is low or no reward is obtained at firing. For the same reason, $\beta$ should be more than 1and $\theta$ should be less than 0.1.

Rewards are generated in accordance with the output of a neuron network and distributed to neurons in the output layer. A neuron that has obtained its reward imparts the reward to neurons that have contributed to the output of which the reward is generated. A brief example that shows the propagation of reward is given in **Fig.4-2**. As is depicted in **Fig.4-2**, *i* is the neuron that gets the reward. The set of neurons of which output $y$ is connected to *i* is denoted as *N*. *j* and *k* are two neurons included in set *N*. When the reward is propagated from *i*, candidates for receiving the reward are chosen from set *N*. Candidate neurons are selected and get rewards according to three steps, i.e., their firing states, the weight of connections, and the similarity of firing conditions. The value of the imparted reward depends on the input to neuron $i$.

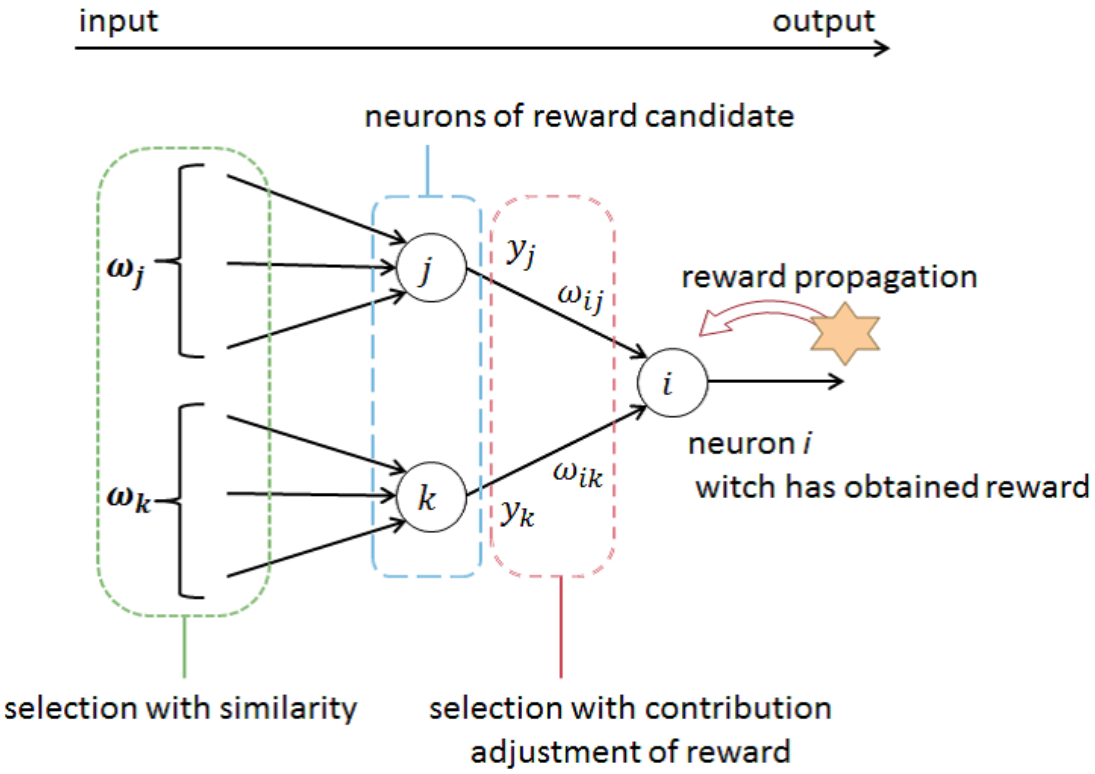



**Fig. 4-2** Example for reward propagation.

**Table 4-1** Selection of reward candidate using firing state and weight.

| | $j$ is firing | | $j$ is not firing | |
|---|---|---|---|---|
| | $w_{ij} \geq 0$ | $w_{ij} < 0$ | $w_{ij} \geq 0$ | $w_{ij} < 0$ |
| $i$ is firing | ○ | | | ○ |
| $i$ is not firing | | ○ | ○ | |

In the first step, connections that contribute to the reward acquisition of neuron *i* are selected (**Table.4-1**). If neuron *i* is firing when getting the reward, then connections with non-negative weight and with firing signal and connections with negative weight and without firing signals are selected. If neuron *i* is not firing when getting the reward, then opposite methods for connection selection are used.

In the second step, among selected connections, those with a high degree of similarity are removed. According to [30], the degree of similarity $S_{jk}$ between neuron *j* and *k* is figured out as follows,

$$S_{jk} = \frac{\left| w_j^T w_k \right|}{\left| w_j \right| \left| w_k \right|}, \tag{4.6}$$

where $w_j$ is the vector of connection weight as the input to neuron *j*. A connection weight that is not in common with neuron *k* is set to 0. If calculated $S_{jk}$ exceeds the threshold for the degree of similarity, the corresponding neuron is removed from among candidates to whom the reward is propagated.

In the third step, the exact value of the reward propagated from neuron *i* to the receiving neuron *j* is computed as

$$reward_{ji}(t) = \frac{\left| w_{ij} y_i \right|}{\sum_{n \in N'} w_{jn} y_n} reward_i(t), \tag{4.7}$$

where $w_{jn}$ is the weight of the connection between neuron *j* and neuron *n* that receives a reward, $y_n$ is the output of neuron *n*, $N'$ is the set of candidate neurons for receiving

rewards, and $reward_i(t)$ denotes the reward for neuron *i* at time *t*. $reward_{ji}(t)$ is considered to be the total amount of the reward that neuron *j* receives from neuron *i*. The reward of neuron *j* is obtained as

$$reward_j(t) = \sum_{m \in M} reward_{jm}(t), \tag{4.8}$$

where *m* denotes a neuron that gives *j* some reward, and M is the set of *m*.

Using the rules explained in **2.2.2,** the generated reward is propagated from the output layer to the input layer, and only neurons that contributed to the current task can restore their lifetimes.

## 4.3 Experiments on structure creation in the process of learning multi-task

Three experiments are conducted for evaluating our two proposed methods. In **4.3.1**, the proposed pruning method is confirmed and compared to the existing pruning method. In 4.**3.2**, the two proposed methods confirm the pruning method's visualization on 2D neuron plane. In 4.**3.3**, the proposed methods are applied to adaptation experiments on robot motion.

In these experiments, the proposal is incorporated into a three layered feed-forward neural network, using a sigmoid function and a BP method.

### 4.3.1 Pruning and learning of single task in UCI dataset

To confirm the performance of the proposed pruning method, the influence of pruning on learning is evaluated in experiments. Comparative experiments between the DNDM [24, 25] and the proposed pruning method were also carried out.

In these experiments, the UCI dataset, which is widely utilized as the benchmark for classification problems, was chosen as the single task. Three decomposed datasets were chosen from UCI dataset: a WDBC dataset (with 699 samples), a glass dataset (with 214 samples), and an iris dataset (with 150 samples).

Before experiments, the initial parameters for the neural network are set, i.e., $\alpha$ was set to 0.9, $\beta$ was set to 1, and $\theta$ was set to 0.1. In order to match with the previous studies [24, 25], the learning rate is set to 0.05, and the initial region for weights is set to ±0.1 for generating initial neurons. Each time output is generated, 0.8 times the total number of neurons in the neural network is used as the size of the reward. The reason for relating the size of the reward to the total number of neurons is that the averaged number of connections for each neuron is maintained, independent of neural networks. To show that the proposed pruning method does not hinder the learning process of neural networks, the learning time of the BP method is compared to that of the modified BP method into which the pruning method is incorporated. The WDBC dataset is used for learning. The learning is conducted for 8000 times averaged in 10 trials, and the average is evaluated.

**Fig.4-3** and **Fig.4-4** show results of experiments for the single task. **Fig.4-3** shows that the number of neurons in the hidden layer is reduced to about 8 in the early stage of the experiment, which indicates that the proposed pruning method tends to remove neurons that

do not work well in learning.

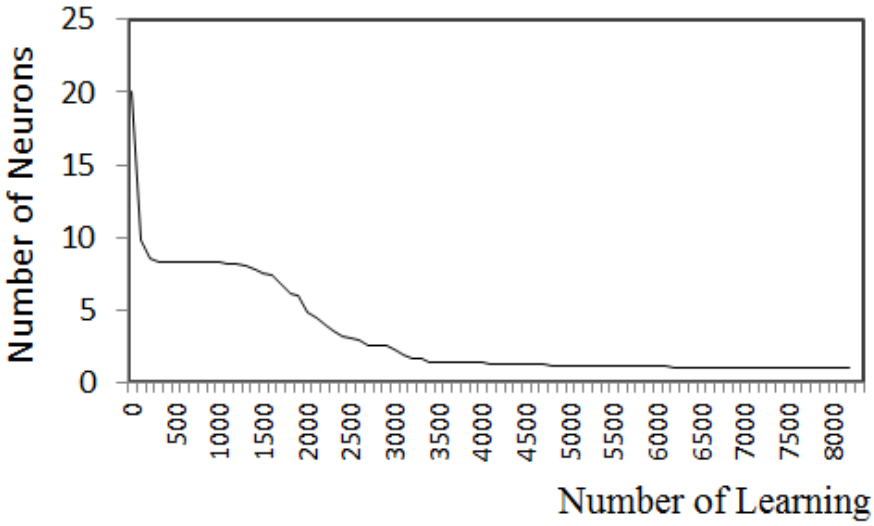



**Fig.4-3** Change of the number of neurons in hidden layer using the proposed pruning.

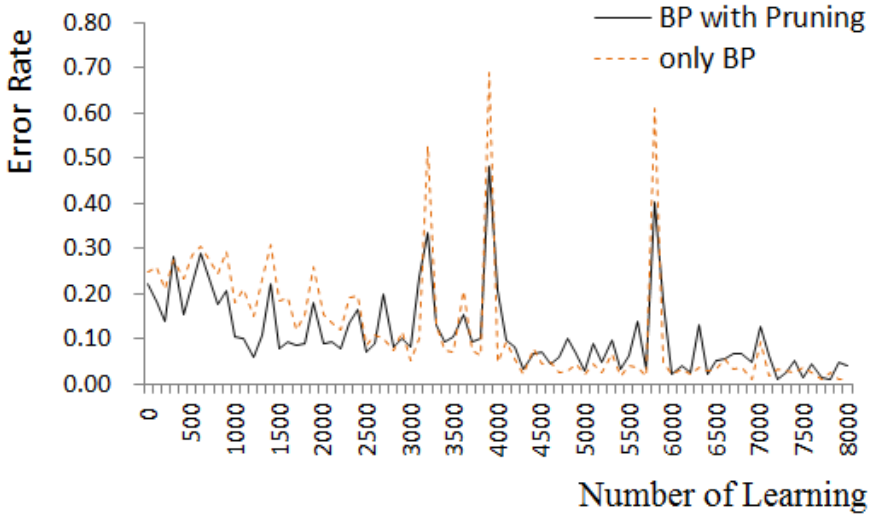



**Fig.4-4** Error rate comparison between BP method with proposed pruning and BP method alone.

In contrast, **Fig.4-4** shows that the error rate of the proposal keeps a low level compared to that of the BP method alone. The proposed pruning method thus improves the learning process of neural networks.

The BP method with the proposed pruning is compared with DNDM [24, 25] to confirm the accuracy of the proposed pruning method. Learning for the three datasets --WDBC, glass, and iris [28]-- for both methods is done in 10 trials, and the mean value of numbers of neurons in the hidden layer and variance are chosen to be the evaluation criteria as shown in **Table 4-2**.

**Table 4-2** Comparison of number of neurons in hidden layers.

| | | WDBC | Glass | Iris |
|---|---|---|---|---|
| Proposed Pruning Method | Mean | 1.0 | 6.0 | 2.1 |
| | Variance | 0.0 | 4.0 | 0.8 |
| | Min | 1.0 | 3.0 | 1.0 |
| | Max | 1.0 | 9.0 | 4.0 |
| DNDM | Mean | 2.4 | 4.0 | 2.7 |
| | Variance | 1.4 | 1.1 | 1.5 |
| | Min | 1.0 | 3.0 | 1.0 |
| | Max | 6.0 | 7.0 | 7.0 |

The BP method with proposed pruning outperforms the DNDM in terms of neurons remaining in the hidden layer for the WDBC dataset. For the glass dataset, the DNDM

works better, while for the iris dataset, the proposed pruning method again has a slightly better performance.

The reason why the proposal did not work well for the glass dataset is explained using **Fig.4-5**. The error curve of learning for the glass dataset shows that the convergence of the learning process is slow compared to that of the other two datasets, which is in turn due to an imbalance in teacher signals for the glass dataset.

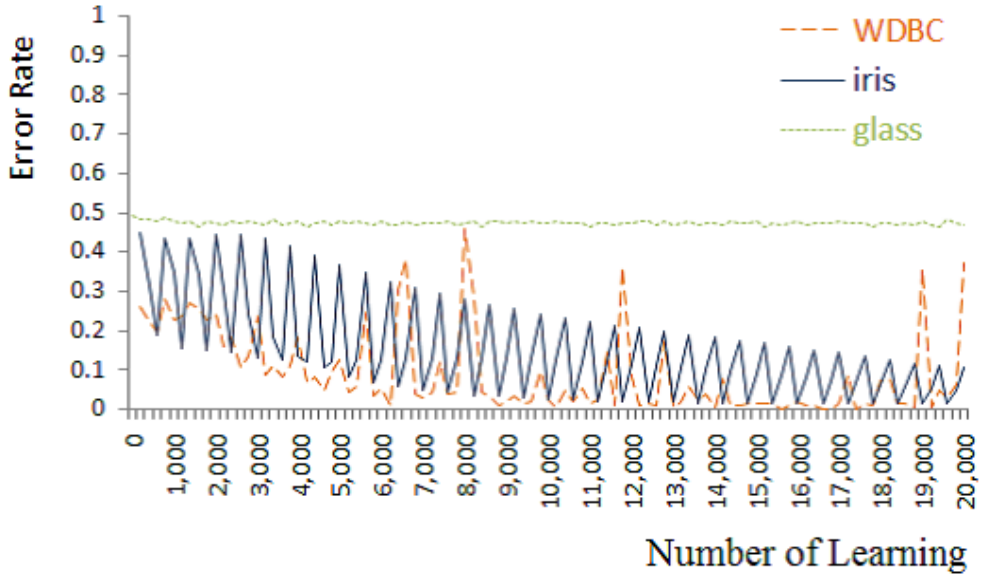



**Fig.4-5** Error in learning process for three datasets.

## 4.3.2 Visualization of composite task in UCI dataset

To confirm decomposition of the network structure, a neural network learned a composite task, combined from the UCI dataset with proposed pruning and visualization.

Evaluation is done based on neuron placement and connection status. The remaining number of neurons in hidden layers after learning is also considered. The WDBC dataset and the iris dataset used in **4.3.1** are highly mutually independent. The two datasets are combined to produce a new composite task containing 480 samples. The same initial parameters as those used in **4.3.1** are applied for generating a neural network. Parameters for the proposed pruning are carefully tuned using an attenuation rate of lifetime $\alpha$ set to 0.98, the recovering rate of lifetime $\beta$ set to 5.0, and the lifetime threshold set to 0.1, respectively. The initial number of neurons in the hidden layer is set to 20. The learning process terminates when average square error between output and the teacher signal is lower than 0.05. Each learning process is carried out for 4 trials. Results for the number of neurons in the hidden layer after learning are shown in **Table 4-3**, together with the results of single tasks from **Table 4-2**. When a composite task (WDBC+ iris) is used to learn the BP method with the proposed pruning, accuracy becomes much lower in comparison to single task settings.

After learning, neurons in the hidden layer are decomposed into two clusters based on location in neuron space. To confirm these neurons' express network structure, we compare clusters and categories of neurons. Neurons are divided into three categories by input, namely WDBC, iris, and WDBC+ iris (the composite task) and clustered into two clusters by k-means of OpenCV[31]. An example of neuron location the neuron space after clustering using k-means is shown in **Fig.4-6**. In both clusters, numbers of neurons whose input comes from a single task surpass those from the other single task.

**Table.4-3** Performance of pruning for a composite task.

| | WDBC+ iris | WDBC | Iris |
|---|---|---|---|
| Mean | 10.0 | 1.0 | 2.1 |
| Variance | 5.5 | 0.0 | 0.8 |
| Min | 8.0 | 1.0 | 1.0 |
| Max | 14.0 | 1.0 | 4.0 |

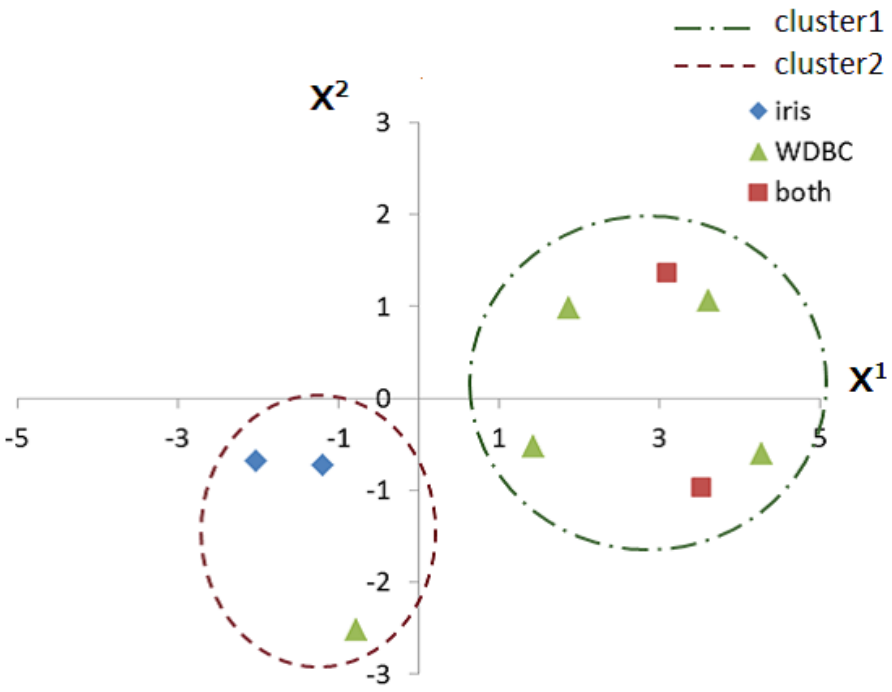



**Fig.4-6** Example of clustering in neuron space.

Task ratios for single tasks are defined as the ratios of neurons whose input is from a single task (WDBC or iris) in one cluster. The task ratio for the WDBC dataset and the iris dataset are denoted as $R_{WDBC}$ and $R_{iris}$. $R_{WDBC}$ and $R_{iris}$ are computed as

$$R_{WDBC} = \frac{N_{WDBC}}{N_{clus}}, \tag{4.9}$$

$$R_{iris} = \frac{N_{iris}}{N_{clus}}, \tag{4.10}$$

where $N_{WDBC}$ denotes the number of neurons whose input comes only from the WDBC dataset, $N_{iris}$ represents the number of neurons whose input come only from the iris dataset, and $N_{cluster}$ denotes the number of neurons in one cluster. A dataset that is related to a larger number of neurons in one cluster is considered the task for that cluster. Neural network structure decomposition is evaluated by task ratios. Average task ratios for both clusters for 4 trials are shown in **Table 4-4**.

**Table.4-4** Task ratios for the two clusters

| Trial | Cluster 1 | | Cluster 2 | |
|---|---|---|---|---|
| | $R_{WDBC}$ | $R_{iris}$ | $R_{WDBC}$ | $R_{iris}$ |
| 1 | 1.0 | 0.0 | 0.0 | 0.8 |
| 2 | 0.6 | 0.0 | 0.0 | 0.9 |
| 3 | 0.9 | 0.0 | 0.0 | 1.0 |
| 4 | 1.0 | 0.0 | 0.3 | 0.7 |

The averaged task ratio for both clusters is 0.8, which is explained the same as that for the neural network of a composite task is composed of more than 80% neurons in each task.

### 4.3.3 Visualization of tasks in neural network learned a robot motion

We expend our proposal to visualization experiments on robot motion based on a simulated six-legged robot installed in PhysX$^{TM}$ [29].

The profile of the six-legged robot is shown in Fig.7. The robot has six legs -- three on each side of its body. The three pairs are positioned on either side of the front, middle, and back of its body. Each leg has two joints -- one connecting the leg and body, and the other connecting the upper and lower part of the leg. The movable angle of each joint is set at 60 degrees.

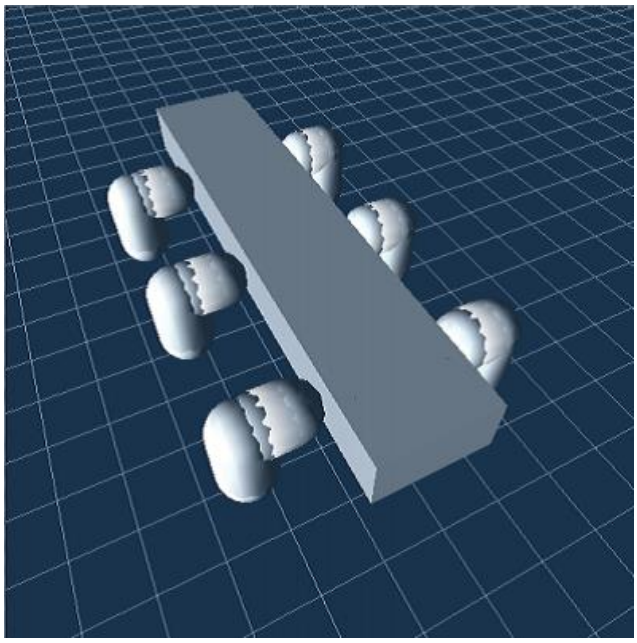


**Fig.4-7** Profile of a six-legged robot.

To assist in the motion of the robot, the legs are move alternately. Concretely speaking, the left leg at the front, the right leg at the middle, and the left leg at the back are designed to share the same motion, as do the remaining three legs. The four torque values for the two forelegs are considered to be input for the robot and the other four legs move accordingly.

A neural network is used, that has four neurons in the output layer and eight neurons in the input layer. Neurons in the output layer correspond to four inputs for the robot, and neurons in the input layer receive eight angles -- current and previous four forelegs angles -- as input.

The BP method with proposed pruning is utilized to construct the neural network for controlling robot walking. Input to the forelegs is set as tasks and locations of neurons in

neuron space are evaluated.

An example of neuron locations in neuron space after walking motions is learned is shown in **Fig.4-8**. In this case, no neurons correspond only to the right foot. In addition, neurons are densely assembled along the $\mathbf{x}^1$ axis. In other cases, neurons are centralized in a small region.

From these results, tasks for learning walking are considered mutually associated.

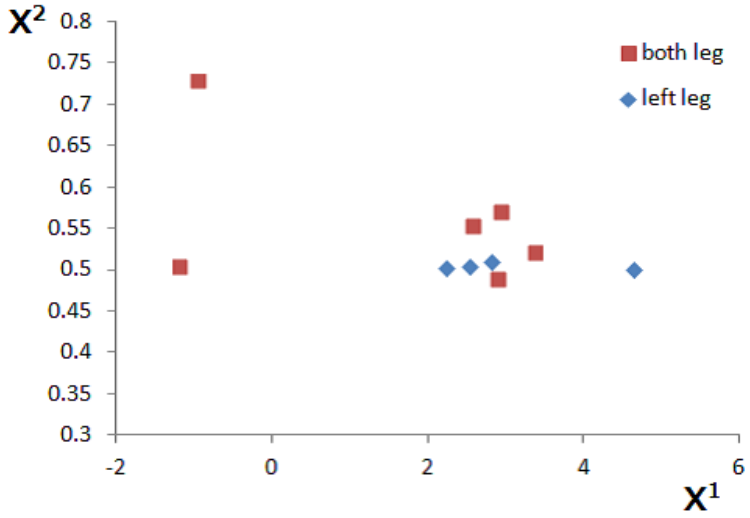



**Fig.4-8** Locations of neurons in neuron space for learning walking.

## 4.4 Conclusions

In this chapter, the proposed algorithms in chapters 2 and 3 have been represented based on alpha-level sets. The representation of each chromosome based on alpha-level sets is able to perform an efficient computation. The effectiveness of the alpha-level sets representation for the chromosomes has been demonstrated in from a theoretical point of view. Furthermore, the proposed system represented based on alpha-level sets generates a weekly schedule within 9 minutes using the algorithm implemented on a PC (Athlon XP 1900+) in C++ language under the same condition described 13 presents in chapter 3.

# Chapter 5

# Conclusions

In the chapter 3, experiments of the identification task for handwritten digits using the MNIST database has done, and it is confirmed by characteristics of neurons in the 2D Euclidean space that neural network size in each layer is estimated as reasonable based on whether its clusters is sparse or dense and that learning progress for each layer is evaluated by variance reduction in neuron clusters reacting to tasks. Further, it is also verified that the neuron cluster represents the characteristics of each learned task by neuron clusters reacting to that task and the variance of neurons in the clusters reacting to a specific task is 15% larger than the averaged variance when the training data for that task is cut by half.

In the chapter 4, experiments on learning using our proposal of a pruning method and its visualization method have been conducted using a PC with a dual core processer (2.5GHz) and 2GB memory. And programming language is in C++. The physics engine used for the simulation of a six-legged robot is PhysXTM SDK 2.8.1.

Experiments on single tasks -- WDBC, glass, and iris dataset -- from the UCI dataset using the backpropagation method with our proposed pruning method have shown that the pruning method tends to remove neurons that do not work well in learning. The backpropagation method with pruning leads to a lower error rate in comparison with the

backpropagation method alone. When the WDBC and iris datasets are learned using the backpropagation method with pruning, the mean numbers of neurons in the hidden layer are reduced by 1.4 and 0.6 in comparison with those of the DNDM method. In the case of the glass dataset, the number of neurons in the hidden layer is increased by 2.0, which was mainly due to the imbalance of the teacher signal in this dataset. The experiment on the composite task using the WDBC + iris dataset has shown that when composite task is used for learning the backpropagation method with our proposed pruning method, accuracy drops noticeably in comparison to that of single task settings. When neurons in the hidden layer after learning are clustered into two clusters in neuron space, the average task ratio for both clusters is over 0.8. Concretely, task ratios for the WDBC and the iris datasets are 0.88 and 0.85, which is evidence showing that the structure of neural networks for this composite task is well decomposed by applying our proposal. In addition, pruning methods is applied to a robot experiment and discussed of its behavior.

From through these results, function in the neural network is represented by the neurons' cluster. It provides that each function in neural network is learned or learning. This is first step of adding function to neural network by locally adjustment.

# Reference


[1] McCulloch, Warren, Walter Pitts. “A Logical Calculus of Ideas Immanent in Nervous Activity,” Bulletin of Mathematical Biophysics 5 (4), pp 115–133, 1943, doi:10.1007/BF02478259

[2] Farley, B.G., W.A. Clark, “Simulation of Self-Organizing Systems by Digital Computer,” IRE Transactions on Information Theory 4 (4), pp 76–84, 1954, doi:10.1109/TIT.1954.1057468

[3] A. K. Sharma, S. Sheikh, I. Pelczer, G. C. Levy, “Classification and Clustering: Using Neural Networks,” J. Chem. Information Computer Science 34 (5), pp 1130-1129, 1994. DOI: 10.1021/ci00021a019

[4] Y. Bengio, P. Lamblin, D. Popovic, H.Larochelle, “Greedy Layer-Wise Training of Deep Networks”, Advances in Neural Information Processing Systems 19 (NIPS'06), pp 153-160, 2007.

[5] Grégoire Montavon, Mikio L. Braun and Klaus-robert Müller, “Deep Boltzmann Machine as Feed-Forward Hierarchies,” Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12), pp 798-804, 2012

[6] Paul Merolla, John Arthur, Filipp Akopyan, Nabil Imam, Rajit Manohar, Dharmendra S. Modha, “A Digital Neurosynaptic Core Using Embedded Crossbar Memory with 45pJ per Spike in 45nm,” Custom Integrated Circuits Conference, pp 1-4, 2011.

[7] Steve K. Esser, Alexander Andreopoulos, Rathinakumar Appuswamy, Pallab Datta, Davis Barch, Arnon Amir, John Arthur, Andrew Cassidy, Myron Flickner, Paul Merolla, Shyamal Chandra, Nicola Basilico, Stefano Carpin, Tom Zimmerman, Frank Zee, Rodrigo Alvarez-Icaza, Jeffrey A. Kusnitz, Theodore M. Wong, William P. Risk, Emmett McQuinn, Tapan K. Nayak, Raghavendra Singh, and Dharmendra S. Modha “Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores,” Neural Networks (IJCNN), The 2013 International Joint Conference on, pp 1-10, 2013.

[8] G. E. Hinton, Y. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” Neural Computation 18 (7), pp 1527-1554, 2006.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” ILSVRC 2014.

[10] R. Caruana, “Multitask Learning”, Machine Learning, vol.28, 1997.

[11] Ronan Collobert, Jason Weston, “A unified architecture for natural language processing: deep neural networks with multitask learning”, Proceedings of the 25th international conference on Machine learning, pp 160-167, 2008.

[12] Hal Daumé III, “Frustratingly Easy Domain Adaptation”, ACL 2007.

[13] J. McCarthy, P. J. Hayes,“Some philosophical problems from the standpoint of artificial intelligence”. Machine Intelligence 4, pp463-502. 1969.

[14] Xiao Luo, A. Nur Zincir-Heywood, “Evaluation of Two Systems on Multi-class Multi-label Document Classification”, Foundations of Intelligent Systems Lecture Notes in Computer Science Volume 3488, 2005, pp 161-169.

[15] Yichuan Tang, Ruslan R Salakhutdinov, “Learning Stochastic Feedforward Neural Networks”, Advances in Neural Information Processing Systems 26 (NIPS 2013)

[16] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion”, The Journal of Machine Learning Research 11, pp 3371-3408, 2010.

[17] R. B. Plam, “Deep Learning Toolbox”

https://github.com/rasmusbergpalm/DeepLearnToolBox

[18] Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, Andrew Y. Ng "Building High-level Features Using Large Scale Unsupervised Learning", Appearing in Proceedings of the 29 th International Conference on Machine Learning, Edinburgh, Scotland, UK, 2012.

[19] Brodmann K. "Vergleichende Lokalisationslehre der Grosshirnrinde," Leipzig: Johann Ambrosius Barth, 1909.

[20] Y. Huang, W. Wang, L. Wang, and T. Tan, "Multi-task deep neural network for multi-label learning", 20th The International Conference on Image Processing, 2013.

[21] Y. LeCun, C. Cortes, and C. J. C. Burges, "THE MNITS DATABASE of handwritten digits": http://yann.lecun.com/exdb/mnist/

[22] K. G. Jolly, R. Sreerama Kumar, R.Vijayakumar, "Intelligent task planning and action selection of a mobile robot in amulti-agent system through a fuzzy neural network approach," Engineering Applications of Artificial Intelligence Vol23, pp.923-933, 2010.

[23] M. Okada, D. Nakamura, Y. Nakamura, “Self-organizing Symbol Acquisition and Motion Generation based on Dynamics-based Information Processing System,” The Japanese Society for Artificial Intelligence, Vol20, No.3, SP-A, pp.177-186, 2005.

[24] M. Shahjahn, K. Murase, “A Dynamic Node Decaying Method for Pruning Artificial Neural Networks,” IEICE TRANS. INF. and SYST., Vol.E86-D, No.4, pp.736-751, April 2003.

[25] M. Shahjahn, K. Murase, “A Pruning Algorithm for Training Cooperative Neural Network Ensembles,” IEICE TRANS. INF. and SYST., Vol.E89-D, No.3, pp.1257-1269, March 2006

[26] S. Kikuchi, N. Nakamura, “Recurrent neural network with short-term memory and fast structural learning method,” System and Computer in Japan, Vol.34, No.6, pp.69-79, 2003.

[27] Z. Zhang, J. Qiao, “A Node Pruning Algorithm for Feedforward Neural Network Based on Neural Complexity,” International Conference on Intelligent Control and Information Processing, Dalian China, pp.406-410, August 13-15, 2010.

[28] C. B. D. J. Newman, S. Hettich, C. Merz, “UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/mlearnMLRepository.html,” 1998.

[29] http://www.nvidia.co.jpobject/physx_new.html.

[30] N. Higa, H. Shirotsuchi, “Studies on An Algorithm for Reducing Redundant Units Based on Geometrical Approach,” The Institute of Electronics, Information and Communication Engineers, technical report (in Japanese language), pp.43-48, June, 2002.

[31] http://opencv.org/

# Acknowledgements

In the first, I would like to express my indeed grateful to my supervisor Professor Kaoru Hirota for his valuable comments and encouragement, which nurtured my research spirit. Also I will like to express my sincere gratitude to Professor Takanori Shibata, Professor Osamu Hasegawa, Professor Toshiaki Murofushi, and Professor Fangyan Dong for their priceless time, valuable comments and advices that have guided me through my doctoral thesis.

I would also like to say thank to my previous supervisor Professor Shigeaki Sakurai for helpfulness in his busy time. His facing seriously in my clumsy research has cheered me up always.

I would like to express my gratitude to a laboratory member and staff, Harumi Hoshino, Toshihiro Akamatsu, Che-Hung Lin, Jiajun Lu, Garcia Sanchez Jesus Adrian, and all of other members for not only suggestion about research, but also helpful advices about my life.

I am very grateful to my graduated laboratory members, Kazushi Okamoto, Martin Tangle, Yuuki Saito, Masashi Yamaguchi, Nantinee Tulyanon, and Chihiro Kawabe for powerful support of research, study and emotionally.

I would like to express my thanks to all members of Hirota Laboratory. Finally, I would like to express my appreciation to my family for giving me the opportunity to carry on my research.